Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Kevin Strandenes

Development of a machine vision based stabilization and positioning system for AUVs

Master's thesis in Master of Science in Cybernetics and Robotics
Supervisor: Annette Stahl
Co-supervisor: Håkon Hagen Helgesen and Rudolf Mester
June 2023

NTNU
Norwegian University of Science and Technology

mohn technology

Kevin Strandenes

Development of a machine vision based stabilization and positioning system for AUVs

Master's thesis in Master of Science in Cybernetics and Robotics
Supervisor: Annette Stahl
Co-supervisor: Håkon Hagen Helgesen and Rudolf Mester
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Kunnskap for en bedre verden

Development of a machine vision based stabilization and positioning system for AUVs

Author:
Kevin Strandenes

Master of Science in Cybernetics and Robotics
Submission date: June 2023
Supervisor: Annette Stahl, ITK
Co-supervisor: Håkon Hagen Helgesen, ITK & Rudolf Mester, IDI

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

I am deeply grateful to my supervisor, Professor Annette Stahl, as well as my co-supervisors Håkon Hagen Helgesen and Rudolf Mester for their invaluable guidance and unwavering support throughout this research. Their expertise and feedback have shaped the trajectory of this thesis. I also extend my appreciation to the members of my thesis committee for their valuable input and suggestions.

I would like to acknowledge Mohn Technology AS for providing the necessary resources and facilities for conducting the experiments and simulations. The collaboration with industry professionals and fellow researchers in fish farming has been crucial in gaining practical insights and shaping the direction of this work.

Abstract

This master thesis focuses on the development of a robust algorithm for fishnet detection and tracking using an Autonomous Underwater Vehicle (AUV). The algorithm aims to overcome the limitations of existing approaches and address the challenges posed by underwater environments. The specific objectives include the creation of virtual fishnets as training data, robust fishnet detection, estimation of fishnet pose, junction point detection, and motion tracking of junction points. Comprehensive experiments using real-world underwater data from fish farming sites are conducted to evaluate the algorithm's effectiveness. The outcomes of this research have significant implications for underwater robotics, particularly in fish farming operations, by enhancing AUV capabilities in fishnet detection and tracking.

Sammendrag

Denne masteroppgaven fokuserer på utviklingen av en robust algoritme for deteksjon og sporing av not ved hjelp av et autonomt undervannsfartøy (AUV). Algoritmen har som mål å overkomme begrensningene til eksisterende metoder og takle utfordringene som oppstår i undervannsmiljøer. Spesifikke mål inkluderer å lage virtuelle not som treningsdata, pålitelig deteksjon av not, estimering av notes posisjon, deteksjon av knutepunkter og sporing av bevegelse for knutepunktene. Omfattende eksperimenter med undervannsbilder fra fiskeoppdrettsanlegg brukes for å evaluere algoritmens effektivitet. Resultatene av denne forskningen har betydelige implikasjoner for undervannsrobotikk, spesielt innen fiskeoppdrett, ved å forbedre AUVs evne til å detektere og spore not.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Chapter 1

Introduction

This thesis focuses on the development of a robust algorithm for fishnet detection and tracking using an Autonomous Underwater Vehicle (AUV). Fish farming operations face challenges in effectively monitoring and managing fishnets in underwater environments. By addressing these challenges, the proposed algorithm aims to enhance the AUV's navigational capabilities and enable efficient interaction with fishnets.

1.1 Background

This section provides a comprehensive overview of the background information relevant to the research topic. It explores the significance of fish farming as a crucial industry for meeting the global demand for seafood. The challenges associated with fish farming, such as overfishing, environmental risks, and the need for sustainable practices, are discussed. Additionally, the section highlights the emergence of Precision Fish Farming (PFF) and its potential to improve the monitoring and control of fish production. The utilization of Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles (ROVs) in fish farming operations is introduced as a promising solution. The collaborative partnership with Mohn Technology AS and the motivation behind this research are also outlined. Overall, this section sets the stage for understanding the context and importance of fishnet detection and tracking using AUVs in the field of aquaculture.

1.1.1 Fish farming

Fish farming is a form of aquaculture in which fish are raised in enclosures for food production and is an essential global industry that aims to meet the increasing demand for seafood. It is the fastest growing area of animal food production, with approximately half of the fish consumed globally being raised in these artificial environments. There are both net pens (mesh cages) submerges in natural bodies of water and concrete enclosures on land as shown in Figure 1.1. Commonly farmed species include salmon, tuna, cod, trout and halibut. While fish farming is considered by some as a potential solution to overfishing, it is not without its challenges.

According to the Food and Agriculture Organization of the United Nations (FAO), in 2019, only 64.6 percent of the fishery stocks of the world's marine fisheries were fished within biologically sustainable levels. Additionally, fish farms pose environmental risks, such as the introduction of diseases, pollutants, and invasive species, which can harm ecosystems. Sea-based fish farms are particularly susceptible to hundreds of thousands of fish escaping annually, with holes in nets being the most common cause. This, along with pollution from waste and the introduction of invasive species, are some of the environmental risks associated with fish farming. From 2010 to 2018, the fish farming industry in Norway alone experienced 305 confirmed cases of escape incidents involving Atlantic salmon or rainbow trout, with 1.96 million registered escapees. These escapes threaten

Figure 1.1: Land and sea fish farms

the genetic diversity and survival of native species. High stocking densities in fish farms lead to pollution from fish waste and uneaten food, creating poor water quality that is low in oxygen and high in ammonia. Fish farms can also attract predatory marine animals, such as sea birds and sea lions, who are sometimes harmed by aqua farmers to protect the fish [1], [2].

Despite the challenges, there is a need to utilize more exposed coastal areas for fish farming in Norway due to the growth of the salmon industry and competition with other coastal industries. However, farming in these areas presents unique challenges such as harsh weather conditions and remoteness which make it difficult to maintain reliable production. Exposed areas may feature more stable water flow conditions and be more distant from wild salmonids in coastal waters, which could contribute to reducing the negative ecological consequences of sea lice and escapees. There is a large industrial interest in enabling safe and sustainable seafood production in exposed coastal and ocean areas. The industry has focused on utilizing the limited shelter and bathymetry of outer coastal areas, and more significant technological and operational changes are required to ensure safe and reliable fish farming in exposed areas. Norwegian maritime industrial clusters and research institutions are at the forefront of innovation and competence focused on demanding maritime operations and can play a significant role in this task [3].

Fish farming is still mainly carried out manually, with many of the day-to-day operations depending on personnel experience, which creates significant risks to health, safety and the environment (HSE). Increasing the automation level of high-risk operations in aquaculture would provide significant benefits, including social, ethical, and economic aspects. With the industry moving towards more exposed locations, safety and sustainability concerns need to be addressed. Norwegian maritime industrial clusters and research institutions are leading innovation and competence in demanding maritime operations and can play an important role in developing technical solutions and operational concepts to maintain safety and ensure reliability in fish farming operations [3].

PFF [4] is an innovative concept that applies control engineering principles to fish production. Its objective is to improve a farmer's ability to monitor, control, and document biological processes in fish farms, thereby moving commercial aquaculture from the traditional experience-based to a knowledge-based production regime. The implementation of PFF requires several technological solutions, which can represent important components in future PFF applications.

There are several reasons why PFF is crucial for sustainable fish farming practices. Firstly, it can help the aquaculture industry to monitor and control the effects of emerging biological, economic, and social challenges that may influence the ability to maintain ethically sound, productive, and environmentally friendly production of fish. Secondly, as the scale of production increases, so does the likelihood that the industry will face emerging challenges that may upscale potential problems. Therefore, PFF is an essential tool for sustainable fish farming practices.

To illustrate the potential of PFF, four case studies have been defined to solve specific challenges related to biomass monitoring, control of feed delivery, parasite monitoring, and management of crowding operations. One way to implement PFF is through the use of AUVs and ROVs. These tools enable farmers to acquire the levels of knowledge, monitoring, and control required to tackle the challenges of modern fish farming. AUVs and ROVs can collect and transmit data on water temperature, dissolved oxygen levels, salinity, pH, and other key environmental factors, which can be used to predict changes in the environment that may affect fish growth and survival. Additionally, AUVs and ROVs can be used to conduct underwater inspections of fish cages, monitor

fish behavior, and detect parasites and diseases.

1.1.2 AUV

In recent years, there has been a significant rise in the utilization of AUVs due to advancements in computational power, hardware components, and control methods. These developments have resulted in more efficient modeling and control techniques, making AUVs highly reliable for practical applications such as underwater surveys, pipeline detection and mapping, data collection, and surveillance [5]. Researchers have also focused on leveraging AUVs for long-term data collection in fields like oceanography and coastal management, as demonstrated by the work of Saghafi and Lavimi in 2020 [6].

Moreover, the cost of commercially available ROVs has become more affordable, with ROVs capable of being enhanced to operate autonomously, similar to high-end AUVs. Consequently, AUVs offer a cost-effective alternative to ROVs and manual labor, reducing the need for human intervention while maintaining operational efficiency [7].

AUVs possess several advantages that make them well-suited for fishnet detection and tracking in fish farming operations. Their autonomy allows them to operate independently, carrying out tasks based on pre-programmed mission plans and navigational instructions. This capability significantly reduces the reliance on manual labor, increasing operational efficiency in fish farming operations.

Another advantage of AUVs is their maneuverability and agility in underwater environments. AUVs are designed with thrusters or propellers that provide precise control over their movements, allowing them to navigate through complex underwater structures, such as net pens. This maneuverability is crucial for effective fishnet detection and tracking, as it enables the AUV to navigate in close proximity to the nets and capture accurate data.

Furthermore, AUVs can be equipped with various sensors and imaging systems that enable them to collect detailed data about their surroundings. These sensors can include cameras, sonars, and other specialized instruments that provide information about the underwater environment, including the presence and characteristics of fishnets. This data can be used to develop algorithms and techniques for fishnet detection and tracking, improving the overall monitoring and management of fish farming operations.

However, utilizing AUVs for fishnet detection and tracking also poses certain challenges. Underwater environments present unique conditions, such as low visibility, variable lighting conditions, and complex underwater structures, which can affect the accuracy and reliability of detection algorithms. Overcoming these challenges requires the development of robust algorithms that can handle noise, occlusions, and variations in fishnet appearance.

Additionally, AUVs must operate within the constraints of limited power and onboard computational resources. Efficient algorithms and data processing techniques need to be implemented to optimize the AUV's performance and ensure real-time or near real-time fishnet detection and tracking capabilities.

By leveraging the advantages of AUVs and addressing the challenges specific to fishnet detection and tracking in underwater environments, this master thesis aims to develop a robust algorithm that enhances the monitoring and management of fish farming operations.

1.1.3 Collaboration

This thesis is a collaborative effort with Mohn Technology AS, a Bergen-based firm specializing in underwater technologies for research, fishing, and aquaculture. Mohn Technology is actively involved in the development of an AUV with advanced capabilities, specifically designed for the inspection and cleaning of net pens in fish farming operations. The collaborative partnership with Mohn Technology provides a unique opportunity to work closely with industry experts and contribute to the advancement of underwater technology.

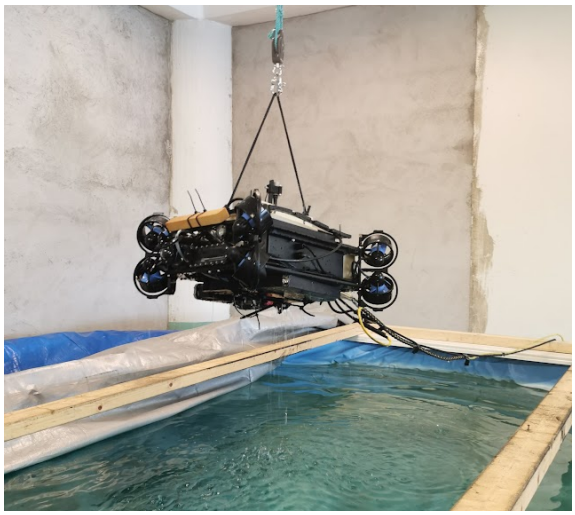


Figure 1.2: Pool testing Mohn Technology's AUV

The AUV developed by Mohn Technology is characterized by its lightweight design, battery-powered operation, and precise maneuverability achieved through eight thrusters, making it fully actuated and enabling efficient low-speed movements. The current version of the AUV is depicted in Figure 1.2, where the robot is being pulled out of their indoor testing pool. In addition to the AUV's hardware capabilities, Mohn Technology has developed a sophisticated simulator that accurately replicates underwater environments, facilitating the development of control systems. Within the scope of this master thesis, particular emphasis will be placed on the AUV's vision system, which plays a pivotal role in the detection and tracking of fishnets.

1.1.4 Motivation

The motivation behind this work stems from the need to overcome the limitations of existing approaches for fishnet detection and tracking using AUVs. The increasing demand for seafood and the expansion of fish farming activities underscore the importance of efficient and sustainable management practices. Automating the inspection and cleaning processes of net pens using AUVs holds tremendous potential to enhance productivity, reduce labor-intensive tasks, and ensure the overall well-being of farmed fish populations.

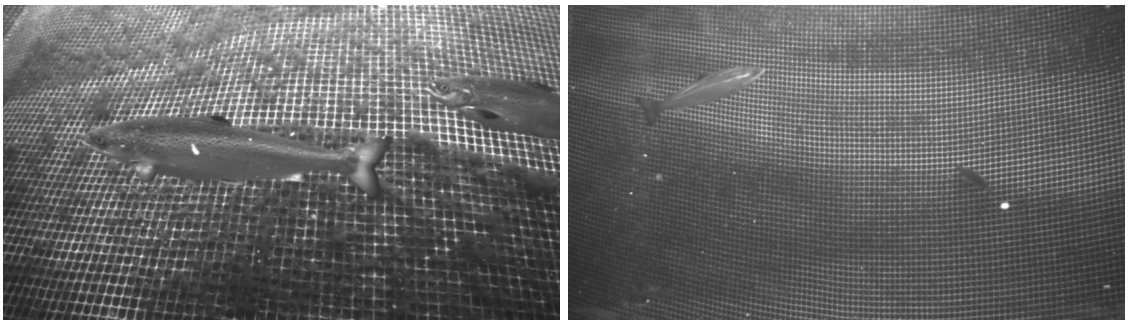
However, the detection and tracking of fishnets in underwater environments present significant challenges due to their intricate structures and the constraints imposed by low visibility and environmental noise. Current methods often fall short in providing reliable and accurate information for effective navigation and interaction with fishnets. Therefore, the development of a robust algorithm specifically designed to address the challenges associated with fishnets is crucial.

The collaboration with Mohn Technology AS and the specific focus on the vision system of their AUV further enhance the motivation behind this research. Mohn Technology is at the forefront of underwater technology development, particularly in the context of inspecting and cleaning net pens in fish farming operations. By working closely with industry experts, this thesis aims to develop a reliable and efficient algorithm that can accurately detect and track fishnets, providing essential information for the AUV's navigation, interaction, and ultimately, the development of effective cleaning and maintenance strategies.

Additionally, the creation of simulated fishnets with known parameters, such as grid spacing and camera intrinsic and extrinsic matrices, allows for accurate testing and evaluation of the proposed algorithm. This controlled approach ensures the algorithm's reliability and effectiveness in different scenarios and contributes to the advancement of fishnet detection and tracking in underwater environments.

1.2 Problem Description

This master thesis aims to develop a robust algorithm for fishnet detection and tracking using an Autonomous Underwater Vehicle (AUV). The goal is to overcome the limitations of existing approaches and address the challenges posed by underwater environments. Some of the challenging scenarios are illustrated in Figure 1.3.



(a) Fishnet occluded by fish (b) Far away fishnet

Figure 1.3: Challenging scenarios of fishnet detection

The proposed algorithm aims to provide accurate and real-time information about the location, orientation, and proximity of fishnets to the AUV. By achieving this objective, it will enhance the AUV's navigational capabilities and enable effective interaction with fishnets in fish farming operations.

The specific objectives of this thesis are as follows:

1. **Creation of Virtual Fishnets as Training Data:** Develop a methodology to create simulated/virtual fishnets with known grid spacing and specified camera intrinsic and extrinsic matrices. This process will generate accurate and controlled training data for evaluating the proposed algorithm's performance.
2. **Robust Fishnet Detection:** Develop an algorithm capable of reliably detecting the presence of fishnets in underwater scenes captured by the AUV's imaging system. This algorithm will leverage the characteristics and distinctive patterns of fishnets to accurately identify their presence, even in challenging underwater conditions.
3. **Estimation of Fishnet Pose:** Develop an algorithm to estimate the pose of the detected fishnet relative to the AUV. This algorithm will provide information regarding the fishnet's location, orientation, and proximity to the AUV's coordinate system, enabling precise navigation and interaction with the fishnet.
4. **Junction Point Detection:** Create an algorithm to identify the junction points of the fishnet. By analyzing the contour properties and connectivity of the detected fishnet regions, this algorithm should accurately locate the critical reference points necessary for fishnet tracking.
5. **Motion Tracking of Junction Points:** Implement a tracking algorithm utilizing optical flow techniques to continuously monitor the motion of the fishnet's junction points over time. This algorithm will enable the AUV to maintain proximity to the fishnet by adjusting its navigation according to the tracked movement of the junction points.

Comprehensive experiments using real-world underwater data from fish farming sites will assess the algorithm's effectiveness in various environmental conditions and scenarios. Performance metrics such as detection accuracy, pose estimation accuracy, and tracking stability will be employed.

The outcomes of this research have significant implications for underwater robotics, particularly in fish farming operations. The developed algorithm can be integrated into existing AUV systems,

enabling autonomous fishnet detection and tracking, facilitating decision-making and information gathering. Moreover, this research paves the way for future advancements in underwater robotics and contributes to the broader field of marine technology. Overall, this master thesis aims to enhance AUV capabilities in fish farming operations, contributing to the sustainability and efficiency of the aquaculture industry.

1.3 Main Contributions

This main contribution of this work is the development of an advanced algorithm that addresses the challenges associated with net pen detection and tracking in aquatic environments. Net pens are extensively utilized in aquaculture for rearing fish and other marine organisms, and accurately monitoring their position and movement is vital for effective farm management and environmental sustainability. The algorithm integrates computer vision techniques and object tracking methodologies, offering a comprehensive solution for real-time identification of net pens in video footage captured by underwater cameras.

The proposed algorithm leverages a Fast Fourier Transform (FFT) approach as its core mechanism. By analyzing the frequency content of pre-processed images, the algorithm exploits the periodic nature of fishnet structures to extract characteristic frequencies associated with net pens. This enables accurate estimation of their presence, orientation, and distance from the Autonomous Underwater Vehicle (AUV). To ensure precise identification and minimize false positives, a verification step is employed, validating the detected fishnet regions. Significantly, the proposed algorithm represents a notable improvement over a previous FFT-based approach, which is discussed in detail in the subsequent chapter, for fishnet detection. Unlike its predecessor, which was constrained to square images, the proposed algorithm can accommodating images of all sizes and shapes, which ensures that frequency content is not potentially lost by cropping an image, thereby allowing for more comprehensive and accurate fishnet detection. Additionally, it incorporates additional checks and filtering techniques on the FFT image to accurately identify the grid structure of the fishnet and evaluate the confidence level of the detection.

Furthermore, the algorithm integrates a novel junction point detection technique to enhance the estimation of the fishnet's pose alongside the FFT method. Through the utilization of a contour detection algorithm, coupled with an analysis of contour properties including shape, connectivity, and curvature, and employing clustering techniques, the algorithm accurately identifies the specific junction points where the ropes intersect within the fishnet structure. By determining the position of these junction points, the algorithm calculates the fishnet's x and y Degrees Of Freedom (DOF). This information, combined with the orientation and distance obtained from the FFT algorithm, allows for a comprehensive estimation of the fishnet's complete pose.

Furthermore, this study investigates the application of established optical flow algorithms for tracking the junction points of the fishnet, an area that has received limited attention in previous research. By continuously monitoring the movement of these reference points with respect to the AUV, the proposed method enables in-depth analysis and lays the groundwork for the development of advanced strategies in AUV-fishnet interaction. These strategies encompass various aspects, including stabilization against the dynamic fishnet, thereby enhancing overall system performance.

In summary, the main contributions of this work include:

1. The development of a novel algorithm that combines computer vision techniques and object tracking methodologies for real-time net pen detection and tracking in aquatic environments.
2. The enhancement of the FFT-based approach for fishnet detection, accommodating images of varying sizes and shapes, and incorporating additional checks and filtering techniques for improved accuracy and confidence estimation.
3. The introduction of a junction point detection algorithm that accurately identifies the intersecting junction points within the fishnet structure, enabling precise calculation of the

fishnet's x and y DOF in conjunction with the orientation and distance from the FFT algorithm, resulting in a complete estimation of the fishnet's pose.

4. The utilization of existing optical flow algorithms to track the fishnet's junction points, contributing to the analysis and enabling advanced AUV-fishnet interaction strategies.

By accomplishing these research objectives, this master thesis aims to advance the field of fishnet detection and tracking using AUVs, providing crucial real-time information for improved navigation, precise interaction, and effective management of fishnets in fish farming operations. These contributions collectively contribute to the field of aquaculture management by providing an effective solution for net pen detection, tracking, and AUV-fishnet interaction. The developed algorithm has the potential to promote sustainable aquaculture practices, enhance productivity, and mitigate the ecological impact of fish farming activities, thus advancing farm management practices and promoting environmental sustainability.

The proposed algorithm, initially designed for fishnet detection and tracking in underwater environments, has the potential to extend its application to address the challenges of underwater localization in general. Underwater localization is a complex problem due to low visibility and environmental noise. The algorithm's capability to accurately estimate the position and orientation of fishnets in challenging underwater conditions indicates its potential to contribute to the broader field of underwater localization. With further exploration and adaptation, the algorithm could enhance underwater localization techniques, benefiting underwater robotic applications and scientific research in underwater exploration and monitoring.

1.4 Thesis Outline

This thesis is structured into several chapters, each focusing on specific aspects of the research. The outline of the thesis is as follows:

Chapter 2: Previous Work

This chapter provides an overview of previous research and literature related to fishnet detection and tracking using AUVs. It examines existing methodologies, identifies gaps in the current knowledge, and highlights the need for further advancements in the field.

Chapter 3: Theory

In this chapter, the fundamental theories and concepts relevant to the research are presented. It includes an explanation of camera projection and the perspective camera model, as well as the theory behind 2D Discrete Fourier Transform (DFT) and optical flow. This theoretical foundation establishes the basis for the subsequent algorithm development.

Chapter 4: Method

The methodology chapter describes the proposed algorithms for fishnet detection and tracking. It outlines the FFT method, junction detection technique, and optical flow tracking approach. Pseudocode is provided to illustrate the algorithms' implementation. Additionally, a section is dedicated to explaining how to create a virtual grid, which simulates a fishnet, to facilitate algorithm testing and evaluation.

Chapter 5: Results

In this chapter, the results of the algorithm testing and evaluation are presented. The algorithms are applied to various datasets, and their performance metrics, such as accuracy, precision, and computational efficiency, are measured and analyzed. The findings are presented in a clear and concise manner, supported by visual representations and statistical analysis.

Chapter 6: Discussion

The discussion chapter critically analyzes the results obtained in Chapter 5. It interprets the

findings, highlights strengths and weaknesses of the proposed algorithms, and compares them with existing methods in the literature. The chapter also explores possible limitations, challenges, and potential improvements for future research.

Chapter 7: Conclusion

The conclusion chapter summarizes the main findings and contributions of the research. It provides a concise summary of the proposed algorithms' effectiveness in fishnet detection and tracking using AUVs.

Chapter 2

Previous Work

This chapter provides an overview of previous research conducted in the field of underwater localization and mapping of unmanned underwater vehicles (UUVs). Considerable research efforts have been dedicated to developing algorithms and techniques for underwater object detection and tracking using AUVs. The literature encompasses approaches employing computer vision, image processing, and machine learning methods to enhance the capabilities of AUVs in underwater environments. Various studies have focused on fish detection, fish species recognition, and environmental monitoring, while others have explored object detection and tracking for different underwater structures.

However, despite these advancements, the detection and tracking of fishnets, pose unique challenges due to their intricate and irregular structures. Fishnets can exhibit varying shapes, orientations, and spatial configurations, making their automated detection and tracking a complex task. Existing approaches often struggle to accurately identify fishnets in underwater scenes due to factors such as low visibility, ambient noise, and occlusion.

By considering the findings from previous research on fishnet detection, junction point detection, and underwater localization, future studies can explore the potential integration or comparison of these techniques to further enhance fishnet detection and tracking in underwater environments.

2.1 FFT algorithm

One notable contribution in the field of fishnet detection is the approach proposed by Schellewald, Stahl and Kelasidi [8]. Their method, based on the 2D FFT algorithm, enables an underwater robot to detect a fishnet, estimate its distance and orientation relative to the camera. The approach analyzes a region of interest (ROI) in the camera's video stream and detects regular peaks in the Fourier Transform, which indicate the presence of a fishnet. By reconstructing a single mesh from the regular peaks, the distance and orientation of the net with respect to the camera can be computed.

In this thesis, the FFT-based algorithm proposed by Schellewald et al. [8] serves as the foundation for fishnet detection. Building upon their work, further improvements and adaptations have been made to enhance the algorithm's performance in accommodating different image conditions and providing more accurate distance and orientation estimations.

2.2 Junction point detection

One paper that addresses the detection of fishnet junction points is titled "Visual Pose Estimation for Autonomous Inspection of Fish Pens." [9]. This paper proposes a novel X-junction detector

called Fast-Cross, which is capable of detecting fishnet knots and their topology in camera images. The detected knots are then utilized to estimate the pose of a camera relative to the fishnets. This approach avoids the need for tracking image features and is robust against repeated scene structures.

The experiments conducted in the paper demonstrate that detecting the fishnet in still images provides an effective means to estimate the distance and orientation of a camera with respect to a fishnet. By directly estimating the scale from the fishnet, this method overcomes the limitations of mono camera systems that typically reconstruct scenes only up to scale.

However, the proposed solution has some limitations. It is unable to estimate camera movement parallel to the fishnet, and it relies on clear visibility of fishnet knots. In situations where the fishnet is located in front of another fishnet or heavily perspective distorted, the performance may be affected. To improve the fishnet detector, the paper suggests guaranteeing an overlapping fishnet patch between subsequent images. This enhancement would enable the estimation of camera movements parallel to a fishnet if the displacements of detected fishnet knots are small enough between subsequent images to avoid ambiguities.

In contrast to the method proposed in the paper, the junction point detection algorithm in this thesis aims to detect junction points in fishnets by analyzing contour properties of the detected fishnet regions.

2.3 Underwater localization

Underwater localization is a challenging problem, prompting the exploration of techniques such as SLAM and visual navigation. Liu et al. [10] conducted a comprehensive study on vision-based techniques for the autonomous navigation and positioning of UUVs. They summarized and reviewed the navigation and positioning of UUVs based on vision and the fusion of vision with other sensors. The paper included a comparison between geometry-based methods and deep learning-based methods, evaluating their performance using a typical underwater dataset. The study also identified upcoming research trends and emphasized the availability and potential of visual navigation and positioning technology for UUVs.

Another notable contribution in the field of AUV navigation and localization is the review article titled "AUV Navigation and Localization: A Review" [11]. The paper discussed the challenges of underwater navigation and localization, including the attenuation of GPS and radio-frequency signals, low bandwidth underwater communications, and the lack of access to a global positioning system. It examined past approaches that used expensive inertial sensors, installed beacons, or required periodic surfacing of AUVs, highlighting their limitations. The review also explored recent advancements in underwater communications and the application of simultaneous localization and mapping (SLAM) technology to the underwater domain. Various state-of-the-art methods for AUV navigation and localization were described, considering different technical approaches, sensor utilization, and collaboration levels. The paper concluded by identifying areas of future research potential in the field.

The advancements in acoustic communications and SLAM have opened up new possibilities for underwater localization algorithms. The use of acoustic modems enables underwater collaboration, while sonar and optical sensors, combined with SLAM techniques, provide bounded localization. Although some methods are still in the early stages of formalization and testing, it is evident that these new techniques can overcome the limitations of traditional systems such as long-baseline (LBL) and ultra-short baseline (USBL).

The challenging and unpredictable nature of the underwater environment poses significant obstacles for autonomous underwater systems. However, recent advancements offer promising solutions to address these challenges and improve the navigation and localization capabilities of AUVs. By considering the findings from previous research on fishnet detection, junction point detection, and underwater localization, future studies can explore the potential integration or comparison of these techniques to further enhance fishnet detection and tracking in underwater environments.

Chapter 3

Theory

The theory chapter of this thesis provides a comprehensive understanding of computer vision and image processing theories, including camera projection, 3D reconstruction, homogeneous coordinates, and the challenges of retrieving 3D information from 2D images. It also covers 2D DFT for frequency analysis and explores optical flow, which estimates motion and can reveals the 3D structure of objects in a scene. This chapter establishes a solid theoretical foundation for the practical applications and methodologies discussed in the following chapters.

3.1 Camera Projection and 3D Reconstruction

This section is based of the computer vision book from Szeliski [12] and provides a comprehensive understanding of camera projection, 3D reconstruction, and homogeneous coordinates. It establishes the foundational concepts necessary for practical applications discussed in subsequent chapters.

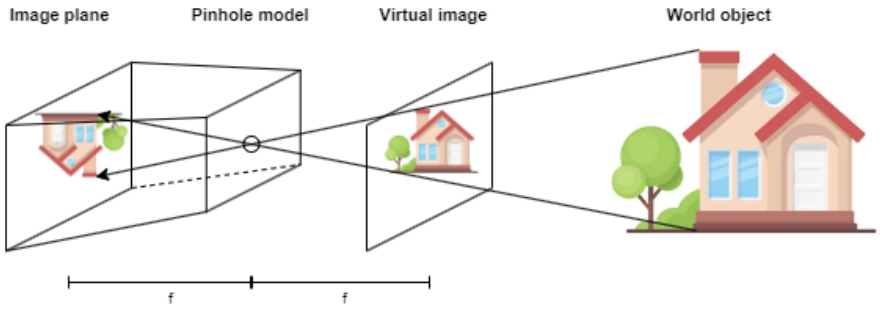
3.1.1 The perspective camera model

The perspective camera model, also known as the *pinhole camera model*, is a mathematical model used to describe how a camera maps 3D points in the world onto 2D points in an image. It is assumed that the camera has pinhole geometry, meaning that it has a small hole through which light passes and forms an image on the opposite side. An ideal pinhole camera is a box with an infinitely small opening, called the optical center, where every point on the imaging surface receives light from a unique direction, all of which intersect at the pinhole. The resulting mapping from the scene to the image is a *perspective projection*. Modern digital cameras work similarly, but the opening is larger, and a lens is placed in front to focus the light. Despite the complex optics of a modern camera, the perspective projection model can still be applied, and camera and lens engineers strive to produce a perspective projection that results in the most natural and pleasing images to the human eye.

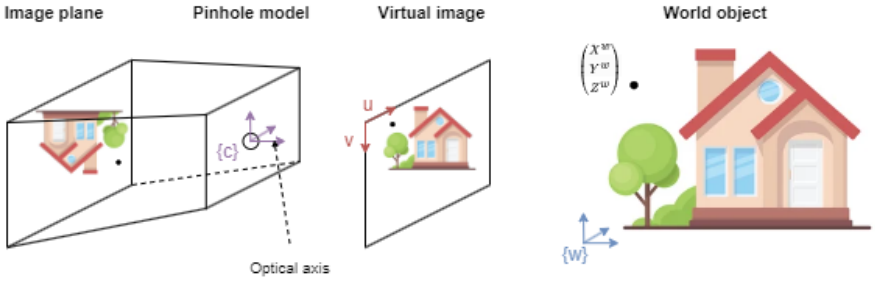
The pinhole model, as shown in Figure 3.1, uses the virtual image plane instead of the physical image plane because it simplifies the mathematics involved in calculating the projection of the 3D scene onto a 2D image. The parameters used in the ideal pinhole camera model are the principal point, pixel density, and focal length. The pinhole model has a focal length of f, a world frame w, a camera frame located at the optical center c, and pixel coordinates in the image (u, v).

Homogeneous coordinates

Homogeneous coordinates are used in computer vision to represent points in a projective space, which is an extension of the Euclidean space where vectors that differs only by a scale is considered to be equivalent. This allows for the representation of both points at finite and infinite distances.



(a) Pinhole model



(b) Pinhole model with axis

Figure 3.1: Pinhole model

Homogeneous coordinates are often used to represent points in camera transformations, where a 3D point in the world is projected onto a 2D image plane. A 2D point can be represented in homogeneous coordinates $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathcal{P}$ and can be transformed back to in-homogeneous coordinates by dividing by the last element:

$$\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) \tag{3.1}$$

The homogeneous coordinates of a point in 3D space are represented as a 4-dimensional vector, denoted as $\tilde{\mathbf{X}} = (\tilde{X}, \tilde{Y}, \tilde{Z}, \tilde{W})$, where $\tilde{W}$ is a scaling factor that allows for the representation of points at infinity.

3.1.2 3D world coordinates to 2D pixel coordinates

To obtain the 2D pixel coordinates of a 3D point in an image, several transformations need to be applied, as illustrated in Figure 3.2. First, the 3D point must be transformed from the world coordinate system to the camera coordinate system. This transformation involves a rigid-body transformation that consists of rotation and translation matrices, which can be combined into a 3×4 or 4×4 matrix known as the camera extrinsic matrix. Once the 3D point is in the camera coordinate system, it needs to be projected onto the 2D image plane. This is achieved by applying a projection matrix, which maps points from the camera coordinate system to the image plane. The projection matrix takes into account the focal length of the camera, and is typically represented by a 3x4 matrix. To obtain the pixel coordinates of the point, an affine transformation is applied, which maps the 2D image coordinates to pixel coordinates. The affine transformation is typically represented by a 3x3 matrix that includes pixel spacing and skew factor parameters.

The extrinsic camera matrix is represented by a 3x4 matrix, written as $[\mathbf{R}|\mathbf{t}]$, where $\mathbf{R}$ is a 3×3 rotation matrix that describes the orientation of the camera and $\mathbf{t}$ is a 3×1 translation vector that describes the position of the camera in the world coordinate system. Often the extrinsic matrix also have an additional row of $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ making it a square 4×4 matrix:

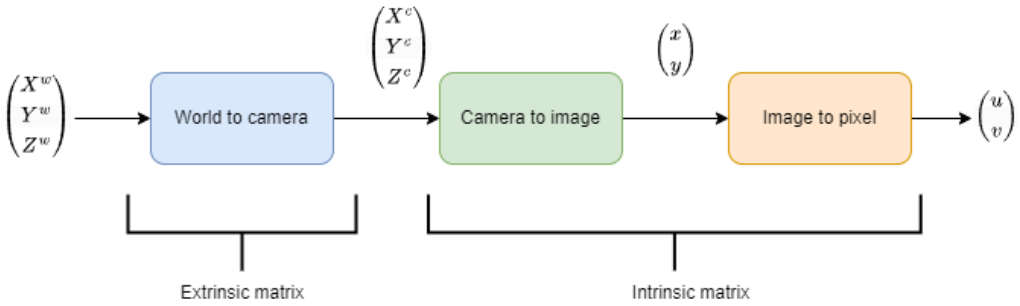


Figure 3.2: World coordinates to pixel coordinates

$$[\mathbf{R}|\mathbf{t}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.2}$$

The general equation for transforming a point in the world frame to the camera frame using the extrinsic matrix is:

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} \tag{3.3}$$

The 3×3 intrinsic camera matrix is denoted by $\mathbf{K}$ and can be written as:

$$\mathbf{K} = \underbrace{\begin{bmatrix} s_x & s_\theta & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Affine coordinate transform}} \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Scaling}} = \begin{bmatrix} fs_x & fs_\theta & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

where s_x and s_y are the horizontal and vertical *pixel density* respectively, given in units of pixels per unit length. In a perfect pinhole model, both of the focal lengths $f_x = fs_x$ and $f_y = fs_y$ have the same value, but flaws in the digital sensor, non-uniform scaling in post-processing, camera lens distortion or errors in camera calibration can make them different [13]. The skew factor s_θ is only non-zero in the unusual case where the pixels are not rectangular, meaning that s is usually 0.

The transformation from 3D camera coordinates $\begin{bmatrix} X^c & Y^c & Z^c \end{bmatrix}^T$ to 2D image coordinates $\begin{bmatrix} x & y \end{bmatrix}^T$ is done using homogeneous coordinates and the transform:

$$\tilde{w} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} \tag{3.5}$$

Combining the intrinsic and extrinsic matrices, the *projection matrix* $\mathbf{P}$ can be calculated and used to transform a point in the world space (X,Y,Z) to its corresponding 2D point in the image plane (u,v) and is given by:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} \tag{3.6}$$

for an arbitrary scale λ.

3.1.3 2D pixel coordinates to 3D world coordinates

Obtaining the precise 3D position of an object from a single 2D image is widely acknowledged as a challenging task in computer vision and 3D reconstruction. This is primarily due to the fact that the 2D projection of a 3D point on an image plane results in the loss of depth information or the Z-coordinate of the point. The loss of depth information in 2D projection occurs due to the fact that a 3D point can be projected onto an infinite number of 2D points. These infinite 2D points will lie on a ray passing through the 3D point and the optical center of the camera. Consequently, relying solely on a single 2D point to determine the 3D position of an object is not possible. To accurately calculate the 3D position, either a second image is required, which can be used for triangulation, or prior knowledge of the Z-value of the 3D point must be available.

The projection matrix $\mathbf{P}$ from Equation 3.6 is of size 3×4, making it non-invertible. If the 3D points lay on a plane in the world frame with $Z^w = 0$ then the extrinsic transformation can be simplified to:

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} X^w \\ Y^w \\ 1 \end{bmatrix} \tag{3.7}$$

The transformation of a point in $\{w\}$ to pixel coordinates is now:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{KH} \begin{bmatrix} X^w \\ Y^w \\ 1 \end{bmatrix} \tag{3.8}$$

Since the product of $\mathbf{KH}$ is a square 3×3 matrix, its inverse exist when its non-singular and 3D points with $Z^w = 0$ can then be found by:

$$\lambda \begin{bmatrix} X^w \\ Y^w \\ 1 \end{bmatrix} = \mathbf{H}^{-1}\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.9}$$

For any scaling factor λ.

Inverse extrinsic matrix

The transform back from camera coordinate system to world frame can be found by calculating the inverse extrinsic matrix. Let $\mathbf{X}^c = \begin{bmatrix} X^c & Y^c & Z^c \end{bmatrix}^T$ and $\mathbf{X}^w = \begin{bmatrix} X^w & Y^w & Z^w \end{bmatrix}^T$. The inverse of the extrinsic matrix can be found by:

$$\begin{bmatrix} \mathbf{X}^c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}^w \\ 1 \end{bmatrix} \tag{3.10}$$

$$\mathbf{X}^c = \mathbf{R}\mathbf{X}^\mathbf{w} + \mathbf{t} \implies \mathbf{X}^w = \mathbf{R}^{-1}(\mathbf{X}^c - \mathbf{t}) = \mathbf{R}^T\mathbf{X}^c - \mathbf{R}^T\mathbf{t} \tag{3.11}$$

Writing Equation 3.11 as a matrix with homogeneous coordinates gives the inverse extrinsic matrix as:

$$\begin{bmatrix} \mathbf{X}^w \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}^c \\ 1 \end{bmatrix} \tag{3.12}$$

3.2 Discrete Fourier transform

The DFT is an algorithm used for transforming a signal from the time domain to the frequency domain, and it is widely used in fields such as signal processing, image processing, and data analysis. In this section, the basics of the DFT and its two-dimensional version, the 2D DFT, will be covered. Important concepts such as fftshift, positive and negative frequencies, and plane waves will also be discussed, which are necessary for understanding how the DFT works. The fftshift operation is used to shift the zero-frequency component to the center of the transformed signal for easier analysis. Positive and negative frequencies correspond to the positive and negative parts of the original signal, respectively. Plane waves, also known as 2-parameter sinusoidal waves, are the building blocks of signals and can be represented using the complex exponential function. Finally, the implementation of the DFT using the NumPy library in Python will be demonstrated, which provides a convenient and efficient way to analyze images and other types of data.

3.2.1 2D DFT

The transformation of an image from the time domain to the frequency domain is achieved through the utilization of the 2D DFT equation. This fundamental concept is extensively explained in Chapter 4 of the book entitled "Multidimensional Signal, Image, and Video Processing and Coding" authored by John W. Woods [14]. The 2D Discrete Fourier Transform (DFT) calculates the frequency values, represented as F(k,l), by summing the product of each image pixel value, f(m,n), with a complex exponential term. This term accounts for the phase shift associated with each frequency component and is determined by the coefficient's position in the frequency domain. It is important to note that the sum in this equation is not normalized. The equation for the 2D DFT is given by:

$$F(k,l) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(m,n) e^{-j2\pi(\frac{km}{M} + \frac{ln}{N})} \tag{3.13}$$

Here, M and N represent the dimensions of the image. To revert the image from the frequency domain to the time domain, the inverse 2D DFT is employed. This transformation is given by Equation 3.14. Each pixel in the original image is computed by summing the product of each frequency coefficient, F(k,l), and a complex exponential term that varies based on the pixel's position. The complex exponential term reflects the phase shift associated with each frequency component and is determined by the coefficient's position in the frequency domain. The sum is normalized by dividing it by the product of the number of rows and columns in the image, denoted as M and N, respectively.

$$f(m,n) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F(k,l) e^{j2\pi(\frac{km}{M} + \frac{ln}{N})} \tag{3.14}$$

FFT shift

The use of fftshift in NumPy is valuable in data analysis and signal processing as it shifts the zero-frequency component of the DFT to the center of the transformed signal as illustrated in Figure 3.3. This allows for easier analysis and visualization of the frequency content, particularly in signals with symmetry properties. Additionally, filtering out specific frequencies becomes more straightforward when the zero-frequency component is located at the center.

Applying DFT and fftshift to an actual image gives Figure 3.4.

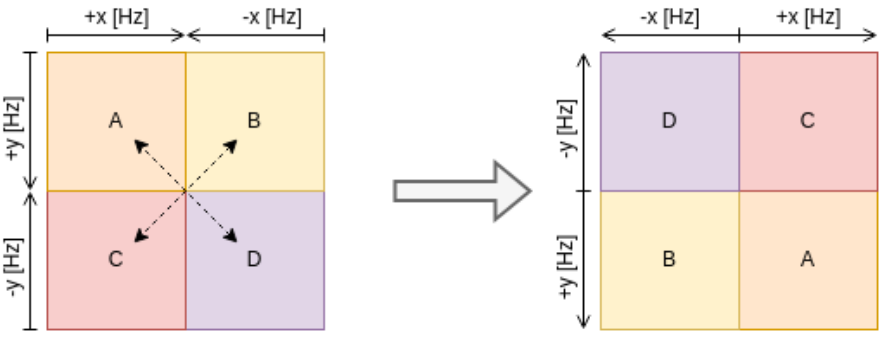
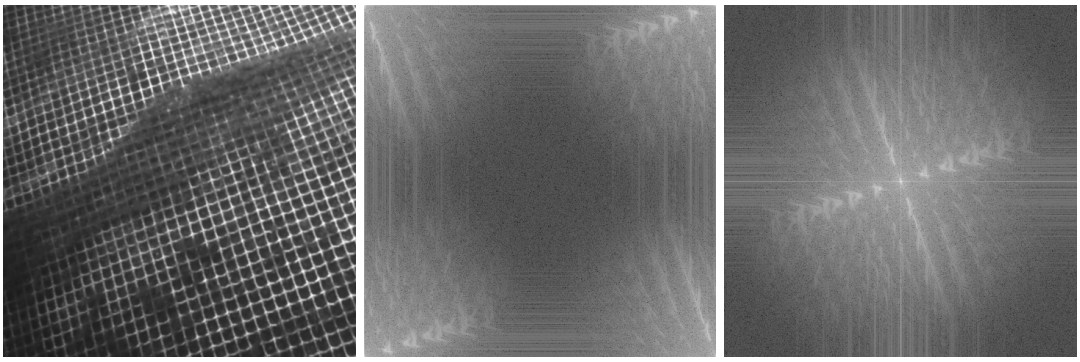


Figure 3.3: NumPy fftshift



(a) Original image (b) FFT of image (c) fftshift of FFT

Figure 3.4: The effects of FFT and FFT shift on an image

3.2.2 Sinusoidal grating

The reconstruction of an image can be achieved by combining a series of sinusoidal gratings. The sinusoidal function that is utilized for this purpose can be expressed in the general form:

$$y = A\sin(\frac{2\pi}{\lambda}x + \phi) \tag{3.15}$$

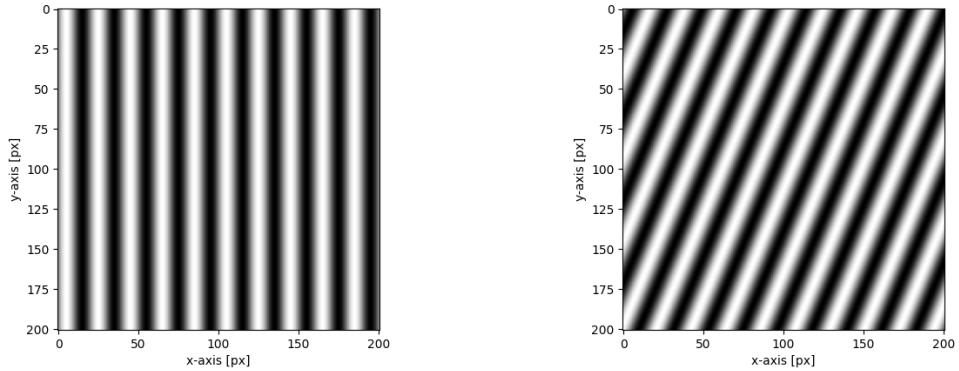
Here, A represents the amplitude, λ denotes the wavelength (which corresponds to the distance between two consecutive peaks), and ϕ signifies the phase shift. By utilizing the meshgrid method on x from NumPy, a plot of the sinusoidal grating can be generated, as shown in Figure 3.5a. To alter the orientation of the grating, an axis-transformation is necessary. For a given angle α, the grating can be calculated using:

$$grating = \sin(\frac{2\pi}{\lambda}(x \cdot \cos(\alpha) + y \cdot \sin(\alpha))) \tag{3.16}$$

Here, x and y represent the spatial coordinates of the image. Choosing a value of $\alpha = \pi/8$ produces the rotated sinusoidal grating shown in Figure 3.5b [15].

The corresponding DFT with fftshift are shown in Figure 3.6. For a single frequency component (i.e., a single sinusoid), the 2D DFT will have non-zero values at two points, located symmetrically about the origin in the Fourier domain. These points correspond to the positive and negative frequency components of the sinusoid.

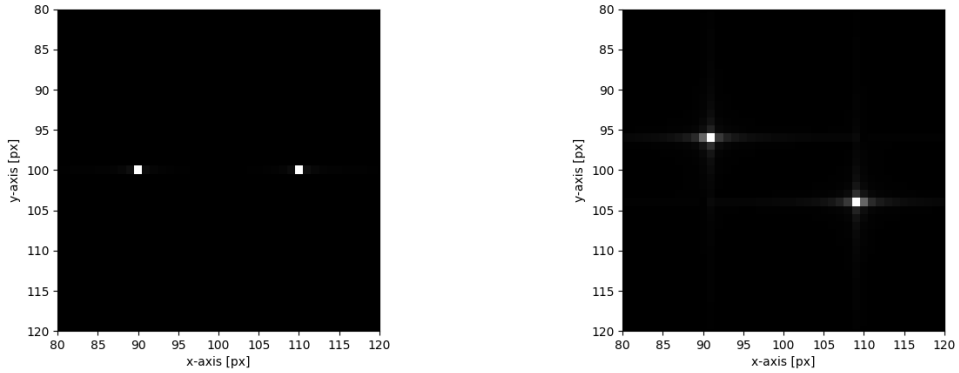
The 2D Inverse Discrete Fourier Transform (IDFT) represented by Equation 3.14 can reconstruct an image by adding a series of sinusoidal gratings. These gratings can be found using Euler's identity, as shown in Equation 3.17.



(a) Grating in x-direction

(b) Grating rotated $\pi/8$ rad

Figure 3.5: Sinusoidal grating



(a) FFT of grating in x-direction

(b) FFT of rotated grating

Figure 3.6: FFT of grating

$$e^{j2\pi(\frac{km}{M}+\frac{ln}{N})} = cos(2\pi(\frac{km}{M}+\frac{ln}{N})) + jsin(2\pi(\frac{km}{M}+\frac{ln}{N})) \tag{3.17}$$

In the 2D IDFT, complex values can cancel out due to the conjugate symmetry property of the Fourier Transform. This property ensures that complex values at frequencies (u,v) and (-u,-v) cancel out during the iDFT, resulting in a real-valued output at the corresponding spatial location (x,y) in the original image. The equation to recreate the grating for a given pixel (k,l) in the DFT image is simplified to:

$$grating(m,n) = \cos(2\pi(\frac{k}{M}m + \frac{l}{N}n)) \tag{3.18}$$

Comparing this to the general equation for a grating in 3.16, both the angle α and the wavelength λ can be found. the reciprocal wavelength, also known as the wave number ($\bar{v}$) is calculated by:

$$\bar{v} = 1/\lambda = \sqrt{\left(\frac{k}{M}\right)^2 + \left(\frac{l}{N}\right)^2} \implies \lambda = \frac{1}{\sqrt{\left(\frac{k}{M}\right)^2 + \left(\frac{l}{N}\right)^2}} = \frac{NM}{\sqrt{(Nk)^2 + (Ml)^2}} \tag{3.19}$$

For a square image where $N = M$, this simplifies to:

$$\lambda = \frac{N^2}{\sqrt{N^2(k^2+l^2)}} = \frac{N}{\sqrt{k^2+l^2}} \tag{3.20}$$

where $\sqrt{k^2+l^2}$ is the distance from the center of the shifted FFT to the point (k,l). The grating function from Equation 3.18 can be rewritten as:

$$grating(m,n) = \cos(\frac{2\pi}{\lambda}(\frac{k\lambda}{M}m + \frac{l\lambda}{N}n)) \tag{3.21}$$

The angle α can then be expressed as:

$$\cos\alpha = \frac{k\lambda}{M}, \quad \sin\alpha = \frac{l\lambda}{N} \tag{3.22}$$

$$\tan\alpha = \frac{\sin\alpha}{\cos\alpha} = \frac{lM}{kN} \implies \alpha = \arctan\frac{lM}{kN} \tag{3.23}$$

For a square image where $N = M$, this simplifies to:

$$\alpha = \arctan\frac{l}{k} \tag{3.24}$$

where the point (k,l) in the shifted DFT image is defined relative to the middle of the image.

3.3 Optical Flow

Optical flow is a fundamental concept in computer vision that provides valuable information about the 3D structure and estimates the motion of objects in a scene. It refers to the apparent motion of objects in an image sequence, caused by their actual motion and the motion of the camera. The notion of optical flow pertains to the movement of intensity patterns, and it is derived from the physiological understanding of how the world is perceived visually through the formation of images on the retina [16]. Optical flow methods are used for motion and object tracking and can be applied in various fields such as robotics, surveillance, and video analysis. It is also a valuable tool for video compression by estimating the motion of pixels between frames and only transmitting the difference, reducing bandwidth requirements. Optical flow is used in a wide range of applicative domains, including action recognition, video indexing and retrieval, video restoration, medical image registration, robot or vehicle navigation, automated video surveillance, fluid flow analysis, and dynamic textures [16], [17].

Differential and non-differential methods are the two main categories of optical flow methods. Differential optical flow methods estimate motion vectors for each pixel in an image sequence by computing the spatial and temporal derivatives of the image. These methods rely on the brightness constancy constraint, which assumes that the intensity of a pixel in an image sequence remains constant over time. The Horn-Schunck algorithm [18] and the Lucas-Kanade algorithm [19] are examples of differential optical flow methods.

Dense optical flow methods compute motion vectors for every pixel in an image sequence, providing accurate results but at a high computational cost. These methods use Partial Differential Equations (PDE) to solve for the optical flow of each pixel in the image. Dense optical flow methods are suitable for applications where high accuracy is required, such as object tracking and motion analysis.

Sparse optical flow methods estimate motion vectors for only a subset of pixels in an image sequence, making them computationally efficient but less accurate than dense optical flow methods. These methods use local optimization techniques, such as iterative Lucas-Kanade or pyramidal

Lucas-Kanade, to estimate the optical flow of a sparse set of pixels. Sparse optical flow methods are suitable for applications where real-time performance is required, such as video stabilization and autonomous vehicles.

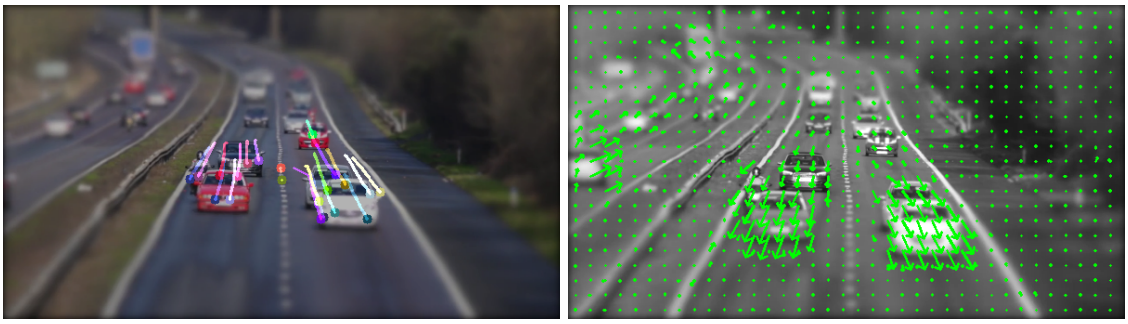
Non-differential methods estimate optical flow by matching features or blocks between consecutive images. Block matching divides images into blocks and estimates the motion by finding the best match between the blocks. Feature matching identifies and tracks specific features in the images, such as corners or edges.

Deep learning methods for optical flow estimation have recently gained popularity due to their ability to learn complex motion patterns directly from data. These methods use convolutional neural networks (CNNs) to estimate the optical flow. The CNNs are trained on large datasets of optical flow to learn the relationship between the input image sequence and the corresponding optical flow. Deep learning methods for optical flow estimation have shown to produce state-of-the-art results and can be used for various applications, such as human action recognition and scene understanding.

The differences between sparse and dense optical flow methods can be illustrated by using the example of a highway with cars. The motion of the cars using optical flow methods can be tracked. An example of a sparse optical flow method is the Lucas-Kanade algorithm, which estimates the optical flow of a sparse set of pixels called feature points. The Lucas-Kanade algorithm assumes that the motion of the feature points between consecutive frames is small and can be approximated by a linear function. However, this method can fail when there is a large motion between frames. To address this problem, image pyramids are used.

An image pyramid is a multi-scale representation of an image, where each level in the pyramid is a smaller version of the image with reduced resolution. The Lucas-Kanade algorithm starts by computing the optical flow at the top level of the pyramid and propagating the resultant flow to the immediate inferior level of the pyramid until the original image is reached. This allows the detection of large movements of pixels in the sequence of images.

The pyramidal Lucas-Kanade (PYR LK) is an iterative version of Lucas-Kanade that uses image pyramids to improve the accuracy of the estimation. Feature points on the cars, such as the corners, can be selected for tracking the motion of these points between consecutive frames using PYR LK. The optical flow resulting from PYR LK consists of motion vectors for each tracked feature point, indicating the direction and magnitude of the motion. he multi-scale approach enables the algorithm to estimate optical flow accurately even in the presence of large motions [20]. One way to visualize it over multiple frames is shown in Figure 3.7a. The tracked points, represented by the colored circles are found using the OpenCV method goodFeaturesToTrack, which is based of the Shi-Tomasi method [21].



(a) Sparse optical flow

(b) Dense optical flow

On the other hand, the Farneback algorithm [22] is an example of a dense optical flow method that computes motion vectors for every pixel in the image. The motion of image pixels is modeled using polynomial expansion, and the optical flow is solved for using PDE. A vector field is produced as the output of Farneback, where each pixel in the image has an associated motion vector. This vector field can be represented using a grid of vectors with a given spacing, where the vector length and direction represent the magnitude and direction of the motion, respectively as illustrated in Figure 3.7b. Alternatively, the vector field can be represented using HSV colors (Figure 3.9), where

the hue, saturation, and value represent the direction, magnitude of the motion. Using HSV colors to represent a vector field has several benefits. The hue represents the direction of motion, allowing for easy identification of the flow direction. The saturation represents the magnitude of the motion, with higher saturation indicating higher motion magnitude. This makes it easier to distinguish between regions of high and low motion, and can be useful in applications such as motion detection and tracking. Additionally, the use of HSV colors allows for a more compact representation of the vector field compared to a grid of vectors, making it possible to represent flow on every pixel.

Figure 3.8: Dense optical flow with HSV colors

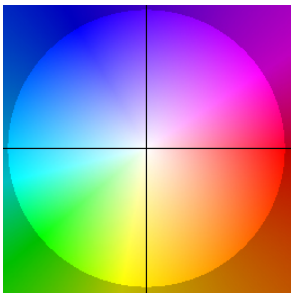


Figure 3.9: HSV colors

Adding the optical flow HSV colors on top of the original highway image gives Figure 3.10.

Figure 3.10: Original image with dense optical flow

To properly analyze the motion and stability of an AUV filming and stabilizing against a fishnet underwater, optical flow methods are essential. However, due to the grayscale images and varying illumination, traditional differential optical flow methods may not be the best option. These methods rely on brightness constraints and may struggle with changing illumination. Additionally, the repeating pattern of the fishnet can make it difficult for sparse optical flow methods to match features accurately. Non-differential methods, such as block matching, are another sparse optical flow approach that can be used in this application. Block matching divides images into small blocks and estimates motion by finding the best match between corresponding blocks in consecutive images. This method does not rely on the brightness constancy constraint and is therefore less sensitive to illumination changes. However, block matching may struggle with accurately matching blocks in the presence of repeating patterns, such as the fishnet.

In summary, the choice of optical flow method depends on the specific application requirements and environmental conditions. While differential optical flow methods are suitable for some applications, they may struggle with varying illumination and large motions. Dense optical flow methods provide accurate results but come with a high computational cost. Sparse optical flow methods, such as iterative Lucas-Kanade or block matching, are computationally efficient but may sacrifice accuracy. Deep learning methods for optical flow estimation have shown promising results in handling complex motion patterns and challenging environments. By selecting the appropriate

optical flow method, the motion and stability of an AUV filming and stabilizing against a fishnet can be properly analyzed.

Chapter 4

Method

This chapter presents a robust methodology for detecting and tracking fishnets using an AUV, with the goal of facilitating effective navigation. The chapter begins by introducing the pipeline for fishnet detection and AUV tracking, followed by the creation of an artificial fishnet for testing purposes. The proposed FFT algorithm is then employed to detect and identify fishnets, utilizing the frequency spectrum of grayscale images. Subsequently, the chapter addresses the estimation of fishnet pose and the detection of junction points. To ensure accurate tracking of the junction points, optical flow techniques, specifically the GMA approach, are applied.

Overall, this comprehensive methodology significantly enhances underwater operations by automating tasks and improving the efficiency and accuracy of fishnet detection and AUV navigation.

4.1 Pipeline

In this section, the pipeline for fishnet detection and tracking using an AUV is presented. The algorithm proposed aims to robustly detect fishnets in underwater scenes and track their junction points, enabling effective navigation of the AUV in their vicinity. The different stages of the pipeline, which contribute to the overall task of fishnet detection and AUV tracking, are illustrated in Figure 4.1.

1. Input image

The pipeline begins with the acquisition of an input image, which represents the underwater scene captured by the AUV's onboard imaging system. The image serves as the primary data source for subsequent processing steps. Underwater scenes pose numerous challenges, such as low visibility, ambient noise, and distortions. Therefore, the subsequent stages of the pipeline are designed to mitigate these challenges and enhance the detection and tracking capabilities.

2. Image pre-processing

To prepare the input image for fishnet detection, a series of image pre-processing techniques are applied. This stage involves conversion to grayscale, transforming the image into the frequency domain, and noise removal.

3. Fishnet detection with FFT algorithm

The core of the fishnet detection stage relies on a FFT algorithm. The FFT algorithm is utilized to analyze the frequency content of the pre-processed image and identify distinctive patterns associated with fishnets. By leveraging the periodic nature of fishnet structures, the characteristic frequencies of the fishnet are extracted, enabling the estimation of not only the fishnet's presence but also its orientation and distance from the AUV. The algorithm searches for spectral peaks corresponding to the fishnet's characteristic frequencies. The detected peaks are subsequently

thresholded to identify potential fishnet regions in the image.

4. Fishnet verification

Once potential fishnet regions are identified, a verification step is performed to validate their presence and discard false positives. If the fishnet detection algorithm determines that no fishnet is present in the current image, the pipeline loops back to the initial stage, ensuring continuous monitoring of the underwater scene. On the other hand, if a fishnet is detected, the pipeline proceeds to estimate the pose of the fishnet relative to the AUV.

5. Junction point detection and pose estimation of fishnet

To determine the pose of the fishnet relative to the AUV, the pipeline employs a contour detection algorithm. This stage aims to identify the junction points of the fishnet, which serve as key reference points for subsequent tracking. By analyzing the contour properties of the fishnet regions, such as shape, connectivity, and curvature, the algorithm accurately localizes the junction points. Furthermore, the contour detection algorithm determines the x and y directions of the fishnet, contributing to the complete estimation of the fishnet's pose relative to the AUV's coordinate system. The estimated pose includes information about the fishnet's orientation and position, providing vital spatial awareness for the AUV's navigation and interaction with the fishnet.

6. Tracking of junction points with optical flow

Once the pose of the fishnet is estimated, the pipeline enters the tracking stage, which utilizes optical flow algorithms to track the junction points over time. By applying optical flow to the junction points, the AUV can continuously track their movements and adjust its navigation to maintain proximity to the fishnet.

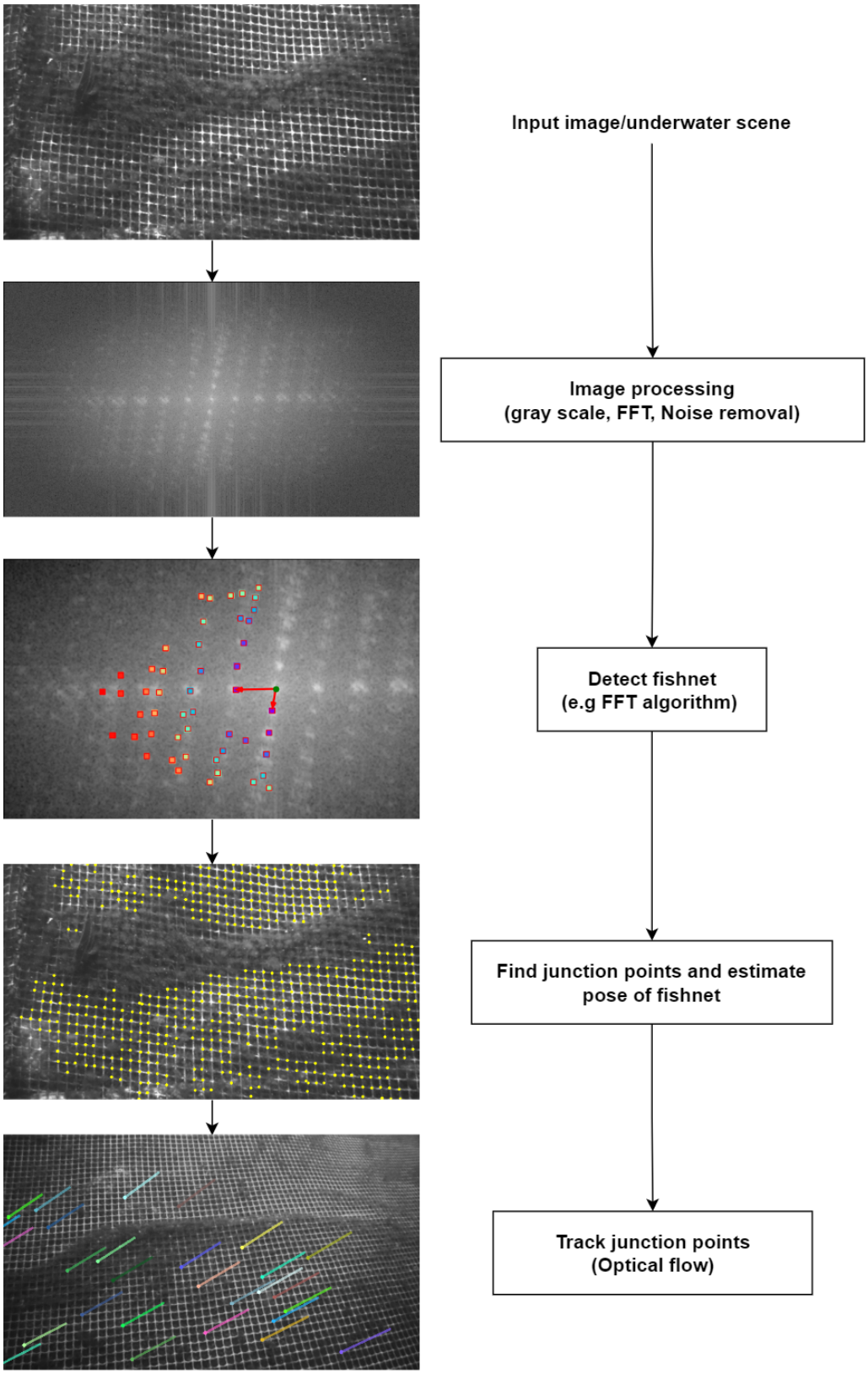


Figure 4.1: Pipeline flowchart

4.2 Creating fishnet

Accurately detecting the position and orientation of a fishnet is crucial for AUVs deployed in netpen environments. Fishnets serve as essential structures for containing fish populations, and effective fishnet detection plays a vital role in various tasks such as fish population monitoring, fish behavior analysis, and net integrity assessment, as discussed in chapter 1. By developing and testing a fishnet detection algorithm, it becomes possible to automate these tasks and improve the efficiency and accuracy of underwater operations.

To analyze, evaluate, and benchmark the performance of the fishnet detection algorithm, simulated fishnets with known poses relative to the AUV are generated. The algorithm's ability to accurately detect and estimate the fishnet's position, orientation, and grid spacing will be evaluated against ground truth data. Measures such as detection accuracy and localization error can be used as evaluation metrics. Additionally, the algorithm's computational efficiency and robustness in handling different environmental conditions, such as varying lighting and water turbidity, can be analyzed.

Algorithm 1 creating fishnet

```
function CreateGrid(n_rows, n_cols, cell_size, K, R, t, img_size)
    grid_points = array of size n_rows*n_cols × 3
    for row in range(n_rows) do
        for col in range(n_cols) do
            grid_points[row*n_cols + col][0] = (col - n_cols/2)*cell_size
            grid_points[row*n_cols + col][1] = (row - n_rows/2)*cell_size
        end for
    end for
    img_points = ProjectPoints(grid_points, R, t, K)
    img_points = img_points.reshape(n_rows, n_cols, 2)
    img = array of size img_size[0] × img_size[1] × 3 filled with a background color
    for row in range(n_rows) do
        for col in range(n_cols) do
            if row < n_rows - 1 then
                DRAW line from img_points[row, col] to img_points[row + 1, col] in img
            end if
            if col < n_cols - 1 then
                DRAW line from img_points[row, col] to img_points[row, col + 1] in img
            end if
        end for
    end for
    return img, grid_points, img_points
end function
```

In order to create simulated fishnets with a known pose, the presented algorithm, outlined in Algorithm 1, is employed. The algorithm generates an artificial planar grid of size $n_rows \times n_cols$ centered around the origin of the world frame. Each grid point is represented as a 3D coordinate with a zero value in the Z-axis and scaled by a given cell size. These 3D points are then projected onto the 2D image plane using the camera intrinsic matrix $\mathbf{K}$ of size 3×3 and the extrinsic matrix $[\mathbf{R}|\mathbf{t}]$ of size 3×4, where $\mathbf{R} = \mathbf{R}^{\mathbf{c}}_{\mathbf{w}}$ represents the rotation matrix from the world frame to the camera frame and $\mathbf{t}$ denotes the translation vector. The projection is performed using the OpenCV method project_points or alternatively with the camera projection equation from Section 3.1.2:

$$\tilde{w} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} \tag{4.1}$$

where (u, v) represents the 2D pixel coordinates and (X^w, Y^w, Z^w) correspond to the 3D coordinates in the world frame. A minimal representation of orientation can be obtained by using a set

of three angles, denoted as ϕ, θ, and ψ. These angles correspond to the rotational transformations around the coordinate axes and allow for the construction of a generic rotation matrix. To ensure non-parallel axes for successive rotations, a suitable sequence of three elementary rotations can be composed. This results in 12 distinct sets of Euler angles, each representing a unique triple combination [23].

In the (aero)nautical field, the Roll-Pitch-Yaw (RPY) angles, denoted as ZYX angles, are commonly analyzed as they describe the standard alterations of attitude for (air)craft. In the convention used by Fossen [24], the ZYX convention, the rotations are defined as follows: yaw around the Z-axis, pitch around the Y-axis, and roll around the X-axis. The resulting rotation matrix $\mathbf{R}$ can be defined as:

$$\begin{aligned}\mathbf{R} = \mathbf{R}_{\mathbf{w}}^{\mathbf{c}} &= \mathbf{R}_{\mathbf{z}}(\psi)\mathbf{R}_{\mathbf{y}}(\theta)\mathbf{R}_{\mathbf{x}}(\phi) \\ &= \underbrace{\begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{yaw}} \underbrace{\begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}}_{\text{pitch}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}}_{\text{roll}}\end{aligned} \tag{4.2}$$

Next, the algorithm draws lines from the grid points to its neighboring grid points in both vertical and horizontal directions. This approach, which creates a more realistic scenario by allowing for some deviation from straight lines, is more robust and gives more possibilities than drawing lines between only the edge points. The resulting image with the drawn grid lines, along with the 3D grid points in the world frame and the pixel values of the grid points (img_points), are returned by the algorithm. These values can be utilized to generate test data for fishnet detection algorithms, and as a ground truth to evaluate the performance of the algorithm.

The process of creating a fishnet using Algorithm 1 is illustrated by two examples below in Figure 4.2. In both the examples, a fishnet with 10 row and column lines and a cell size of 0.05 meters was created. The image size was chosen as 848×480 pixels and a camera intrinsic matrix $\mathbf{K}$ as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 600 & 0 & 848/2 \\ 0 & 600 & 480/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The first example in Figure 4.2a has no rotation, meaning that the $\mathbf{R}$ matrix is an 3×3 identity matrix and it only has translation in z-direction, moving it 1.5 meters away in the optical axis from the camera origin. Because c_x and c_y are half the image sizes, the grid will be centered in the image. The second example in Figure 4.2b includes an additional translation of 0.5 meters in the x-direction, followed by $\pi/6$ rad rotation in roll and $\pi/4$ rad rotation in yaw.

(a) Grid: $\mathbf{t} = (0, 0, 1.5)^T$ (b) Grid: $\mathbf{t} = (0.5, 0, 1.5)^T$, $\mathbf{R} = \mathbf{R}_{\mathbf{x}}(\pi/6)\mathbf{R}_{\mathbf{z}}(\pi/4)$

Figure 4.2: Simulated fishnet

These examples demonstrate how the algorithm can be used to create simulated fishnets with known poses, which can be used to test and evaluate the performance of fishnet detection algorithms for

AUVs operating in netpen environments.

4.3 Detecting & Recognizing fishnets

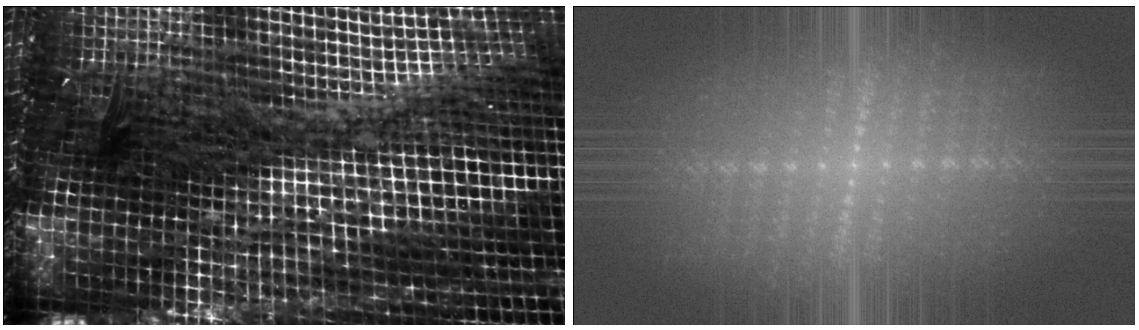
In this section, the detection and identification of a fishnet in underwater images are discussed using the Fast Fourier Transform (FFT) method, based on the work of Schellewald et al. [8]. The FFT method exploits the frequency spectrum of the grayscale image to identify repeated patterns, which appear as peaks in the 2D FFT. Additionally, the method provides an estimate of the orientation and distance of the fishnet relative to the camera.

4.3.1 Image pre-processing

Underwater images can be affected by various factors such as water turbidity and scattering of light, leading to noise and poor visibility. To mitigate these factors, grayscale images are commonly used in image processing and detection algorithms. Grayscale images simplify the analysis process, as they have higher contrast compared to color images, making it easier to detect objects and features in the image. Additionally, grayscale images have only one channel of information, whereas color images have three, making detection algorithms simpler and faster to process. However, it's important to note that grayscale images are still influenced by lighting conditions and can vary in brightness and contrast due to changes in lighting. Therefore, when comparing grayscale images taken at different times or under different lighting conditions, it's necessary to account for these variations in order to make accurate comparisons. Despite this limitation, grayscale images remain a useful tool for image processing and analysis in underwater environments.

To identify repeated patterns in the image, a DFT is performed on the grayscale image to obtain its frequency spectrum. By default, the output of the DFT in NumPy places the zero-frequency component (also known as the DC or average value of the image) at the top left corner of the frequency domain, while higher frequencies are located further away. This can make it challenging to visualize or analyze the frequency components, especially if the focus is on low-frequency components. However, by applying the fftshift function, the zero-frequency component is shifted to the center of the frequency domain, while the frequency values increase in all directions away from the center. This results in a more intuitive and convenient arrangement of the frequency components, making it easier to analyze the frequency content of the image.

The frequency values of each pixel in the Fourier domain are complex numbers, which can be challenging to visualize. To simplify this, the magnitude spectrum of the Fourier transform is used, which provides information about the amplitude of each frequency component. The values of the magnitude spectrum are then normalized to the range of 0 to 255 for better visualization, and is illustrated in Figure 4.3. To reduce noise and enhance the detection of peaks, a Gaussian blur filter is applied to the FFT image.



(a) Original image grayscale (b) FFT of image

Figure 4.3: Reprocessing of FFT

4.3.2 Peak detection

The detection of local maxima in a 2D image is a fundamental task in many computer vision and image processing applications, including object detection, feature extraction, and pattern recognition. A common method for local peak detection involves using a sequence of filters and thresholding techniques. The method is based on the principle that local maxima in an image are associated with pixels that have high values relative to their neighboring pixels. To detect these local maxima, the image is first filtered with a maximum filter using a specified neighborhood size. The maximum filter replaces each pixel in the image with the maximum value within its neighborhood, effectively smoothing the image and enhancing local maxima.

After filtering the image with the maximum filter, a binary mask is created where the value is 1 at positions where the pixel value in the image is equal to the corresponding maximum pixel value in the filtered image. This binary mask represents the positions of local maxima in the image. To refine the detection of local maxima, a minimum filter is applied to the image, which smooths out the noise and provides a baseline for the intensity values. The difference between the maximum and minimum filtered images is then calculated, and a binary mask of significant maxima is created by thresholding this difference. This binary mask represents the positions of local maxima that are significantly different from their surrounding pixels.

Algorithm 2 Detecting peaks

```
function FindPeaks(img, neighborhood_size, threshold, min_peak_dist)
    data_max = maximum_filter(img, neighborhood_size)
    maxima = (image == data_max)

    data_min = minimum_filter(img, neighborhood_size)
    diff = ((data_max - data_min) > threshold)
    maxima[diff == 0] = 0

    labels = label(maxima)
    peaks = center_of_mass(img, labels)
    if size(peaks) == 0 then
        return empty array
    end if

    dbscan = DBSCAN(eps=min_peak_dist, min_samples=1)
    clusters = dpscan.fit_predict(peaks)
    unique_labels = Unique(clusters)

    max_peaks = empty array
    for label in unique_labels do
        cluster_peaks = peaks[clusters == label]

        max_intensity_idx = argmax(img[cluster_peaks])
        max_peaks.append(cluster_peaks[max_intensity_idx]
    end for
    return max_peaks
end function
```

In some scenarios, when multiple pixels with the same peak value are in adjacent locations within an image, they all count as peaks, resulting in connected regions covering multiple pixels. In such cases, it becomes necessary to merge these connected regions into a single peak. The position of this merged peak is determined by calculating the center of mass within the connected region. This step ensures that each distinct peak is represented by a single coordinate, even when the peak spans multiple adjacent pixels.

To accomplish this task, the Python library SciPy's ndimage function *label* is used on the binary mask of significant maxima, which assigns a unique label to each connected region. This process

results in an array of the same shape as the input binary mask, where every pixel in a connected component is assigned the same integer label. The center_of_mass() function from the SciPy library is then employed to calculate the center of mass for each labeled object or connected component. The output of the function is a numpy array containing the center of mass coordinates (x, y) for each connected region, where x and y represent the column and row indices in the image array, respectively.

An additional step that can be done is to find clusters of peaks that are within a given number of pixels from each other and only keep the largest one. This would remove some of the bad peaks that come from noise and other objects in the image than the fishnet. To find the clusters, a clustering algorithm called Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [25] is used. The DBSCAN algorithm, which is implemented using the scikit-learn library, uses two main parameters: min_samples and eps. The min_samples parameter specifies the minimum number of potential junction points required to form a cluster. This should be set to 1 because it is desired to keep all single peaks that are found. The eps parameter specifies the maximum distance between two points for them to be considered part of the same cluster. This should be tuned to match to be somewhere in the range of $0 < eps <$ min_peak_dist, where min_peak_dist is the smallest expected distance in pixels between two peaks. Optimally eps should be tuned adaptively relative to the distance the AUV is from the fishnet.

The peak detection algorithm is presented in Algorithm 2, providing an efficient and accurate approach for detecting local maxima in a 2D image.

To reduce computation and optimize the analysis of peaks in the FFT image, it suffices to consider only half of the image. This reduction is possible due to the symmetry of the Fourier Transform of a real-valued signal, wherein the second half of the FFT image mirrors the first half. Consequently, the second half contains redundant information and can be safely discarded.

Additionally, within the frequency domain representation, the detection of peaks becomes progressively more challenging and blurred when moving away from the central pixel due to the higher frequencies towards the image edges. Consequently, only a cropped rectangle of the FFT image is needed for further peak analysis. Since the peaks outside an ellipse are going to be filtered out in the subsequent subsection 4.3.3, the size of the rectangle should be based upon the radii of the ellipse (r_h and r_v). The cropping of the image along the x- and y-directions can be described as follows:

$$c_x - r_h \leq x \leq c_x \tag{4.4}$$

$$c_y - r_v \leq y \leq c_y + r_v \tag{4.5}$$

where (c_x, c_y) represents the center of the image in pixels. Some additional pixels or padding could be added to make peaks at the edge area detectable. This rectangular cutout minimizes unnecessary processing in the algorithm and the proceeding steps.

Upon applying the peak detection algorithm (Algorithm 2) to the cropped version of Figure 4.3b, the resulting image is depicted in Figure 4.4. The color mapping employed for the peaks is purely for visualization purposes, with the peaks sorted by distance. The closest peaks exhibit a violet color, while the furthest peaks from the center appear red, following the colors of the rainbow.

4.3.3 Peak filtering

To ensure the accuracy and reliability of the analysis, it is necessary to filter out peaks with low relevancy. To achieve this, peaks that are closer to the edge of the image and tend to be less distinct and blurry are excluded. Only peaks that fall within a certain radius, denoted as r, are considered.

In the case of rectangular images, the radii r_h and r_v can be used to describe the selection of peaks within an ellipse, as demonstrated in Figure 4.5a.The location of each peak can be defined by its

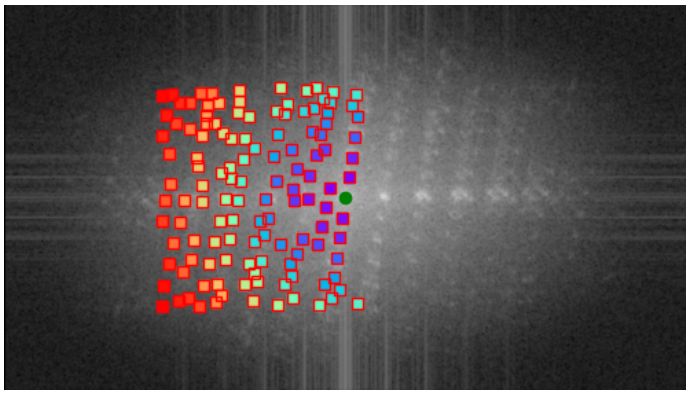


Figure 4.4: FFT with peaks

coordinates (x_p, y_p) with respect to the center of the image. In order to determine which peaks lie within the ellipse, the following condition is applied:

$$\frac{x_p^2}{r_h^2} + \frac{y_p^2}{r_v^2} < 1 \tag{4.6}$$

A lower threshold approach is implemented to address the issue of insignificant peak intensities in the FFT image. The threshold value is based on the intensity of the peaks in relation to the center of the FFT image, which contains the highest intensity peaks. The threshold value starts at the intensity value of the center peak and decreases linearly towards the edge of the ellipse. At the edge of the ellipse, the threshold value is set to the average intensity value of the FFT image. Peaks with intensities below the threshold value are considered insignificant and are excluded from further analysis. This approach enabled the focus on the most significant and informative peaks, while eliminating those that were less relevant for the analysis.

To determine the distance a peak has traversed on the path from the center to the edge of the ellipse, two essential quantities are required: the distance between the center and the peak and the distance between the center and the edge of the ellipse (radius) in the direction of the peak. Polar coordinates are the most appropriate method to obtain the ellipse radius in the direction of the peak as they include an angle θ and a radius r. Solving for r requires the equation of the ellipse in polar coordinates as the starting point:

$$\frac{x^2}{r_h^2} + \frac{y^2}{r_v^2} = 1 \implies \frac{r^2 cos^2(\theta)}{r_h^2} + \frac{r^2 sin^2(\theta)}{r_v^2} = 1 \tag{4.7}$$

$$r = \frac{1}{\sqrt{cos^2(\theta)/r_h^2 + sin^2(\theta)/r_v^2}} = \frac{r_h r_v}{\sqrt{r_v cos^2(\theta) + r_h sin^2(\theta)}} \tag{4.8}$$

where r_h and r_v are the horizontal and vertical radii respectively. The next step is to calculate the angle θ, which is just the angle of the peak:

$$\theta = \arctan y_p/x_p \implies \cos\theta = \frac{x_p}{\sqrt{x_p^2 + y_p^2}}, \quad \sin\theta = \frac{y_p}{\sqrt{x_p^2 + y_p^2}} \tag{4.9}$$

Then $\cos\theta$ and $\sin\theta$ can be substituted back in Equation 4.8:

$$r = \frac{r_h r_v}{\sqrt{r_v x_p^2/(x_p^2 + y_p^2) + r_h y_p^2/(x_p^2 + y_p^2)}} = \sqrt{x_p^2 + y_p^2} \cdot \frac{r_h r_v}{\sqrt{r_v x_p^2 + r_h y_p^2}} \tag{4.10}$$

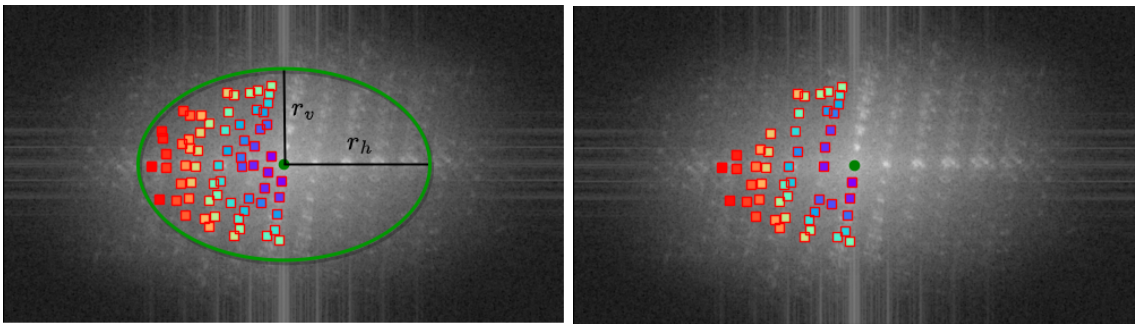
The traversed distance s is simply calculated by dividing the distance to the peak from the center by the radius r:

$$s = \frac{\sqrt{x_p^2 + y_p^2}}{r} = \frac{\sqrt{r_v x_p^2 + r_h y_p^2}}{r_h r_v} \tag{4.11}$$

The threshold value should be determined such that it ranges from the center peak value, with a margin subtracted, to the average value of the FFT image. Using the average value as a lower bound is acceptable, as it is unlikely that any significant peak would have a value lower than that. The threshold for a given peak (x_p, y_p) can be obtained as follows, with $\max_{\text{thres}}$ denoting the center value minus a margin and $\min_{\text{thres}}$ representing the average FFT image value:

$$\text{threshold} = (\max_{\text{thres}} - \min_{\text{thres}}) \cdot s + \max_{\text{thres}} \tag{4.12}$$

Applying the lower threshold to Figure 4.5a gives Figure 4.5b.



(a) Elliptical region for identified peaks

(b) Filtered low-intensity peaks

Figure 4.5: Filtering out peaks from FFT image

4.3.4 Identification of Significant Peaks

Several factors are taken into consideration when determining which peaks constitute the basis vectors of the grid. While the closest and brightest peaks may appear to be the appropriate candidates, this assumption is not always accurate due to noise interference and imperfect repetitive grid structures present in the fishnet images.

A more robust approach involves assigning a comprehensive value to each peak, taking into account three factors:

1. The peak value
2. Beam value in the direction of the peak
3. Repetitive pattern in the direction of the peak

Each factor should be weighted by coefficients k_1, k_2, k_3, respectively, based on their significance and potential impact on the determination of the basis vector peaks. The peak value itself, represented by the pixel value of the peak, typically holds the most importance. Consequently, the first factor should be appropriately scaled by k_1 to amplify its significance.

In the next factor, a beam is projected from the center towards the edge of the ellipse in the direction of a given peak. The length of the beam can be calculated by Equation 4.10. Currently, a beam width of 3 pixels is used, and the mean value of all the pixel intensities in the beam is calculated. This value is denoted as the beam value. This beam value is then subtracted from

the mean pixel value of the whole cropped image defined by Equation 4.4, and multiplied by the weight k_2. Given that the discrepancy between the mean pixel value and the beam value is typically relatively small compared to the peak values, it is necessary to use a relatively large scaling factor.

Lastly, the peaks of the basis vectors are expected to have the longest repetitive pattern of peaks within their respective directions. To identify this repetitive pattern, a vector $\vec{v}$ can be constructed as $\vec{v} = \text{peak} - (c_x, c_y)$, where (c_x, c_y) denotes the center of the image. The surrounding area around a point created by adding $2\vec{v}$ to the center should be searched for peaks within a given radius. In the presence of multiple peaks found, the one with the highest intensity value is selected. This process continues until no further peaks are found. To diminish the influence of each subsequent peak, a scaling factor, which decreases for each succeeding peak, is applied to the found peak values, in addition to weight factor k_3 and added to the total value. To further enhance the robustness of the algorithm, additional checks can be implemented, such as ensuring that the peak value falls within a certain range in comparison to the preceding peak. This comes from the expectation that the peaks generated by the grid pattern should exhibit relatively similar values. The combination of these factors enables the algorithm to identify the peaks constituting the fundamental vectors of the grid. The complete algorithm is presented in Algorithm 3.

Algorithm 3 Peak Values

```
function PeakValues(img, peaks, k1, k2, k3, scale_factor, radius)
    c = center of img
    avg = Mean(img)
    total_values = empty list

    for peak in largest peaks do
        total_value = 0
        total_value += (img[peak] - avg) × k1
        total_value += (Beam(img, peak) - avg) × k2

        vec = peak - c
        search_point = c + 2 × vec
        scale = 1

        for i in range(3, max searches do
            potential_peaks = peaks[Norm(peaks - search_point) ≤ radius]
            if potential_peaks is empty then
                break
            end if

            peak = largest of potential_peaks
            total_value += img[(peak - avg)] × k3 × scale
            scale = scale × scale_factor

            search_point = c + i × vec
        end for
        total_values.append(total_value)
    end for

    return total_values
end function
```

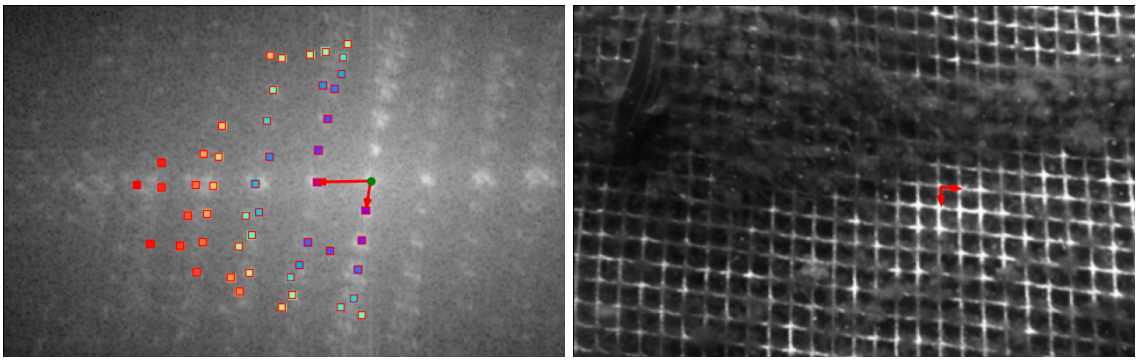
One way to improve the accuracy of the estimated basis vector $\vec{v}$ is by leveraging the fact that all peaks in the repeated pattern should ideally lie on the same line originating from the center of the image. Initially, the vector is determined as the difference between the first peak and the center, as described earlier. When a new peak is detected using the inner loop in Algorithm 3, a new vector can be formed from the center to the new peak, divided by two to account for its double length compared to the first vector. This new vector is denoted as $\vec{v}_2$. By taking the average of the original vector and the new vector, a combined estimated of the actual vector can be obtained.

Assigning additional weight to the first vector, the equation becomes:

$$\vec{v} = \frac{\vec{v}(i-1) + \vec{v}_2/(i-1)}{i} \tag{4.13}$$

The accuracy of the vector improves as more peaks of the repeated pattern are found. Applying Algorithm 3 to the remaining peaks after filtering in Figure 4.5 yields the basis vectors. These vectors, represented as red arrows, are drawn from the center pixel (marked by a green circle) to the two largest peaks. If the angle between the two largest peaks is too small, the next largest peak is checked for a suitable angle. The second basis vector is selected from the next largest peak that meets the angle requirement.

4.3.5 Time domain vectors



(a) FFT with peaks and basis vectors (b) Grayscale image with basis vectors

Figure 4.6: Analysing FFT

To obtain the time domain (spatial domain) basis vectors of the grid the angle and spacing of the vertical and horizontal grid lines need to be found. The grid can be seen as a sinusoidal grating, which was explained in Section 3.2.2. The length of each vector will be the wavelength λ from Equation 3.19. The basis vector will be perpendicular to the grid lines and therefore have an angle $\alpha \pm \pi/2$, where α is calculated by Equation 3.23. To rotate the vectors $\pi/2$ rad clockwise the transform $(x, y) \rightarrow (y, -x)$ can be used. The basis vectors v_1 and v_2 are therefore given by:

$$v_i = (\lambda_i \sin \alpha_i, -\lambda_i \cos \alpha_i), \quad i = 1, 2 \tag{4.14}$$

The rotated and scaled basis vector, which is now aligned with the fishnet grid, is now shown in Figure 4.6b. The next step is to create 2d to 3d correspondences to estimate the pose of the grid. Let's define the two basis vectors as $\mathbf{b_1}$ and $\mathbf{b_2}$ and these vectors can now be used to define four 2d pixel points in the image that describe a single cell:

$$x1 = -\mathbf{b_1}/2 - \mathbf{b_2}/2 + \mathbf{c} \tag{4.15}$$
$$x2 = -\mathbf{b_1}/2 + \mathbf{b_2}/2 + \mathbf{c} \tag{4.16}$$
$$x3 = \mathbf{b_1}/2 + \mathbf{b_2}/2 + \mathbf{c} \tag{4.17}$$
$$x4 = \mathbf{b_1}/2 - \mathbf{b_2}/2 + \mathbf{c} \tag{4.18}$$

Where $\mathbf{c} = (c_x, c_y)$ is the center of the image in pixels.

To get the corresponding 3D points, the grid size must be known and is assumed to be $b = grid\_size/2$. Assuming the fishnet is a plane, then in the frame of the fishnet the 3D points are

$$X1 = \begin{bmatrix} -b & -b & 0 \end{bmatrix} \tag{4.19}$$
$$X2 = \begin{bmatrix} -b & +b & 0 \end{bmatrix} \tag{4.20}$$
$$X3 = \begin{bmatrix} +b & +b & 0 \end{bmatrix} \tag{4.21}$$
$$X4 = \begin{bmatrix} +b & -b & 0 \end{bmatrix} \tag{4.22}$$
$$\tag{4.23}$$

By using OpenCV's solvePnP algorithm and the 2D to 3D correspondences with the camera matrix, a rotation vector and a translation vector are obtained. The rotation vector can be transformed into a rotation matrix by applying OpenCV's Rodrigues transform and only the z component of the translation matrix is valid. This is because if the grid was moved in the x or y direction, the FFT would still get the same frequency image. The x and y component is therefore rather found in the next subsection, and will be close to zero in this case.

Confidence score

To calculate how confident the choices for the peaks for the basis vectors are, a simple method can be utilized. By utilizing Algorithm 3, if the two largest peaks exhibit significantly higher values compared to the third largest value, it is highly likely that those peaks are the correct ones. The confidence score of the first basis vector is determined by dividing the largest peak value by the third largest peak value, while the confidence of the second basis vector is obtained by dividing the second largest peak value by the third largest peak value. The resulting confidence scores will fall within the range of $[1, \text{inf})$; however, empirical testing has shown that under normal circumstances, they typically range from $(1, 2.2)$.

4.3.6 FFT improvements

One limitation of the FFT method is that it does not differentiate between positive and negative angles in rotations around the x and y axes. In other words, it can calculate the rotation but cannot determine the direction of the rotation. However, the rotation around the z-axis can be obtained using the FFT method.

To address this limitation and enhance the performance of the FFT method, particularly for images with multiple frequencies, which often occurs when the fishnet is tilted or when the AUV is at a further distance from fishnet, the image of the fishnet can be divided into four (or more) sections. The FFT peak analysis can then be performed on each section individually, resulting in four individual distance estimates. From these estimates, the three most confident distances can be selected based on their confidence score, and a plane can be constructed using these three points. The pose of the plane, and thus the fishnet, can then be determined.

By dividing the image into multiple sections, each section will have a narrower range of frequencies. Furthermore, estimating four different orientations and distances to the fishnet provides redundancy and allows for comparison and scaling of confidence scores.

In this approach, the center of each section becomes the new center for performing the FFT analysis and will be used for the the 2D points in Equation 4.15, while the camera intrinsic matrix remains the same. The solvePnP algorithm can be applied to each section using the same set of 3D points as described in Section 4.3.5. This process generates non-zero components of the translation vector $\mathbf{t}$ in the x and y axes.

Since each section has its own rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, the 3D points in each section are defined in their respective world frames. To make these points comparable, they need to be described in a common frame. Transforming them to the camera frame is a straightforward choice since all four sections have their own rotation and translation matrices. The middle point of each section can be used to describe the plane, as it is equal to the translation vector $\mathbf{t}$ of that section.

To select the most reliable points, a confidence score can be utilized. If there are four qualified points, the three points with the highest average confidences should be chosen.

Let $\mathbf{p_i}, \quad i = 1, 2, 3$ denote the three best points. The plane can be represented as $ax + by + cz = d$, where the normal vector of the plane $\mathbf{n} = (a, b, c)$ can be calculated using the following equations:

$$\begin{aligned} \mathbf{v_1} &= \mathbf{p_3} - \mathbf{p_1} \\ \mathbf{v_2} &= \mathbf{p_2} - \mathbf{p_1} \\ \mathbf{n} &= \mathbf{v_1} \times \mathbf{v_2} \\ d &= dot(\mathbf{n}, \mathbf{p_3}) \end{aligned} \tag{4.24}$$

The normal vector will be the z-axis of the world frame. To ensure that the normal vector points in the same direction as the camera coordinate system, it should have a positive z component. If it does not, the normal vector and d need to be negated. The z distance from the AUV's camera to the middle of the fishnet can be found by setting x and y to zero in the plane equation and solving for z:

$$z = \frac{d}{c} \tag{4.25}$$

Now the world frame should be defined on the plane, with x and y equal to zero. This provides the translation vector to the camera frame $\mathbf{t} = \begin{bmatrix} 0 & 0 & z \end{bmatrix}^T$.

The rotation matrix $\mathbf{R} = \mathbf{R}_w^c$ from the world frame to the camera frame should rotate the world frame's normal vector to align it parallel to the camera frame's z-axis. Additionally, the x and y axes of the world frame should be parallel to the fishnet's grid. Since each of the four split piece of the fishnet has estimated the orientation of the fishnet cells, their combined estimation can be used. Assuming the fishnet cells are square, only one of the basis vectors is necessary to define an axis of the world frame, and the remaining axis can be obtained through a cross product of the other two. The basis vector with the highest combined confidence score is selected. To determine the combined basis vector $\bar{\mathbf{b}}$, the weighted arithmetic mean can be used, where the confidence scores act as weights. The calculation can be expressed as:

$$\bar{\mathbf{b}} = \frac{\sum_i^n \mathbf{b_i} \cdot w_i}{\sum_i^n w_i} \tag{4.26}$$

Here, $\mathbf{b_i}$ represents each of the basis vectors in the 2D image plane. To align the world frame with the fishnet's grid cells, one of the axes should point in a projected direction of $\bar{\mathbf{b}}$ on the plane, while the other axis should be perpendicular to both $\bar{\mathbf{b}}$ and the z-axis in the world frame. By extending the basis vector to three dimensions it can be written as $\mathbf{B} = \begin{bmatrix} \bar{\mathbf{b}} & 0 \end{bmatrix}$. Assuming that $\mathbf{B}$ points in the x-axis of the grid cells, the y-axis of the world frame can be calculated as:

$$\text{y-axis} = \hat{\mathbf{n}} \times \hat{\mathbf{B}} \tag{4.27}$$

Here, $\hat{\mathbf{n}}$ and $\hat{\mathbf{B}}$ are the normalized vectors of $\mathbf{n}$ and $\mathbf{B}$. The x-axis of the grid should be perpendicular to both the y-axis and z-axis of the world frame:

$$\text{x-axis} = \text{y-axis} \times \hat{\mathbf{n}} \tag{4.28}$$

To derive the rotation matrix from the world frame to the camera frame, the following matrix can be assembled:

$$\mathbf{R}_{\mathbf{w}}^{\mathbf{c}} = \begin{bmatrix} \text{x-axis} \\ \text{y-axis} \\ \text{z-axis} \end{bmatrix} \tag{4.29}$$

If the y-axis basis vector of the grid has the highest confidence, the cross product of Equation 4.27 and Equation 4.28 need to have a reversed order.

An alternative plane estimation technique, based on Singular Value Decomposition (SVD) [26], offers an alternative approach when all four points exhibit high confidence. The plane fitting process using SVD is outlined as follows:

The 3D points $(x_i, y_i, z_i), \quad i = 1, 2, 3, 4$ are organized into matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix} \tag{4.30}$$

By performing SVD on matrix $\mathbf{A}$, the singular value decomposition is obtained:

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{A} \tag{4.31}$$

where matrices $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ are derived, and $\mathbf{\Sigma}$ represents a diagonal matrix containing the singular values. The last row of matrix $\mathbf{V}$ (referred to as $(\mathbf{v}_{4,:})$) is extracted and divided by its last element to yield the plane coefficients:

$$\mathbf{B} = \frac{\mathbf{v}_{4,:}}{\mathbf{v}_{4,4}} \tag{4.32}$$

Normalization is performed on the first three elements of $\mathbf{B}$ by dividing them by the Euclidean norm of $\mathbf{B}_{0:3}$):

$$\mathbf{B}_{0:3} = \frac{\mathbf{B}_{0:3}}{\|\mathbf{B}_{0:3}\|} \tag{4.33}$$

The estimated plane equation is expressed as:

$$\mathbf{B}_0 x + \mathbf{B}_1 y + \mathbf{B}_2 z + \mathbf{B}_3 = 0 \tag{4.34}$$

Furthermore, the normal vector of the plane is given by:

$$\mathbf{n} = (-\mathbf{B}_0, -\mathbf{B}_1, -\mathbf{B}_2) \tag{4.35}$$

If required, this plane estimation method can be extended to incorporate additional points.

4.4 Detect junction points and pose estimation

After detecting the fishnet using the FFT algorithm, the next step is to estimate the complete pose of the fishnet relative to the AUV and detect the junction points of the fishnet. A proposed junction detection algorithm 4 can be used for this purpose. Since the FFT algorithm can estimate the orientation and z-direction of the fishnet, assuming it is approximately a plane, the junction

point detection algorithm only needs to estimate the last two degrees of freedom, which are the x and y directions of the net's position.

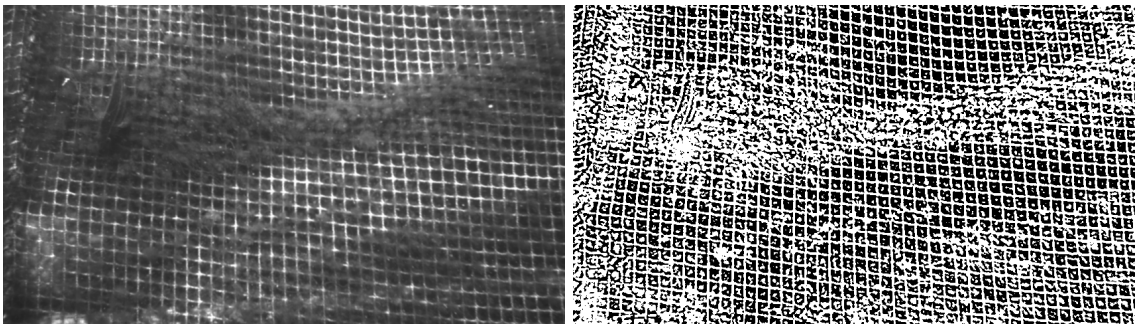
The proposed junction detection algorithm follows several steps to detect the junction points of the fishnet grid:

- Convert the image to grayscale
- Blur the image using a Gaussian filter
- Apply adaptive thresholding to the blurred image
- Find contours in the thresholded image
- Loop over all contours and filter out contours that do not meet certain criteria based on their area and aspect ratio.
- For each remaining contour, find the four corners of the minimum area rectangle that encloses it.
- Append all the corners to a list of junctions and perform DBSCAN clustering on the list
- Calculate the centroids of the clusters, which correspond to the detected junction points.
- Find the minimum rectangle that covers all the junction points
- Return the junction points and the center of the minimum rectangle

Firstly, the input image is converted to grayscale to simplify the subsequent processing steps. Then, a Gaussian filter is applied to blur the image, which reduces noise and smoothens the edges, making it easier to identify the contours.

Next, an adaptive thresholding method from the OpenCV library is applied to the blurred image to obtain a binary image. The method calculates a threshold value for each pixel in the image based on the local neighborhood of that pixel. This is different from global thresholding, which applies the same threshold to all pixels in the image.

To calculate the threshold value for each pixel, the algorithm takes a small rectangular neighborhood around each pixel and calculates the mean or Gaussian-weighted mean of the pixel values in that neighborhood. The Gaussian-weighted mean gives more weight to pixels that are closer to the center of the neighborhood, resulting in a smoother transition between foreground and background regions. The threshold value for that pixel is then set to be a certain amount above that mean value. This process is repeated for all pixels in the image, resulting in a new image where each pixel is either set to white or black based on whether its intensity value is above or below the threshold value.



(a) Fishnet (b) Adaptiv threshold applied on fishnet

Figure 4.7: Applying adaptive threshold

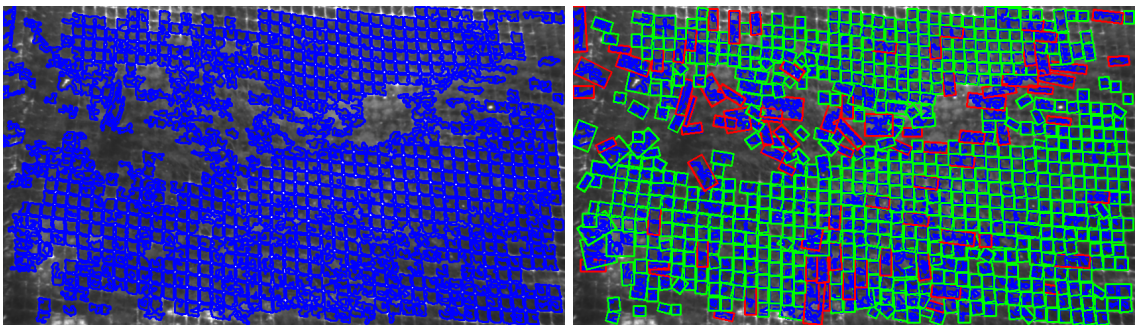
The benefit of using adaptive thresholding is that it can compensate for non-uniform lighting and contrast in the image, as well as variations in the background and foreground intensity values.

This helps to segment the fishnet from the background in a more robust way compared to global thresholding, which may result in over-segmentation or under-segmentation in areas with different illumination. The resulting binary image is a black and white image where the white pixels correspond to the fishnet and the black pixels correspond to the background as shown in Figure 4.7.

After obtaining the binary image, the findContours method from the OpenCV library is applied to identify the contours. However, not all contours are deemed relevant for further processing. In order to filter out any spurious contours that do not correspond to the fishnet, only contours whose area falls within the range of

$$\text{thresh\_min} < \text{contour\_area} < \text{thresh\_max} \tag{4.36}$$

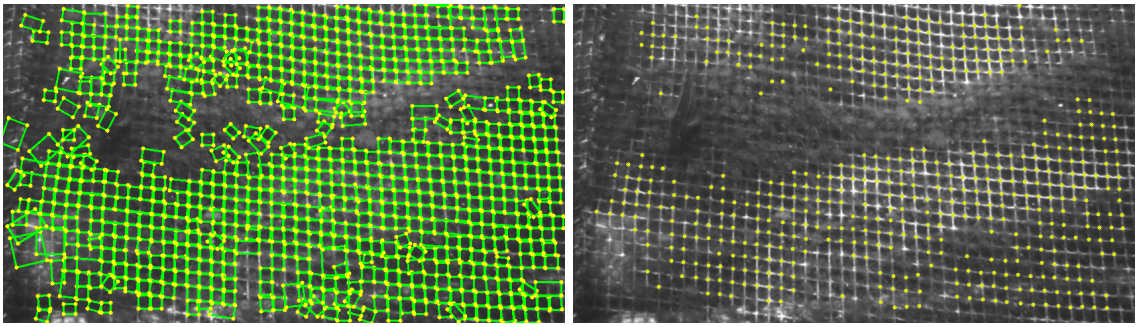
are considered. The values of thresh_min and thresh_max are determined based on the estimated area of a fishnet cell, which is calculated using the basis vectors obtained from the FFT algorithm. By applying this filtering step, the algorithm is able to eliminate unwanted contours and focus only on the relevant ones. The resulting contour is then displayed in Figure 4.8a.



(a) Fishnet with contours (b) fishnet with boxes

Figure 4.8: Box estimation of fishnet

For each remaining contour, the minimum area rectangle that encloses it is identified using the minAreaRect function from OpenCV. These rectangles are assessed for their degree of squareness and their area compared to the contour area. Only rectangles with a degree of squareness above a set threshold and a contour-to-rectangle area ratio within a certain range are considered valid. The resulting figure is shown in Figure 4.8b, where the green boxes represent accepted rectangles and the red boxes represent rejected ones.



(a) Fishnet with accepted boxes (b) Fishnet with junction points

Figure 4.9: Finding junction points

The coordinates of the corners of the accepted rectangles are added to a list of potential junction points of the fishnet, as shown in Figure 4.9a, together with the accepted boxes. These potential junction points are then processed by the DBSCAN clustering algorithm to filter out unreliable points and detect the actual junction points. The two main parameters from the DBSCAN algorithm: min_samples and eps, represents the min_samples parameter specifies the minimum

number of potential junction points required to form a cluster and the maximum distance between two points for them to be considered part of the same cluster respectively. In an ideal scenario, each internal junction of the fishnet would have four potential points stacked on top of each other. However, this is often not the case due to imperfect fishnets in images and algorithmic imperfections. Therefore, to account for such deviations, min_samples is set to three. The center of each cluster, which is computed as the mean of the points in the cluster, is considered a valid junction point and added to the final list of junction points, as shown in Figure 4.9b.

To determine the x and y components of the fishnet's pose, the algorithm finds the smallest possible rectangle that covers all the junction points, and its center is used as the x and y position. However, if the rectangle almost covers the entire image, the center of the image may be used as the x and y position, since it is likely that the fishnet covers the entire image in this case. Note that the junction points algorithm will not detect junction points close to the image since it requires at least three potential junction points to consider it as an actual junction.

Once the x and y components of the fishnet are found, along with the previously estimated orientation and z component from the FFT algorithm, the 6 DOF pose of the fishnet, estimated as a plane, is illustrated in Figure 4.10. For visualization purposes, the y-axis and z-axis are flipped to highlight the z-axis pointing outward. The axes are color-coded, with red representing the x-axis, green representing the y-axis, and blue representing the z-axis. The axes are also scaled to three times the cell size.

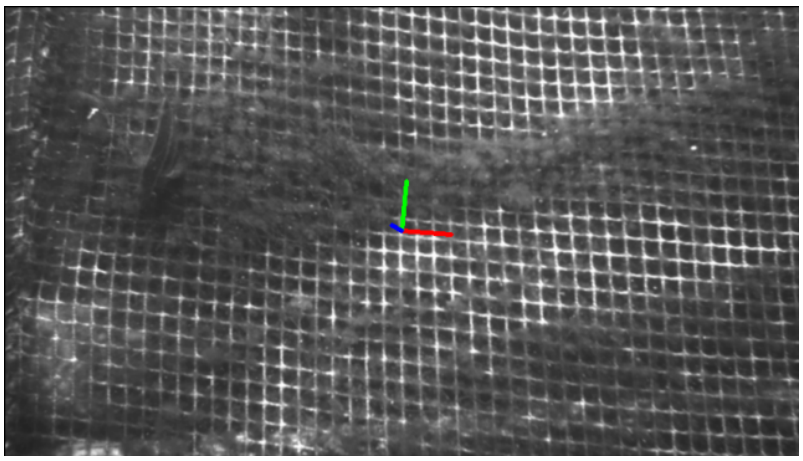


Figure 4.10: Fishnet with coordinate axis

To demonstrate the positioning of the world frame's origin, a virtual grid that does not cover the entire image can be utilized. Figure 4.11 displays the virtual fishnet with its coordinate axis, highlighting that the origin is near the center of the grid.

The proposed junction point detection algorithm is summarized in pseudo-code in Algorithm 4.

4.4.1 Variable area estimation

In the context of fishnet contour detection, it is observed that the area of fishnet cells can vary considerably when the fishnet image is tilted. To address this issue, a large threshold value for the estimated cell area can be utilized, but this approach may lead to a high number of false positives of objects other than the cells. A more effective solution is to utilize a variable area estimation method that is based on the planar estimation of the fishnet. In this method, the estimated cell area decreases as the distance from the optical center of the camera increases.

The estimated cell area can be obtained using the FFT algorithm. It is expected that the estimated cell area will be close to the average size of a cell, which should be close to the center cell. However, when the image is tilted, the side length of a cell that is further away from the optical center will be shorter. To estimate the pixel size of a side of a fishnet cell, two lines of equal length s, representing

Algorithm 4 Junction point detection

```
function DetectJunctionPoints(image, thres_min, thres_max, eps, min_samples)
    Convert the image to grayscale
    Blur the image using a Gaussian filter
    Apply adaptive thresholding to the blurred image
    Find contours in the thresholded image

    junctions = [ ]
    for contour in contours do
        contour_area = controurArea(contour)
        if thresh_min < contour_area < thres_max then
            rect = minAreaRect(contour)
            if rect is close to square and has similar area as contour_area then
                for corner in rect do
                    junctions.append(corner)
                end for
            end if
        end if
    end for

    dbscan = DBSCAN(eps, min_samples)
    clusters = dbscan.fit_predict(junctions)

    cluster_labels = Unique(clusters)
    n_clusters = len(cluster_labels) - 1
    cluster_centers = [ ]
    for i in 0 to n_clusters - 1 do
        cluster_points = junctions[clusters == i]
        cluster_center = mean(cluster_points)
        cluster_centers.append(cluster_center)
    end for

    if len(cluster_centers) ≥ 3 then
        rect = minAreaRect(cluster_centers)
        midpoint = center of rect
    end if

    return cluster_centers, midpoint
end function
```

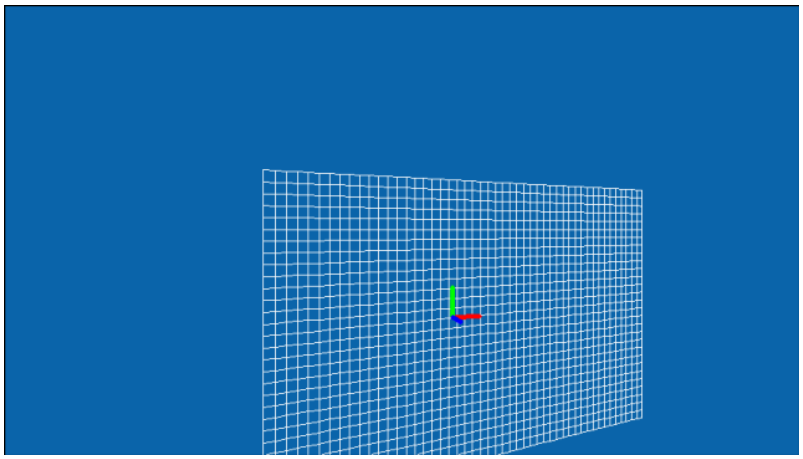


Figure 4.11: Virtual fishnet with coordinate axis

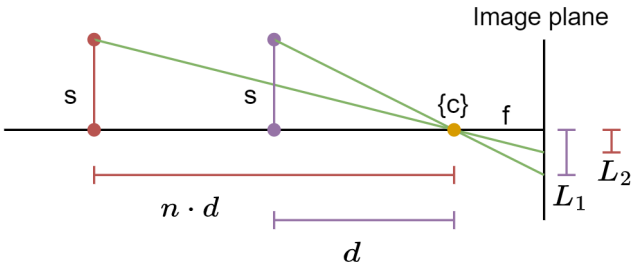


Figure 4.12: Scale of cells

the side length of a cell at a distance d and $n \cdot d$ away from the optical center, are drawn through the optical center of the camera frame c and intersect the image plane as shown in Figure 4.12. The resulting lengths on the image plane, L_1 and L_2, are related to the side length of a cell s and the focal length of the camera f using the principal of similar triangles as follows:

$$\frac{s}{L_1} = \frac{d}{f} \implies L_1 = \frac{s \cdot f}{d} \tag{4.37}$$

$$\frac{s}{L_2} = \frac{d \cdot n}{f} \implies L_2 = \frac{s \cdot f}{d \cdot n} = \frac{1}{n} L_1 \tag{4.38}$$

$$\tag{4.39}$$

Based on the principle of similar triangles, it can be concluded that the resulting pixel size of a side of a fishnet cell is $1/n$ times the length of the side for a cell that is n times further away. Therefore, the total area of a cell that is n times further away should be $1/n^2$ of the size of the cell.

To find out how far away each cell of the fishnet are relative to the center cell, a simple approach would be to only calculate the four corner values and then linear interpolate the rest of the pixel on the image to find all the n values. The first step would be to find the 3D points of the corners by using the inverse projection or 2D to 3D projection as described in Section 3.1.3.

Since all the fishnet junction points are assumed to lay on a plane with $Z^w = 0$, the 3×3 extrinsic transformation matrix $\mathbf{H}$ can be defined as in Equation 3.7 and $\mathbf{K}$ is the intrinsic camera matrix. Inverse projection of the four homogeneous corner points $\tilde{\mathbf{u}}_i = [u_i, v_i, 1], \ i = 0, 1, 2, 3$. The combined matrix containing the vectors can be defined as:

$$\tilde{\mathbf{U}} = \begin{bmatrix} \tilde{\mathbf{u}}_0 \\ \tilde{\mathbf{u}}_1 \\ \tilde{\mathbf{u}}_2 \\ \tilde{\mathbf{u}}_3 \end{bmatrix} \tag{4.40}$$

Then the 2D to 3D projection of these points is calculated as in Equation 3.9, just extended to four vectors instead of one:

$$\begin{bmatrix} \tilde{X}_0^w & \tilde{X}_1^w & \tilde{X}_2^w & \tilde{X}_3^w \\ \tilde{Y}_0^w & \tilde{Y}_1^w & \tilde{Y}_2^w & \tilde{Y}_3^w \\ \tilde{W}_0^w & \tilde{W}_1^w & \tilde{W}_2^w & \tilde{W}_3^w \end{bmatrix} = \mathbf{H}^{-1}\mathbf{K}^{-1}\tilde{\mathbf{U}}^T \tag{4.41}$$

where Z^w is excluded from the equation since its 0. The homogeneous 3D vectors can be written as $\tilde{\mathbf{X}}_i^{\mathbf{w}} = [\tilde{X}_i^w/\tilde{W}_i^w, \tilde{Y}_i^w/\tilde{W}_i^w, 0, 1] = [X_i^w, Y_i^w, 0, 1], \;\; i = 0, 1, 2, 3$. Now to find the $n \cdot d$ z-distance values for each of the corners, the four corners in world coordinates $\{w\}$ can be transformed to the camera frame $\{c\}$. This is done by performing the extrinsic camera transform from Equation 3.2:

$$\begin{bmatrix} \mathbf{X}_0^c & \mathbf{X}_1^c & \mathbf{X}_2^c & \mathbf{X}_3^c \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}_0^w & \tilde{\mathbf{X}}_1^w & \tilde{\mathbf{X}}_2^w & \tilde{\mathbf{X}}_3^w \end{bmatrix} \tag{4.42}$$

where $\mathbf{R}$ and the z value of $\mathbf{t}$ are found from the FFT method. The x and y values are just assumed to be zero, which is in the middle of the image. If the fishnet is not centered then that would be a bad approximation, but the x and y value are not calculated before applying the junction detection algorithm, so they are unknown at this stage of the algorithm.

Since the estimated area of a fishnet cell from the FFT method is for a center cell, the distance d as illustrated in Figure 4.12 is the z value of the vector $\mathbf{t} = [0, 0, t_z]$, and $n \cdot d$ is $Z_i^c, \;\; i = 0, 1, 2, 3$, the cell area scale value $1/n_i^2$ can be calculated by:

$$1/n_i^2 = (\frac{t_z}{Z_i^c})^2, \qquad i = 0, 1, 2, 3 \tag{4.43}$$

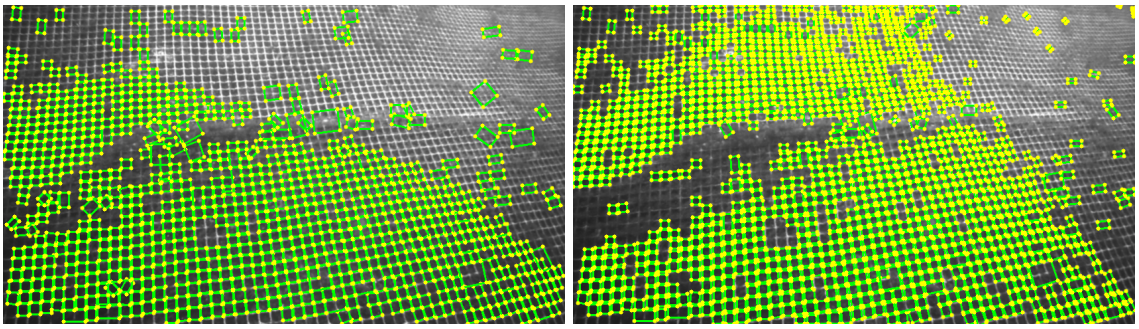
Algorithm 5 Variable area estimation for junction detection

function VARIABLEAREA(image, cell_area, $\mathbf{R}, \mathbf{t}, \mathbf{K}$)
  h, w = shape of image
  pts_2d = List($[[0, 0, 1], [0, h, 1], [w, 0, 1], [w, h, 1]]$)
  $\mathbf{H}$ = hstack($\mathbf{R}[:, :2], \mathbf{t}$)
  pts_3d_world = inv($\mathbf{H}$) × inv($\mathbf{K}$) × pts_2d$^\mathsf{T}$ ▷ 3D coordinates: $[\tilde{X}^w, \tilde{Y}^w, \tilde{W}^w]$
  pts_3d_world = normalize_homogeneous(pts_3d_world)
  pts_3d_world$[2, :] = \mathbf{0}$ ▷ Add Z=0
  pts_3d_world = vstack(pts_3d_world, $\mathbf{1}$) ▷ Add W=1

  $\mathbf{H}$ = stack($\mathbf{R}, \mathbf{t}$)
  pts_3d_camera = $\mathbf{H}$ × pts_3d_world
  scaling = ($\mathbf{t}[2]$/pts_3d_camera$[2, :]$)^2
  corner_areas = cell_area × scaling

  left_edge_areas = linspace(corner_areas$[0]$, corner_areas$[1]$, h)
  right_edge_areas = linspace(corner_areas$[2]$, corner_areas$[3]$, h)
  scaled_cell_areas = [linspace(i, j, w) for i, j in zip(left_edge_areas, right_edge_areas)]
  return scaled_cell_areas
end function

Utilizing this variable area estimation method results in more accurate detection of fishnet cells, particularly in tilted images. This is illustrated in Figure 4.13, where it can be observed that a large number of cells were detected using the variable area estimation method as opposed to using a fixed threshold.



(a) Without variable area estimation (b) With variable area estimation

Figure 4.13: Improvements with variable fishnet area threshold

4.5 Optical flow

In this section, the theoretical foundations of optical flow discussed in Section 3.3, are built upon. As mentioned, extensive exploration has been conducted on optical flow estimation methods, but traditional approaches may encounter limitations when employed in challenging underwater environments. Therefore, in order to tackle these challenges and ensure the accurate tracking of the junction points identified in Section 4.3 and Section 4.4, the utilization of a deep learning-based optical flow method, namely the Global Motion Aggregation (GMA) approach [27], is proposed.

Traditional optical flow methods often struggle to handle the relatively small AUV when it is experiencing significant motion, resulting in substantial image frame displacement, changes in illumination, and the repetitive pattern of the fishnet. Moreover, the presence of occlusions in the form of fish, sea snow, and other objects within fish farms can further exacerbate the inaccuracies associated with traditional optical flow techniques.

To overcome these limitations, deep learning-based optical flow methods, such as the GMA approach, offer a promising alternative. The GMA approach leverages the power of deep neural networks to learn intricate motion patterns directly from the data, enabling enhanced robustness against changes in illumination and challenging environments. In their study, the authors propose a GMA-based approach specifically designed to estimate optical flow in occluded regions by modeling image self-similarities and performing global motion aggregation.

It is demonstrated by the authors that optical flow estimates in occluded regions are significantly improved by the GMA approach while preserving performance in non-occluded regions. The effectiveness of this method is highlighted by the establishment of new benchmarks on the demanding Sintel dataset. Through the adoption of the GMA approach, enhanced robustness against occlusions caused by the abundance of fish, sea snow, and other objects present in the underwater environment of the fish farm is anticipated. However, it is important to acknowledge that optimal performance with deep learning-based methods typically necessitates a substantial amount of training data.

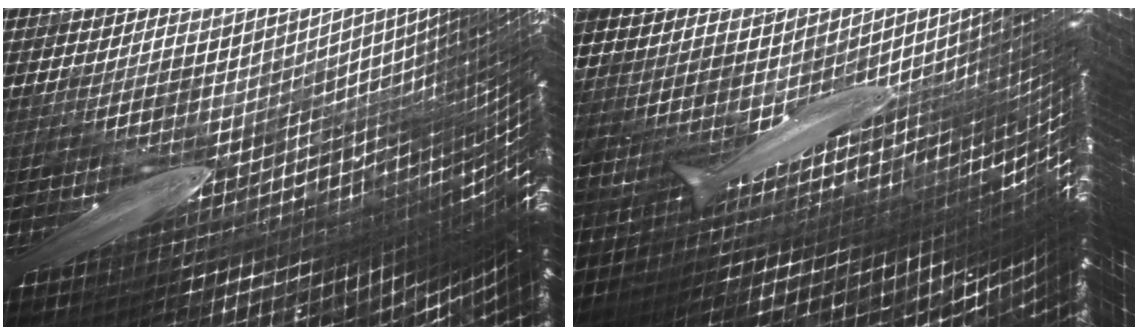
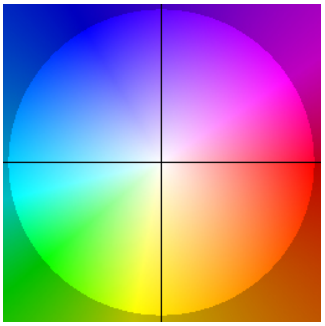


Figure 4.14: Underwater netpen images

To demonstrate the effectiveness of GMA, two consecutive images captured within a fish farm setting are depicted in Figure 4.14. In this scenario, the fish exhibits an upward and rightward motion, while the fishnet displays a simultaneous leftward displacement. In order to visualize the dense flow estimated by the GMA method, the utilization of Hue-Saturation-Value (HSV) color coding scheme is employed. The resulting flow image, displaying the calculated flow vectors, is portrayed in Figure 4.15a.

Notably, the application of GMA yields a clear segmentation between the fishnet and the fish, with the flow vectors accurately representing the motion patterns. This highlights the superior performance of GMA in handling the challenges posed by large fish displacement and the estimation of motion within the repeated pattern of the fishnet, which would typically pose difficulties for traditional optical flow methods.

(a) Optical flow using GMA

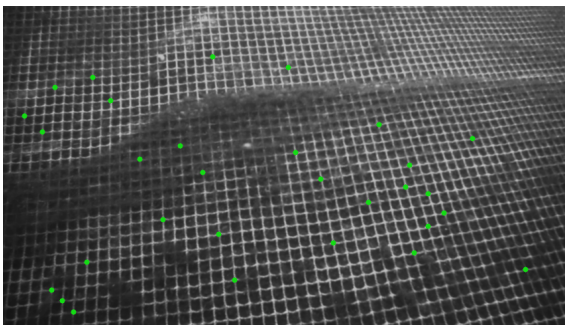


(b) HSV colors for optical flow

4.5.1 Tracking junction points

In this subsection, the utilization of the GMA optical flow method for estimating the flow and tracking the junction points within the fishnet will be discussed. The objective is to demonstrate the effectiveness of the GMA approach in accurately capturing the motion patterns and tracking the desired points of interest.

The junction points of the fishnet, as depicted in Figure 4.13, are targeted for optical flow estimation and tracking. To avoid overcrowding the image, a subset of the junction points is selected, as shown in Figure 4.16a. The GMA optical flow method is applied to calculate the dense flow field between the current image and an image 10 frames ahead, allowing for significant motion between the frames while maintaining the applicability of the GMA method.

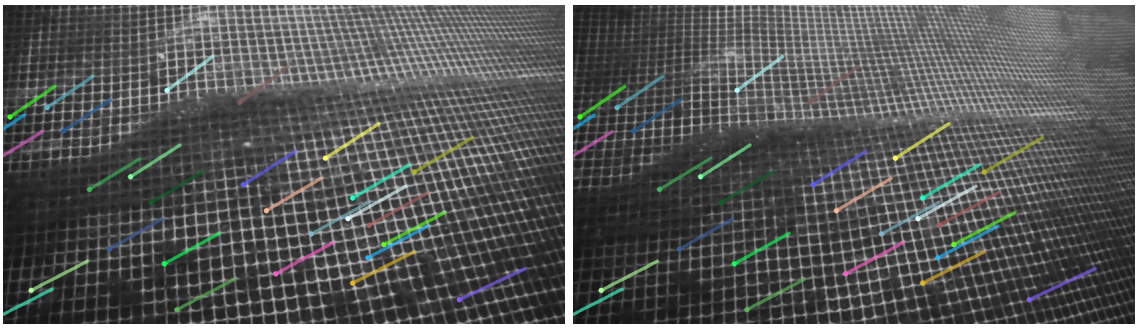


(a) Subset of junction points

(b) Optical flow of fishnet

The junction points will serve as the initial position for the points that are being tracked. By applying the optical flow vector at the pixel of each of the associated junction points, the sparse trailing vectors are drawn behind each point, illustrating their historical motion paths. The motion vectors are drawn both on top of the starting image in Figure 4.17a and on top of the ending image in Figure 4.17b, showing that junction points are correctly tracked.

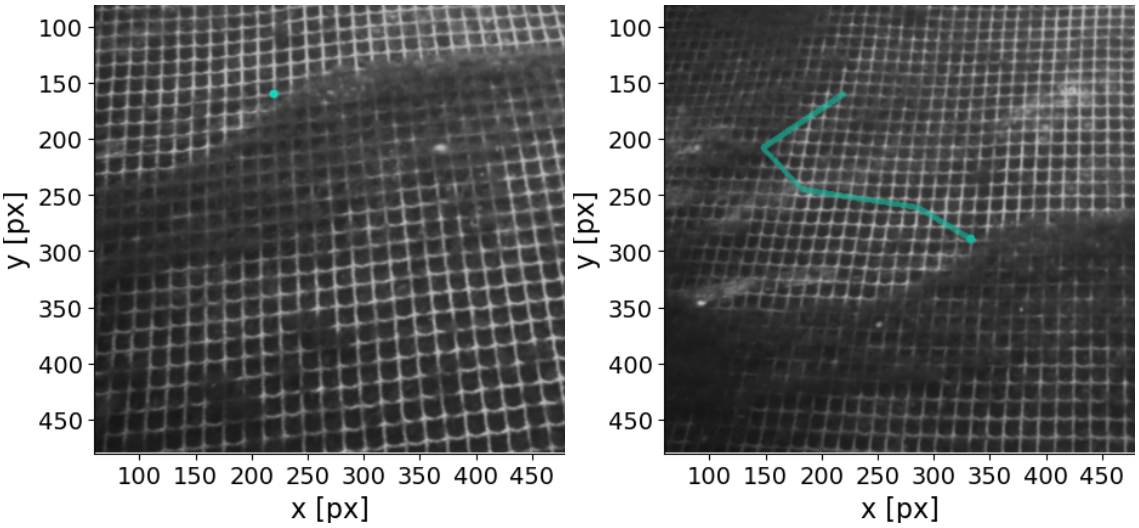
To assess the tracking performance and evaluate the accuracy of the GMA method, a specific



(a) Flow vectors drawn on the first image (b) Flow vectors drawn on the second image

junction point is selected as the target for tracking over multiple frames. The motion of this selected point is continuously monitored, and a comparison is made against the expected position based on its initial location, as illustrated in Figure 4.18a. The resulting tracking performance after four frames is depicted in Figure 4.18b.

By leveraging the GMA optical flow method, reliable tracking of the fishnet junction points is achieved, enabling the analysis of their motion patterns and facilitating subsequent motion-based analysis and applications in the underwater environment.



(a) Initial starting position of junction point (b) Tracked junction point over 4 frames

In summary, the proposed method for optical flow estimation using GMA is expected to provide robust and accurate results in the challenging underwater environment of the fish farm. The use of GMA allows for reliable data on the motion of the points of interest to be obtained, which is critical for subsequent analyses. The choice of optical flow method depends on the specific application requirements and environmental conditions.

Chapter 5

Results

This chapter presents the results obtained from the evaluation and testing of the developed algorithm for fishnet detection and tracking using an AUV. The primary objective of this chapter is to provide a thorough analysis of the algorithm's performance, including its accuracy, robustness, and limitations. The results are presented in a systematic manner, covering different aspects of the algorithm's functionality and its comparison with existing methods.

The chapter begins with an evaluation of the algorithm's performance using virtual grids, which allow controlled testing of various scenarios. These tests aim to assess the algorithm's capability to detect fishnets under different conditions, such as varying angles, distances, and occlusion. Through the examination of the algorithm's performance in these controlled settings, insights can be gained into its limitations and the behavior it exhibits in challenging situations.

Following the evaluation on virtual grids, the algorithm is tested on real-life images captured in fish farming environments. This evaluation aims to validate the algorithm's performance under realistic conditions, where lighting variations, low visibility, and the presence of fish can pose challenges to accurate fishnet detection. The results obtained from these real-life image tests provide valuable insights into the algorithm's applicability in practical scenarios and its ability to handle the complexities associated with fish farming operations.

Furthermore, a comprehensive comparison is made between the developed FFT algorithm and existing FFT algorithm, reported in the literature. This comparison allows for an objective assessment of the algorithm's performance in relation to other state-of-the-art approaches. The evaluation is based on the previous FFT algorithms results, where each methods strengths and weaknesses are identified.

5.1 Peak filtering and peak value tuning

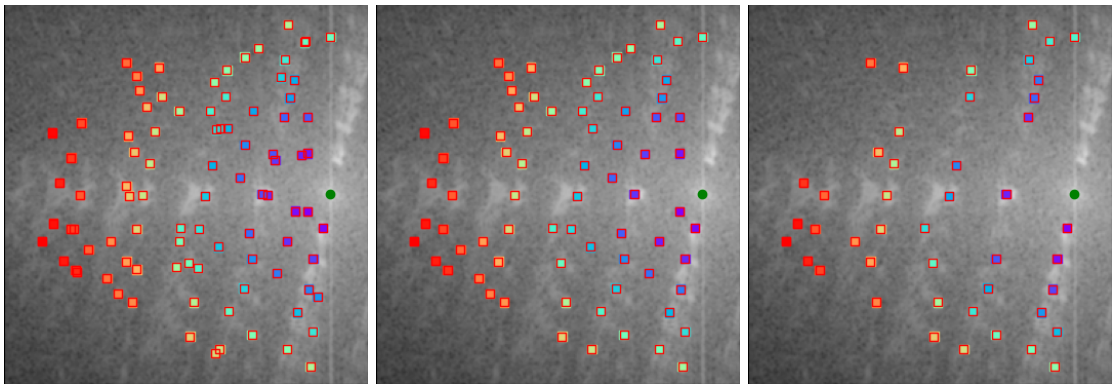
This section focuses on the results of the tuning of the FFT algorithm to recognize the basis vectors by filtering out irrelevant peaks and determining the correct peaks.

5.1.1 Peak Filtering

The peak filtering process aims to refine the results of the peak finding algorithm by removing irrelevant peaks and retaining the ones that are more likely to represent the basis vectors in the FFT image.

Image Figure 5.1a illustrates an example of the peaks detected using the peak detection algorithm (Algorithm 2) without the DBSCAN. In this particular case, a total of 92 peaks were initially identified, with a threshold parameter set to 20 and a neighborhood value of 16. Additionally,

peaks located outside an elliptical region were filtered out during the detection process.



(a) Peaks inside ellipse (b) Filtered clusters (c) Lower threshold filter

Figure 5.1: Filtering peaks from frequency domain image

The next step involves clustering the peaks and keeping only the largest peak in each cluster. The DBSCAN algorithm is employed with an epsilon value of 10 and a minimum number of samples of 1. As a result, the number of peaks is reduced to 74 as shown in Figure 5.1b.

Finally, a lower threshold filter is applied to further refine the peaks. The upper value is determined as the middle peak value minus 40. After applying this filter, only 45 peaks remain, which is approximately half of the initial number of peaks. The filtered peaks are illustrated in Figure 5.1c.

5.1.2 Peak value tuning

In order to identify the basis vectors of the FFT image, the peak value tuning algorithm 3 was developed. This algorithm assigns a total value to each peak based on its intensity pixel value, beam value, and repeated pattern value. To determine the optimal weights for these factors, their importance and impact on the algorithm's robustness were considered.

The weights (k_1, k_2, k_3) were initialized as 1, and an example of peaks found in an FFT image is illustrated in Figure 5.2. The contribution of each factor for the four largest peaks (p_1 to p_4) and their total values are summarized in Table 5.1.

The peak value is an important indicator, but the difference in values among the peaks is not significant. Therefore, it should be weighted more to increase its significance. The beam value consistently shows larger values in the direction of the correct peaks, making it suitable for filtering out incorrect peaks. The repeated pattern value reflects the presence of a recurring pattern in the FFT image, but its significance may vary depending on the image's characteristics.

	Peak value	**Beam**	**Repeated Pattern**	**Total value**
$\mathbf{p_1}$	68	34	149	252
$\mathbf{p_2}$	64	24	116	204
$\mathbf{p_3}$	54	33	38	125
$\mathbf{p_4}$	52	23	28	104

Table 5.1: Largest peak values with $k_1 = 1$, $k_2 = 1$, $k_3 = 1$

Based on the analysis, p_1 and p_2 exhibited the highest peak values and obtained the largest total values, indicating their potential as basis vectors. Confidence scores of p_1 and p_2 are calculated by:

$$\text{conf}_1 = \frac{252}{125} = 2.02, \qquad \text{conf}_2 = \frac{204}{125} = 1.63 \tag{5.1}$$

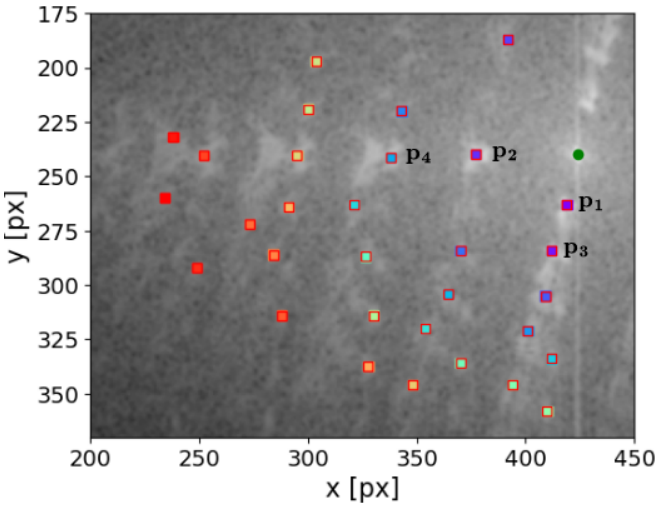


Figure 5.2: Four largest peak values

	Peak value	Beam	Repeated Pattern	Total value
p1	102	68	149	319
p2	96	48	116	260
p3	81	67	38	186
p4	78	47	28	154

Table 5.2: Largest peak values with $k_1 = 1.5$, $k_2 = 2$, $k_3 = 1$

To adjust the weights and prioritize certain criteria, the weights were modified to $k_1 = 1.5$, $k_2 = 2$, and $k_3 = 1$. The recalculated total values using the modified weights are shown in Table 5.2. By adjusting the weights, the algorithm becomes more proficient at accurately selecting the basis vectors.

The updated confidence scores were calculated, with p_1 having a score of 1.72 and p_2 having a score of 1.40. The decreased scores still indicate the significance of these peaks in determining the correct basis vectors. Considering the angle difference, p_3 is not chosen as the second basis vector due to its small angle difference from p_1. Instead, the confidence score of p_2 is compared to that of p_4, resulting in a score of 1.69 for p_2, indicating its higher significance.

It is important to note that the choice of weights may vary depending on the characteristics of the FFT images and the desired criteria for peak selection. Experimentation and analysis of various weight configurations can help determine the optimal weights for a given application.

5.2 Evaluation on Virtual Grids

This section focuses on the evaluation of the algorithm's performance using virtual grids, which provide controlled testing environments for assessing its effectiveness. The objective is to analyze the algorithm's behavior and limitations under various conditions, including different angles, distances, and occlusion scenarios. Through a comprehensive evaluation in these controlled settings, valuable insights can be gained regarding the algorithm's performance and its applicability in challenging situations.

5.2.1 Closest Detectable Distance of Grid

When the fishnet is in close proximity, the peaks in the frequency domain become close together, resulting in a higher density of peaks. This is illustrated in Figure Figure 5.3. The intensity values

of each peak are similar, making it challenging to identify the correct basis vectors. However, by utilizing the assigned three weights values discussed in Section 5.1.2, with emphasis on identifying repeating patterns in the peak directions, the algorithm can prioritize the actual basis vectors. The estimated distance to the net is 0.598 meters, while the actual distance is 0.6 meters, given that the fishnet cells are 5cm × 5 cm.

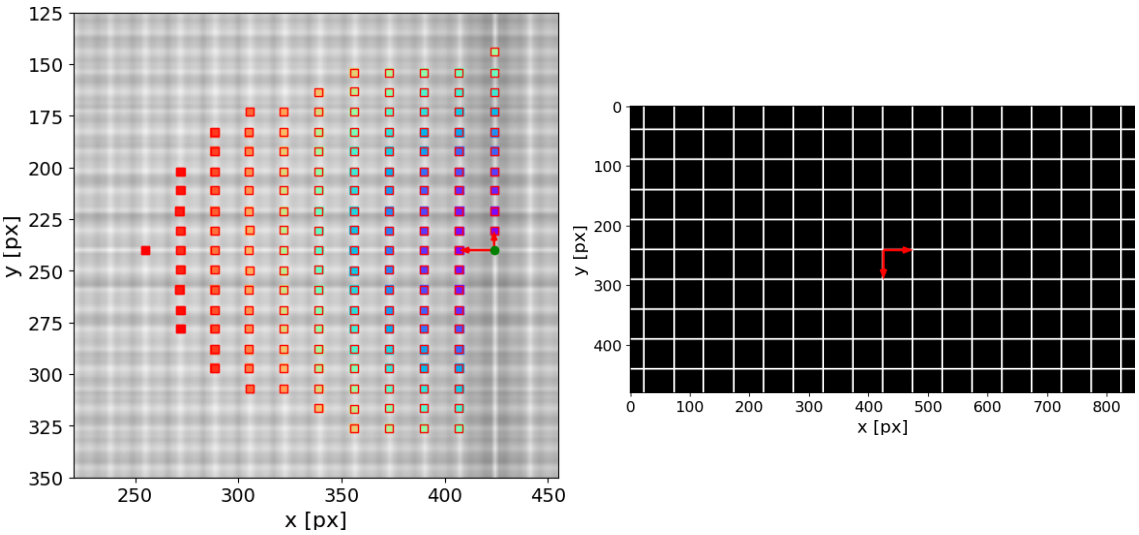


Figure 5.3: Peak detection of fishnet 60 cm away

The FFT algorithm has a maximum number of peaks it considers during the detection process. Initially, all peaks are identified, and then they are sorted based on their pixel intensity values. For ideal grid images, many peak will have similar or equal values, possibly resulting in not having the correct peaks for the basis vectors in the number of peaks checked. Empirical testing on realistic fishnet images suggests that it is sufficient to check 20 peaks or even fewer.

5.2.2 Largest detectable distance of grid

In addition to determining the closest detectable distance of the grid, it is essential to assess the algorithm's performance at larger distances. This subsection presents the results obtained when detecting a grid at distances of 3 meters and 4 meters from the camera.

At a distance of 3 meters, the algorithm successfully identifies the grid's basis vectors, as shown in Figure 5.4. The confidence scores of the two basis vectors are calculated as 1.37 and 1.29, respectively and the estimated distance is 3.003 meters.

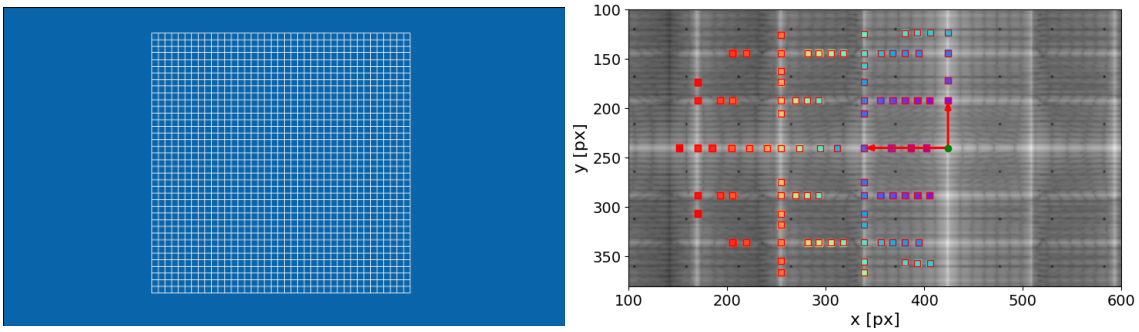


Figure 5.4: Peak detection of fishnet 3 meters away

However, when the grid is placed 4 meters away from the camera, the algorithm encounters challenges in correctly identifying the basis vectors. In this case, a new peak appears between the origin and the actual basis vector peaks, as shown in Figure 5.5. Despite having lower pixel intensity

values of 208 and 203 compared to the actual basis vector peaks (225 and 209), this new peak achieves a higher total value due to the increased repeating pattern values. Consequently, the algorithm misidentifies this intermediate peak as part of the basis vectors, leading to inaccurate distance estimation. The chosen peaks has confidences of 1.15 and 1.02 and should therefore not be accepted as the basis vector peaks by the algorithm.

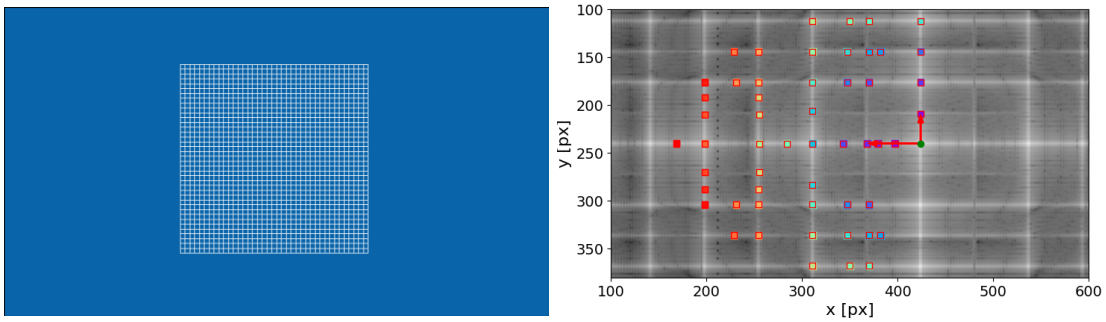


Figure 5.5: Peak detection of fishnet 4 meters away

5.2.3 junction points

To evaluate the performance and tuning of the junction point detection algorithm, a virtual grid depicted in Figure 4.11 was utilized. The grid's center was positioned 3 meters away and tilted at a 30-degree angle, creating a realistic scenario.The grid consists of 30×50 lines. The result of applying adaptive threshold is shown in Figure 5.6, where almost every cell is accurately captured, enabling precise detection of the rectangles, as illustrated in Figure 5.7. However, some of the smaller cells were less accurately captured.

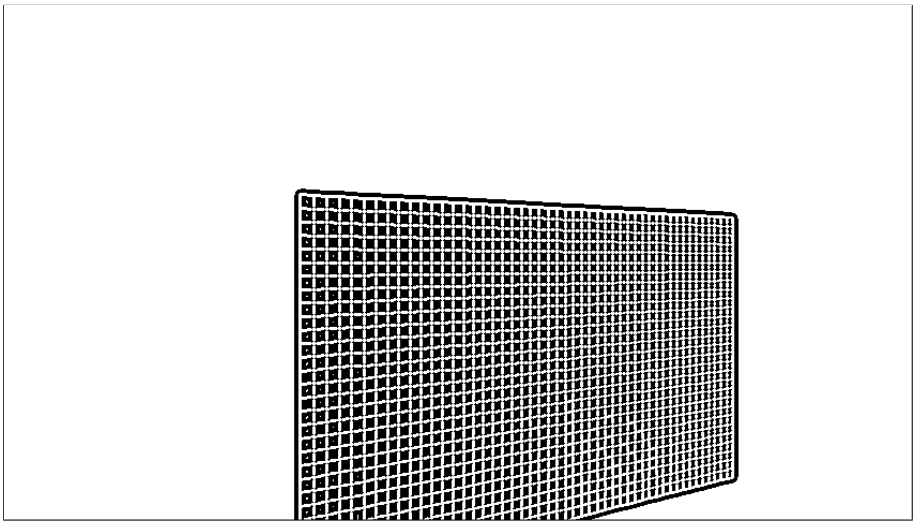


Figure 5.6: Threshold of tilted grid

The junction points detected, as illustrated in Figure 5.8, were determined using the DBSCAN algorithm with an epsilon value of 6. This setting allowed for the identification of potential junction points within a distance of 6 pixels, forming clusters that constituted the detected junction points. Out of a potential count of $30 \times 50 = 1500$, with the exception of points located outside the frame, a total of 878 junction points were successfully detected. It is worth noting that the algorithm does not capture points at the edge of the grid due to the requirement of at least 3 points for a junction to be considered.

The x and y position of the pose of the fishnet is found in Figure 5.9, indicated by the red circle. The green square is the minimum rectangle that covers all the junction points, and the center of the rectangle is chosen as the red dot.

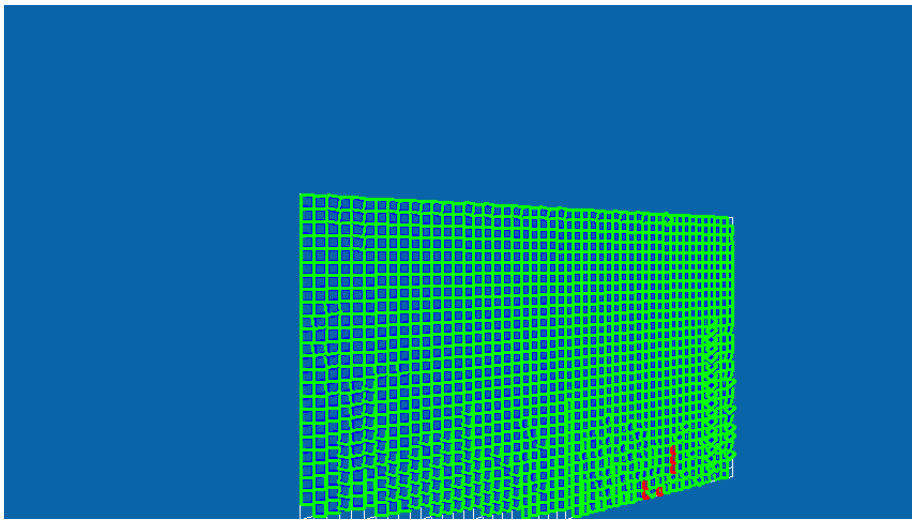


Figure 5.7: Accepted and denied boxes tilted grid

The x and y position of the fishnet's pose was determined and presented in Figure 5.9, where the red circle indicates its location. The green square represents the minimum rectangle that encompasses all the junction points, with the center of the rectangle chosen as denoted by the red dot.

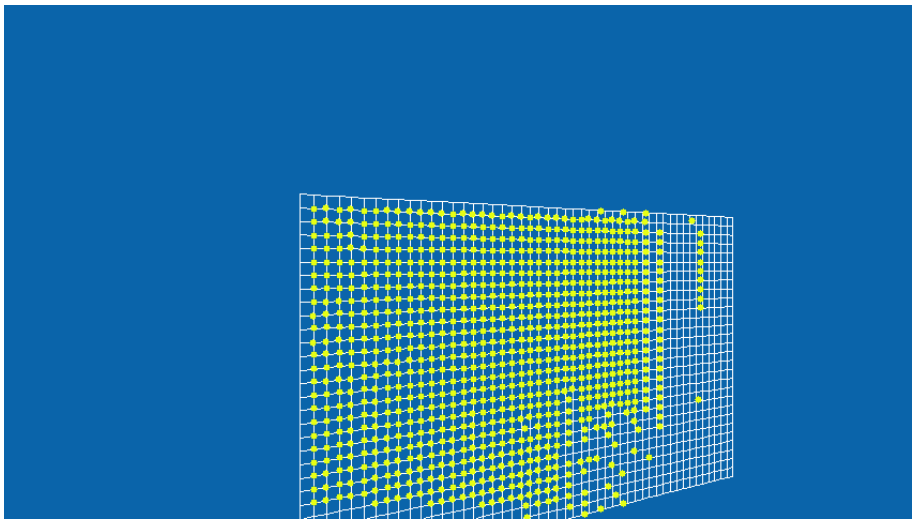


Figure 5.8: Junction points of tilted grid

5.3 Evaluation on Real-Life Images

The algorithm is tested on real-life images captured in fish farming environments in this section. The purpose of this evaluation is to validate the algorithm's performance under realistic conditions, where factors such as lighting variations, low visibility, and the presence of fish can introduce complexities to accurate fishnet detection. The results obtained from these tests provide valuable insights into the algorithm's applicability in practical scenarios and its ability to handle the inherent challenges of fish farming operations.

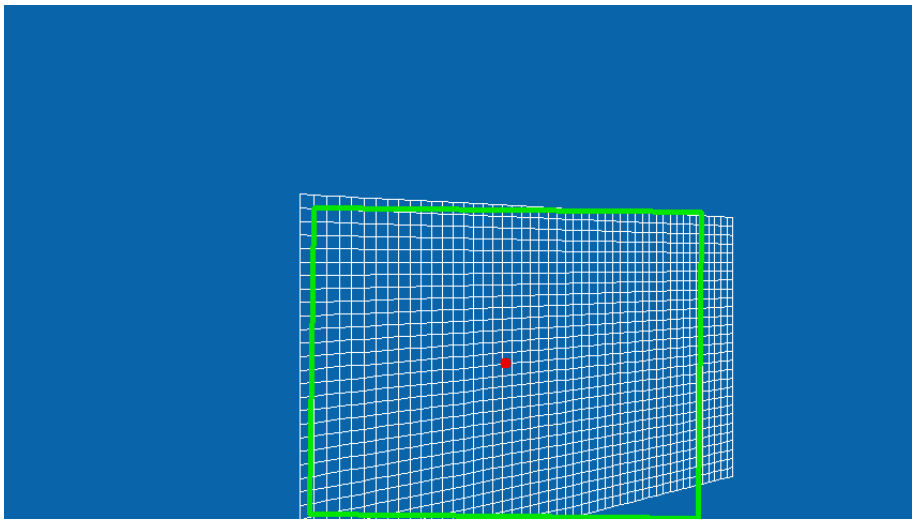


Figure 5.9: Estimated area and center of tilted grid

5.3.1 Largest detectable distance of fishnet

To assess the detectability of the fishnet in relation to its distance from the AUV's camera, Figure 5.10 was used.

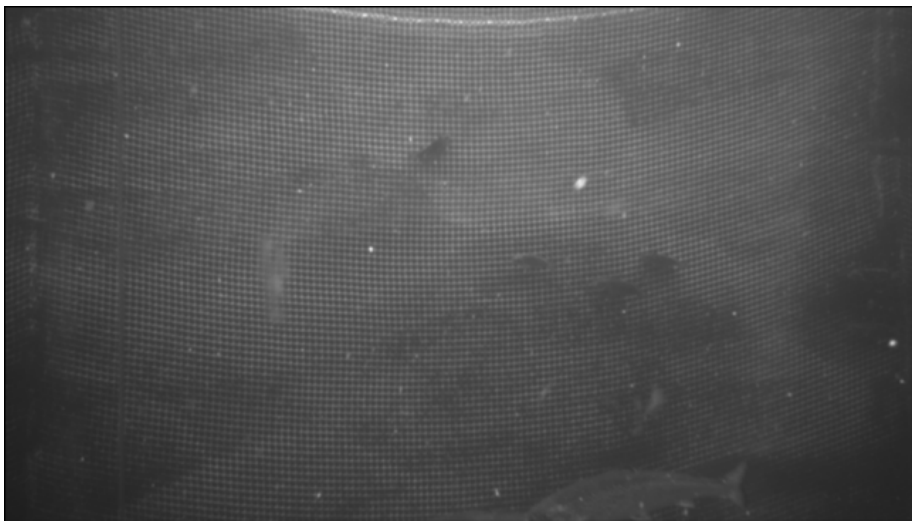


Figure 5.10: Fishnet at a large distance

The peaks found with the FFT method are shown in Figure 5.11. The right image shows that the estimated basis vectors almost align with the grid cells. The confidence of each basis vector are 1.19 and 1.17, which is not the most reliable values. The estimated distance to the fishnet is 5.15 meters, assuming a 5cm $\times$ 5cm fishnet grid cell.

5.3.2 Multi-layer fishnet

A challenging environment for fishnet detection and pose estimation is when fish farms use multiple layers of fishnets for redundancy. This creates multiple sets of frequencies and poses on top of each other, to create redundancy. This is displayed in Figure 5.12.

The peak vectors that make up the basis vectors are found in Figure 5.13a. The vectors have confidence scores of 1.41 and 1.32. The time domain basis vectors of the grid are displayed in Figure 5.13b.

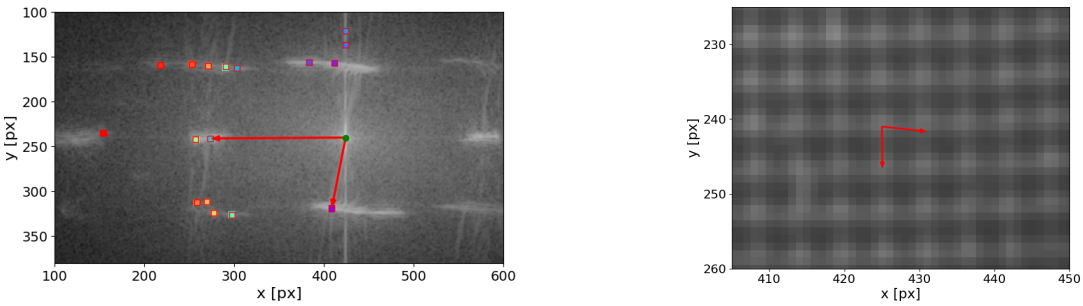


Figure 5.11: Peak detection of far away fishnet

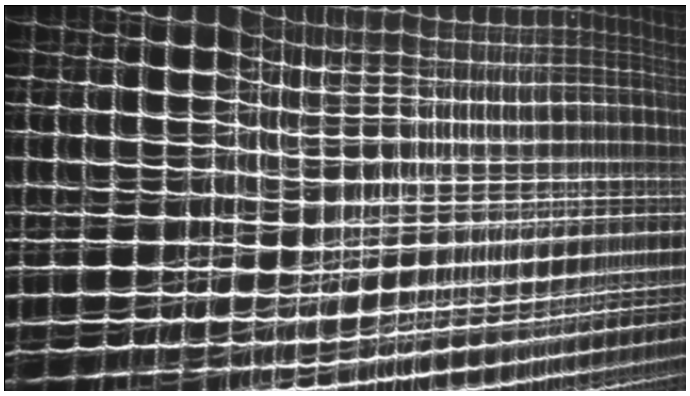


Figure 5.12: Multi-layer fishnet

The multiple layered fishnet creates sub rectangles within each fishnet cell, making it hard for the junction point algorithm to detect the junction points of the furthest out fishnet. The resulting image after applying adaptive threshold on Figure 5.12, is shown in Figure 5.14a. The threshold method detects much of the fishnets behind as well, making the contours detected not meeting the area and and shape requirements needed to be counted as a fishnet cell. The detected fishnet cells are shown in Figure 5.14b, showcasing the relative poor performance.

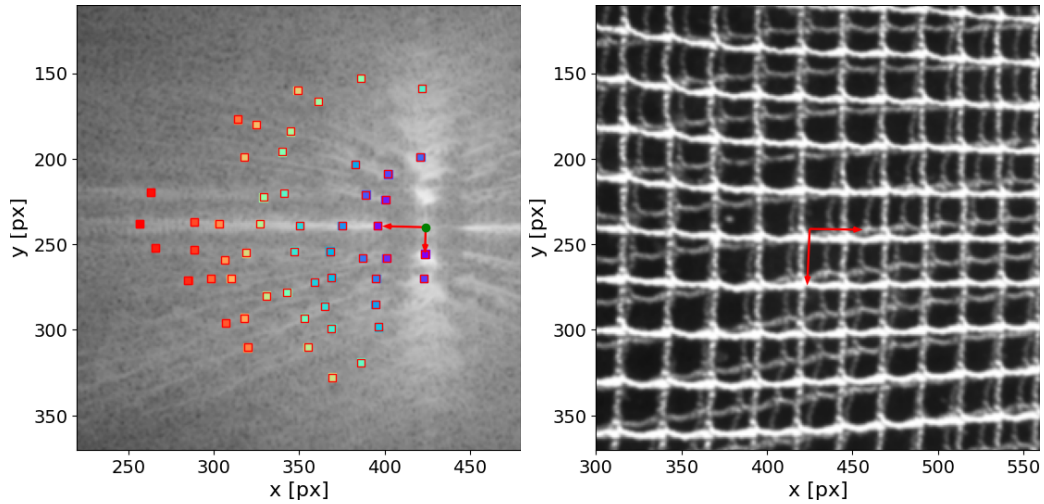
5.3.3 Occlusion

Occlusion poses a common challenge in fish farm environments, where fish can swim in between the AUV and the fishnet, resulting in parts of the fishnet being obscured. Additionally, the presence of growth on the fishnet can hinder the detection of junction points by the proposed algorithm and reduce the clarity of the FFT algorithm in detecting peaks. However, these occlusions and growth create distinct features that can be utilized for optical flow tracking.

In this experiment, the improved FFT algorithm described in Section 4.3.6 and the variable area estimation technique discussed in Section 4.4.1 were employed to detect the junction points of an occluded fishnet image (Figure 1.3a). The detected junction points are shown in Figure 5.15.

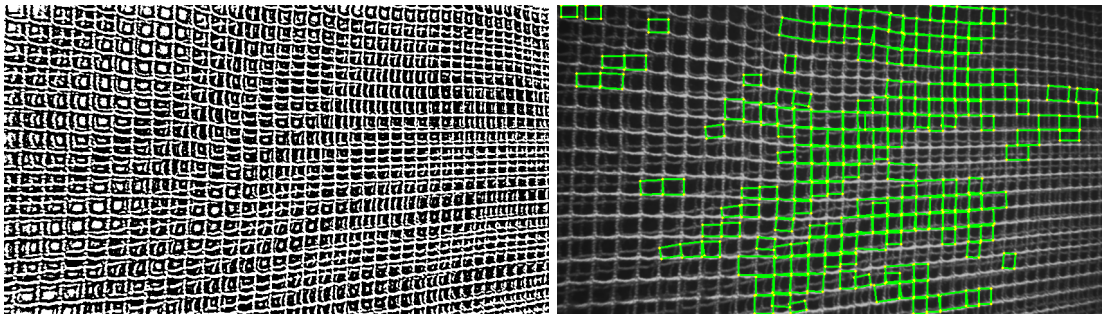
The junction points detected exhibited a high level of accuracy at the closer distance junctions. However, as the distance from the AUV increased, fewer junctions were detected, and the accuracy of the detection decreased.

The estimated pose of the fishnet is depicted in Figure 5.16. The algorithm estimated the distance to the fishnet to be 2.25 meters, assuming a fishnet cell size of 5cm x 5cm. The confidence scores obtained from the four split sections of the FFT algorithm ranged from 1.66 to 2.28, indicating a reliable estimation.



(a) FFT peaks of multi-layer fishnet (b) Multi-layer fishnet with basis vectors

Figure 5.13: Frequency analysis of multi-layer fishnet



(a) Adaptive threshold of multi-layer fishnet (b) Detected cells of multi-layer fishnet

Figure 5.14: Cell detection of multi-layer fishnet

5.4 Comparison with Existing Methods

The developed FFT algorithm is based on a previous FFT method, which has been previously discussed in the literature. However, certain parameters and settings used in previous work are not explicitly stated. To address this, the grid size chosen for this thesis is set at 5x5 cm. The camera matrix used is given in Equation 5.2, and the image size is set to 480x848 pixels, consistent with the image data captured by the AUV. To replicate the approach of merging the virtual fishnet into an underwater scene, the grid is overlaid on top of an underwater scene image with transparent grid lines, as illustrated in Figure 5.17.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 600 & 0 & 848/2 \\ 0 & 600 & 480/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

The current algorithm does not perform well at detecting fishnets at close ranges, making it unable to recognize the fishnet at 40 cm away. Therefore, the comparison starts at an 80 cm distance.

Tables 5.3, 5.4, and 5.5 present the compared results of the FFT method with the tilt angles of the grid of 0°, 20°and 40°respectively, at various distances. It can be noted that for distances at 120 cm and below, certain parameter adjustments were made, such as reducing the DBSCAN filtering method for close peaks, adjusting the lower threshold, and refining the repeated pattern search area and the ellipse area where peaks are identified.

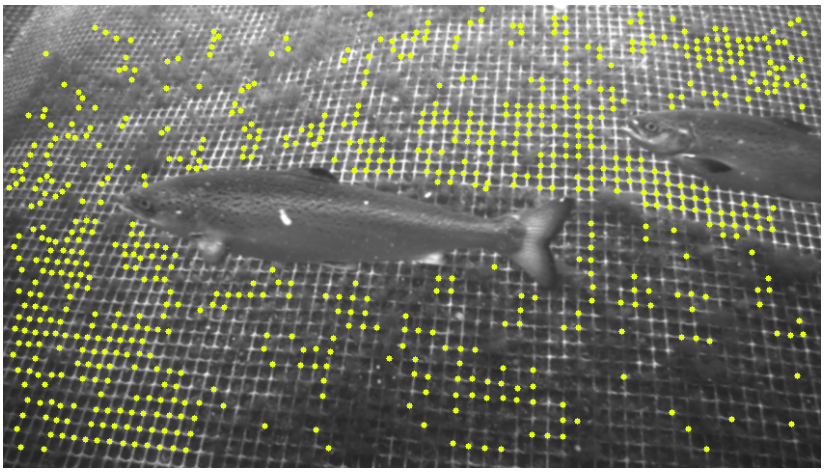


Figure 5.15: Detected junction points of occluded fishnet

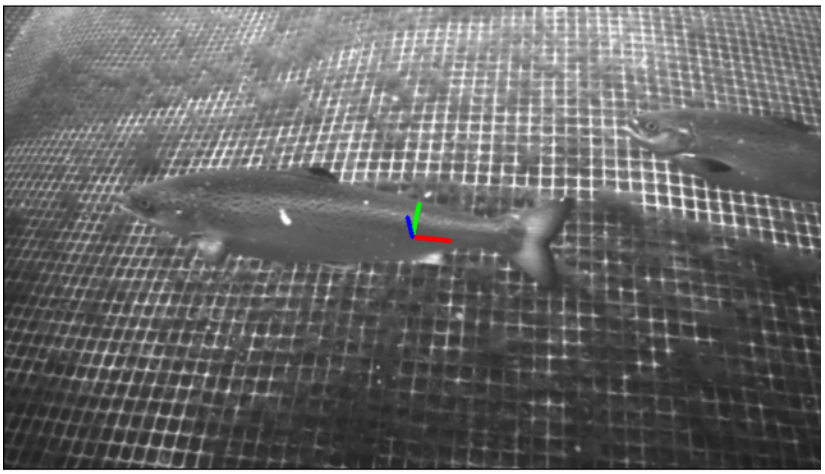


Figure 5.16: Estimated pose of occluded fishnet

All the values in the tables are in cm, except for the angles, which are written in parenthesis.

5.5 Tracking

To examine the drift of estimation error over time, 100 flow estimations using the GMA optical flow method were performed. The images were filmed with a frame rate of 10 Frames Per Second (FPS). The initial starting point, selected as a junction point, is shown in Figure 5.18. The tracking of this point over the next 99 frames is displayed in Figure 5.19a. The estimated junction point coordinates are $(171, 167)$, while the ground truth is $(177, 170)$, resulting in a distance of approximately 8 pixels for the 100 images or 99 flow estimates.

The performance of tracking over 100 images, can be compared directly to obtaining the optical flow from the first to the last image, as shown in Figure 5.19b. In addition, this also illustrates the performance of tracking over large distances. The resulting estimated position, rounded to the nearest whole pixel, was found to be $(176, 171)$, resulting in a significantly improved proximity to the ground truth. The average optical flow of all pixels in the image is calculated as $(-113.5, 62.1)$.

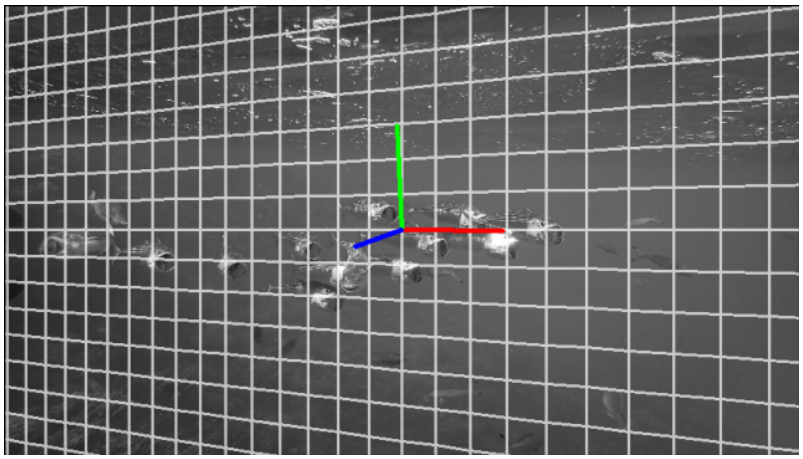


Figure 5.17: Grid in underwater scene, 80 cm away with 20 degrees angle

z [cm]	Baseline method	Proposed method
80	78.56 (4.9°)	77.3 (4.2°)
120	118.2 (4.7°)	118.8 (0.2°)
160	156.15 (6.6°)	159.7 (0.05°)
200	198.5 (5.2°)	201.7 (1.5°)

Table 5.3: Compared results of FFT method with 0° tilt of grid

In contrast, when tracking every individual frame and summing the optical flow of each image, the resulting average optical flow vector is $(-162.4, 65.9)$. To ensure a more accurate optical flow estimate, 100 pixels were removed from each edge to avoid including pixels that could go out-of-frame, in the average calculation.

Furthermore, an additional test was conducted to evaluate the performance of tracking the in a loop, back to the initial image. The 25 additional images were added to fill the remaining frames to get the tracked point back to the initial point. The resulting image is shown in Figure 5.20. The estimated ending pixel is $(278, 45)$ compared to the starting pixel at $(287, 55)$. The average summed optical flow vector, after removing a 100-pixel border, is $(-38.9, 6.5)$. Ideally, this vector should be $(0, 0)$.

pixel close to each other have a very similar optical flow vector for the most. This means that it should be possible to estimate the optical flow on an image with reduced pixel density/lower resolution, and thereby lowering the computational cost. It should also be possible to train an optical model specific to the type of data and reduce parameters used in the model to increase speed. Also note that the optical used in this is only run on the CPU, making it a lot slower, than what it possibly could be. Uses on average 3 seconds per optical flow estimate.

z [cm]	Baseline method	Proposed method
80	80.06 (16°)	84.4 (22.3°)
120	119.21 (11.6°)	120.4 (22.8°)
160	158.75 (1.6°)	160.8 (16.98°)
200	200.76 (4.2°)	199.6 (13.78°)

Table 5.4: Compared results of FFT method with 20° tilt of grid

z [cm]	Baseline method	Proposed method
80	79.13 (5.3°)	73.8 (21.7°)
120	119.05 (2.8°)	104.2 (18.7°)
160	153.38 (5.9°)	149.0 (32.5°)
200	197.51 (6.2°)	179.5 (33.5°)

Table 5.5: Compared results of FFT method with 40° tilt of grid

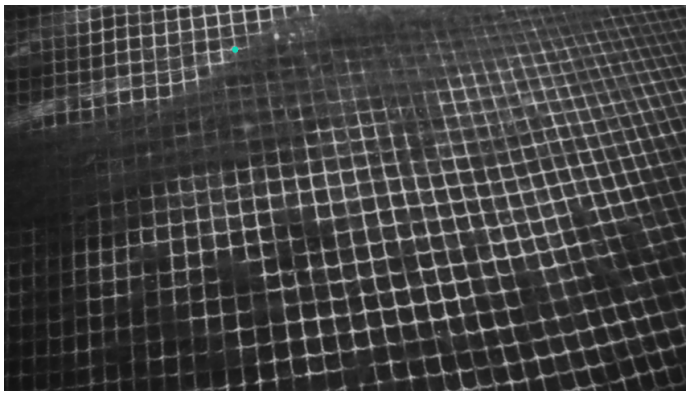
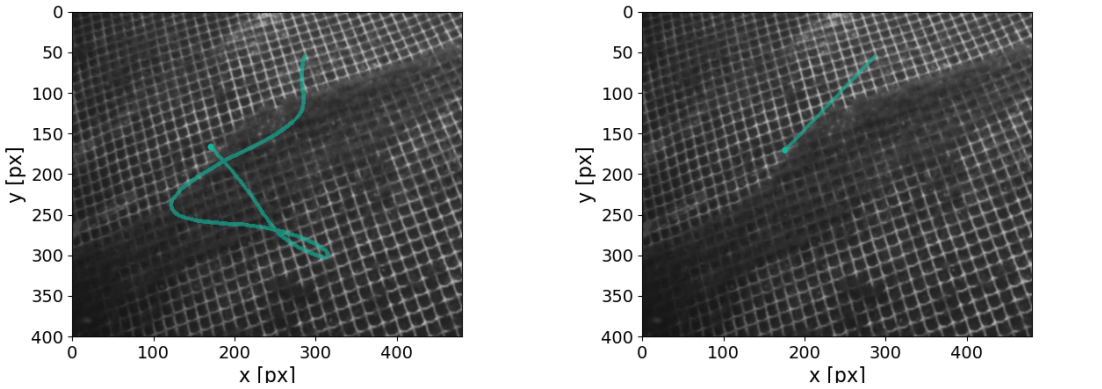


Figure 5.18: Starting junction point for tracking



(a) Tracking point over 100 images

(b) Tracking point directly from first to last image

Figure 5.19: Tracking junction points

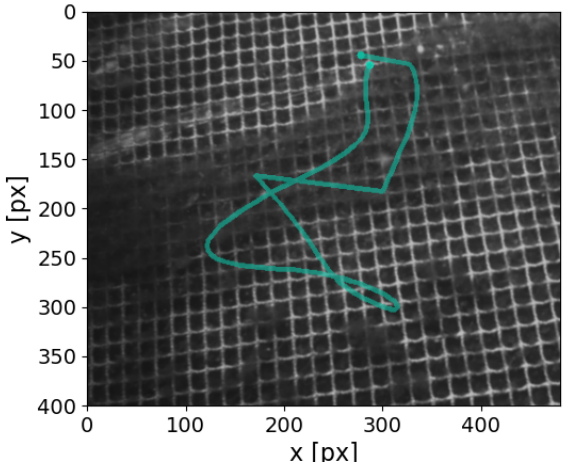


Figure 5.20: Tracking back to start

Chapter 6

Discussion

The discussion chapter analyzes and evaluates key components of the proposed algorithm. The peak filtering process enhances peak detection accuracy through clustering and threshold filtering. The value tuning algorithm assigns weights and uses confidence scores for peak selection. Performance evaluation considers detectable range, challenges with closer fishnets and multi-layer configurations, and the impact of occlusions. The algorithm's accuracy in detecting junction points, a comprehensive comparison with an existing method, and the effectiveness of GMA optical flow for fishnet motion tracking are also discussed. The chapter concludes by suggesting refinements to improve algorithm robustness and practical applications in fish farming.

6.1 Peak Filtering and Value Tuning

The peak filtering and value tuning process significantly improved the results obtained from the peak finding algorithm, enhancing the identification of relevant peaks corresponding to the basis vectors in the FFT image. Initially, 92 peaks were detected within an elliptical region using a threshold value of 20 and a neighborhood value of 16. The subsequent application of the DBSCAN algorithm with an epsilon value of 10 and a minimum number of samples of 1 reduced the number of peaks to 74 by eliminating duplicate and false peaks.

Further refinement was achieved through a lower threshold filter, where the upper value was determined by subtracting 40 from the intensity value of the middle peak. This additional filtering step reduced the number of remaining peaks to approximately 45, approximately half of the initial count.

The results demonstrate the effectiveness of the peak filtering and value tuning process in removing extraneous peaks and retaining the most relevant ones for subsequent analysis. However, it is important to note that the order of the filtering steps can impact the speed performance of the algorithm. Further investigation and optimization of the peak filtering process can lead to improved computational efficiency and speed.

Exploring different strategies and fine-tuning approaches in the peak filtering process can further enhance the overall efficiency of the algorithm. By carefully adjusting the parameters and order of the filtering steps, it is possible to achieve even better results in peak identification and analysis.

The results of the peak value tuning algorithm demonstrated its effectiveness in selecting the correct basis vectors from the FFT images. By assigning weights to the peak value, beam value, and repeated pattern value, the algorithm was able to filter out incorrect peaks and accurately identify the basis vectors. The peak value, which represents the intensity of each peak, was found to be a reliable indicator. However, it required a higher weight to increase its significance, as the differences in peak values among the selected peaks were not substantial.

The beam value, indicating the directionality of the peaks, proved to be a valuable criterion for

filtering out incorrect peaks. Peaks that were not in the direction or close to the basis vectors exhibited significantly lower beam values. By assigning a higher weight to the beam value, the algorithm effectively prioritized peaks in the correct direction. The repeated pattern value, which reflects the presence of recurring patterns in the FFT image, was also considered in the algorithm. While it demonstrated significance in the analyzed image, its reliability may vary in more challenging images with different frequency contents or titled images. Therefore, a balanced weight was assigned to the repeated pattern value to ensure a robust approach without overemphasizing this factor.

The confidence scores calculated based on the total values provided a measure of the relative strength or significance of each peak. The higher confidence scores for the selected basis vectors (e.g., p_1 and p_2) confirmed their potential as accurate representations of the basis vectors in the FFT image. Adjusting the weights allowed for further customization of the algorithm according to the specific requirements of different images. By giving more weight to the peak and beam values, while slightly downplaying the repeating pattern value, the algorithm became more proficient at accurately selecting the basis vectors.

It is worth noting that the choice of weights may vary depending on the characteristics of the FFT images and the specific criteria for peak selection. Further experimentation and analysis of various weight configurations can help fine-tune the algorithm and optimize its performance.

Overall, the peak value tuning algorithm demonstrated its effectiveness in identifying the basis vectors of FFT images. By considering and adjusting the weights of the different factors, the algorithm achieved improved accuracy and robustness in peak selection. The results obtained lay the foundation for further analysis and interpretation of the basis vectors in subsequent stages of the research.

6.2 Performance Evaluation of the FFT Algorithm

6.2.1 Detectable Range of Fishnet

The performance of the proposed FFT algorithm was evaluated through various tests, including the detection of artificial grids and real fishnets in different scenarios. These evaluations provided valuable insights into the algorithm's capabilities as well as its limitations in recognizing fishnets.

As the fishnet moved closer to the camera, the density of peaks in the frequency domain increased, presenting challenges in accurately identifying the basis vectors due to similarities in intensity values and peak clusters. In an ideal scenario with a simulated grid and a noise-free background, the similarity in peak values makes the repeating pattern factor crucial for determining the correct basis vectors, as illustrated in Figure 5.3. However, in real-world images, the number of detected peaks is lower, and there is a larger difference in intensity values.

Although the FFT algorithm could potentially detect a fishnet at an even closer distance to the camera, several challenges would arise. The increasing proximity of peaks would result in them becoming densely packed, potentially leading to being filtered away by the peak filtering algorithm. To address this, adjustments such as reducing the clustering distance of the DBSCAN algorithm and decreasing the neighborhood size in the peak detection algorithm would be required. Furthermore, as the fishnet moves closer, the peaks become less distinct, posing additional challenges to accurate detection.

On the other hand, the algorithm demonstrates promising results in detecting fishnets at larger distances. In an ideal environment, the algorithm can detect artificial grids up to a range of 3 to 4 meters. By fine-tuning the down-weighting of repeated patterns, intermediate peaks can be minimized, potentially enabling detection at even greater distances. In real images of actual fishnets, the algorithm exhibits detection up to 5 meters, indicating its practical application potential. Notably, in real scenarios, the insignificance of intermediate peaks, coupled with slight changes in the repeating pattern over large areas of the fishnet, allows the algorithm to detect larger distances in real images compared to ideal ones.

6.2.2 Challenging environments

Some fish farms utilize multi-layered fishnets for redundancy, preventing fish from escaping. However, overlapping frequencies from the different layers result in more spread-out and less distinct peaks, even in clear images of fishnets at close distances, without any growth or occlusions. Nonetheless, the algorithm is still capable of detecting the fishnet, as depicted in Figure 5.12, with acceptable confidence scores of 1.41 and 1.32 for the basis vectors.

Occlusions, which are commonly encountered in fish farm environments, were also considered in the evaluation. The algorithm, particularly the improved version that splits the image into four sections for separate analysis, demonstrated minimal effects from occlusions by fish and marine growth. Figure 5.16 showcases successful estimation of the pose of an occluded fishnet, including both growth and fish. The estimated axis aligns closely with the actual fishnets, and confidence scores in each section ranging from 1.66 to 2.28 indicate a high level of confidence in the estimates. The growth and distinct features would on the other hand make it easier for the optical flow tracking.

Evaluation on real-life fish farming scenarios revealed promising performance, despite challenges such as occlusions, uneven lighting conditions, and debris or marine growth on the fishnets. Techniques such as peak filtering were employed to refine peak detection and improve accuracy. Further refinements, parameter adjustments, and adaptive techniques based on image characteristics can enhance performance in real-life scenarios. The FFT algorithm demonstrates the ability to detect fishnets within the range of 0.6 meters to 5 meters, but for reliable results, it is recommended to use the algorithm in the range of 0.9 meter up to 4 meters.

The evaluation provided valuable insights and feedback for further improvements and optimizations of the FFT algorithm. Continued refinement and validation on diverse real-life datasets will ensure the algorithm's robustness and reliability in practical fish farming applications.

6.3 Junction point

This section focuses on the detection of junction points in fishnets, discussing the performance of the proposed algorithm and its limitations. The accuracy of the junction point detection algorithm is somewhat influenced by the accuracy of the fish cell area estimate produced by the FFT algorithm. However, the threshold in area allows the FFT algorithm to provide reasonably accurate estimates without the need for perfect precision.

The algorithm's performance was evaluated on a tilted ideal grid, as shown in Figure 5.8. It was observed that the larger cells junction points were accurately detected, while the accuracy decreased for smaller cell sizes. The main factor affecting this performance is the cluster distance parameter epsilon of the DBSCAN algorithm. Currently, for fishnet section is further away, many potential junction points are clustered together due to the smaller distance between cells. The mass center of these larger clusters will count as the estimated junction point, which often will not be located where the actual fishnet junction is.

To address this, a smaller epsilon value should be used. Reducing the epsilon value from 6 to 3 results in Figure 6.1. With this adjustment, the algorithm detected 1201 junction points, compared to the original 878 detected junction points, representing a significant increase in performance. Reducing the epsilon value too low may lead to worse performance, specially for real images. The cells and rope may not be as ideal, making the spacing between potential junction points vary more. This can be a problem for images of the fishnet taken close to the camera. Improved suggestions would be to use a adaptive approach of choosing the epsilon value based of the distance to the net, and limit the cluster sizes to maximum using the four closest potential junction points.

Detecting junction points in multi-layered fishnets poses a more challenging scenario. These fishnets create sub-rectangles within each fishnet cell, making it difficult for the algorithm to separate the layers and detect only the front fishnet layer. As a result, many of the detected contours correspond to the smaller sections within the fishnet cells, which do not meet the area and shape requirements

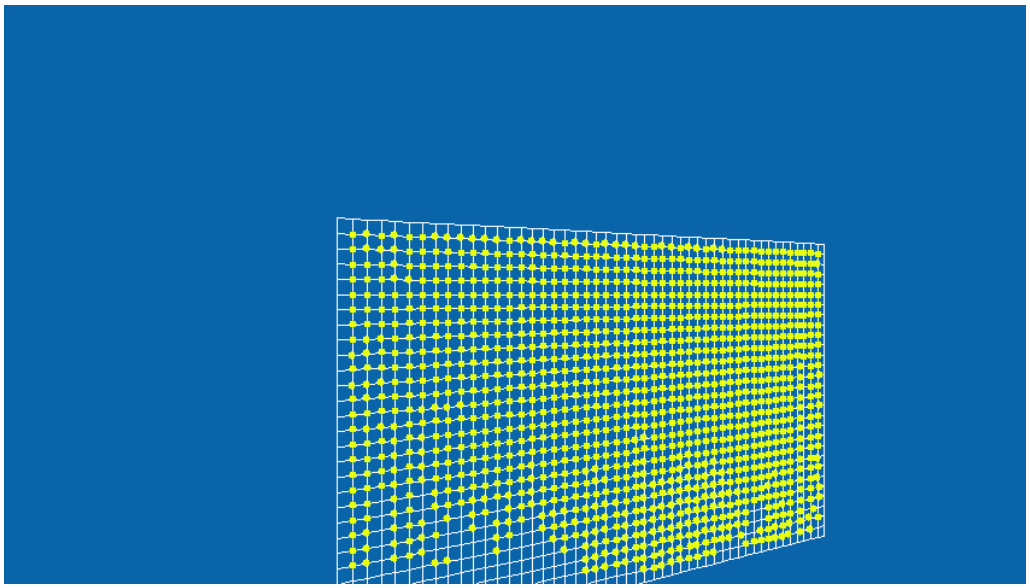


Figure 6.1: Junction points with reduced epsilon in DBSCAN

to be considered valid fishnet cells. Consequently, the algorithm's performance is relatively poor in such scenarios, and modifications are necessary to enhance its effectiveness.

In terms of occlusion, the algorithm demonstrates robust performance. Figure 5.15 illustrates that junction points in regions not covered by growth are detected with relatively high accuracy. It should be noted that the algorithm requires at least three points in a cluster to be counted as a junction point of the fishnet. Therefore, areas with extensive growth will naturally have fewer points detected.

In summary, the junction point detection algorithm shows promise in various scenarios. However, further improvements are required to enhance its performance in multi-layered fishnets. Additionally, the algorithm should be refined to optimize its performance under different occlusion conditions and improve its accuracy in detecting junction points in the presence of substantial growth.

6.4 Comparison with Existing Methods

In this section, a comprehensive comparison is made between the developed FFT algorithm and an existing FFT method reported in the literature for detecting peaks and estimating the distance and angle of the fishnet. The objective is to assess the performance of the algorithm, identifying its strengths, weaknesses, and potential areas for improvement.

The potentially different camera intrinsic matrix, image, and grid size make it hard to determine how well the FFT algorithms actually perform in comparison size the FFT images will look different.

The first thing to note is that the previous method detected a grid 40 cm away relatively accurately. With the current camera specs the peaks in the frequency transformed image will be too close together to be reliably detectable.

At 80 cm the previous FFT method was more accurate with the distance estimation. At distances further away than that, the proposed method had comparable or better results, with the exception for 40 °angle, where the baseline method still got accurate results.

A notable improvement is the angle prediction, which is achieved by the FFT method improvements explained in Section 4.3.6. Splitting the underwater image into four quadrants reduces the reliance

on the angle estimation of a single FFT, which is challenging when dealing with blurred and merged peaks in tilted images. Instead, the four-split method relies more on area estimation for each quadrant and uses the distance and y-axis rotation captured by the FFT method, which tends to be more accurate, and uses the difference in area of each section and use that for angle estimation.

The strategy employed by the previous FFT method for selecting the correct peak that constitutes the basis vectors of the grid is not discussed in detail. In contrast, the current method involves a multitude of tuning parameters, which can offer advantages but also increase complexity. It should be emphasized that each tuning parameter can be individually adjusted since they do not directly affect one another. The first set of tuning parameters focuses on filtering out undesirable FFT peaks. This involves fine-tuning the ellipse search area, determining the minimal acceptable distance between two peaks using DBSCAN, and selecting an appropriate lower threshold value for the peaks, as described in Section 4.3.3. To optimize performance, these values should be decreased when the image is in close proximity and increased when it is farther away. Subsequently, the three weights utilized for selecting the correct peaks can be fine-tuned. Consideration can also be given to dynamically adjusting these parameters to achieve improved results.

This comparison with existing methods demonstrates the algorithm's performance and highlights its unique contributions. However, further exploration of parameter settings and additional enhancements are necessary to address its limitations and improve accuracy, particularly in challenging conditions.

6.5 Tracking

The GMA optical flow method has demonstrated its effectiveness in capturing motion patterns and tracking junction points within the fishnet, as illustrated by the results of the tracking experiments. The examination of estimation error drift over time revealed reasonably accurate tracking performance, with a distance of 8 pixels observed for 99 flow estimates, highlighting the method's capability to handle significant motion while maintaining accuracy.

Furthermore, a comparison between direct optical flow estimation from the first image to the last image and tracking each frame individually revealed insightful findings. The direct estimation approach provided an estimated position closer to the ground truth compared to tracking each frame separately. The average optical flow vector for the entire image was significantly different depending on the tracking method employed. Tracking every frame resulted in an average optical flow vector of $(-162.4, 65.9)$, while the direct estimation approach produced a different average vector of $(-113.5, 62.1)$. These variations can be attributed to the diverse motion patterns captured during tracking and the summed drift of the optical flow vectors.

The GMA optical flow method, although slower, has the potential to estimate large displacements accurately, making it suitable for capturing significant motion between frames. On the other hand, other optical flow methods that estimate smaller motion at each step may benefit from performing optical flow on each frame individually. This approach could provide better accuracy overall, albeit at the cost of speed compared to the direct estimation approach.

To address the slight drift in junction point estimation, a local matching algorithm could be utilized to estimate the closest junction points and obtain the best match. Fast clustering algorithms can efficiently find the nearest junction points within a given area, improving the accuracy of the tracking results.

Additionally, in scenarios where there are significant environmental changes or complex motion patterns, employing optical flow estimation on all images may yield better results than directly estimating the flow from the first to the last image. This is because the larger number of flow estimates can capture more fine-grained motion details and account for variations caused by environmental factors.

To further evaluate the performance, an additional test was conducted to track back to the initial

point using extra images. However, a discrepancy was observed as the estimated ending pixel was $(278, 45)$ instead of the starting pixel at $(287, 55)$. This discrepancy highlights the challenges encountered when accurately tracking points over extended periods, particularly in complex underwater environments.

However, the discussion of the tracking experiments brings forth several considerations for further improvement. The use of dense deep learning methods, like GMA, introduces computational challenges due to their inherent slowness. Since only the junction points of the fishnet are required for tracking, a sparse deep learning method could be more efficient in this context. Alternatively, exploring faster optical flow methods, currently unavailable due to their non-commercial nature, could offer valuable insights for optimizing and developing a new optical flow approach.

Although the GMA optical flow method exhibited impressive performance without specific training on the dataset used in this study, employing a more specialized method trained on the specific dataset could potentially achieve even better results while requiring fewer parameters for the deep learning model. Additionally, it is worth noting that the optical flow estimation was performed solely on the CPU, limiting its speed. Exploring optimization techniques to leverage both the CPU and GPU could significantly enhance the computational efficiency of the method. The average computation time for each optical flow estimate was approximately 3 seconds, indicating the potential for improvement through hardware acceleration and GPU utilization.

Moreover, the optical flow estimate obtained from the tracking experiments opens up possibilities for estimating changes in the fishnet's pose. By integrating the optical flow estimate with the pose estimation obtained through the FFT method and junction point method, a more comprehensive understanding of the fishnet's relative motion characteristics can be achieved.

A notable characteristic worth mentioning is the similarity of optical flow vectors among neighboring pixels. This characteristic can be exploited to estimate optical flow on images with reduced pixel density or lower resolution, effectively reducing the computational cost associated with dense optical flow estimation. Careful selection of an appropriate resolution for optical flow estimation allows for a balance between computational efficiency and tracking performance.

In summary, the GMA optical flow method demonstrates robust tracking capabilities for junction points in the fishnet, highlighting its effectiveness in capturing motion patterns and facilitating subsequent analysis and applications in underwater environments. However, there are several avenues for improvement. Exploring alternative sparse deep learning methods, optimizing the optical flow estimation process, integrating pose estimation with optical flow analysis, and leveraging the correlation between neighboring pixels can lead to faster and more accurate tracking results. Additionally, training a specialized optical flow model for the specific dataset and exploring hardware acceleration techniques are important considerations to address the current limitations in computational efficiency. The choice of optical flow method should be guided by specific application requirements and environmental conditions. Further advancements are necessary to address cumulative error and enhance accuracy over extended tracking durations.

Chapter 7

Conclusion

This master thesis aimed to develop a robust algorithm for fishnet detection and tracking using an AUV. The proposed algorithm addressed the limitations of existing approaches and successfully tackled the challenges posed by underwater environments. It achieved accurate and real-time detection of fishnets, estimation of fishnet pose, and tracking of junction points.

The algorithm's performance was evaluated through comprehensive experiments using real-world underwater data from fish farming sites. It demonstrated promising results in various environmental conditions, including scenarios with occlusions of fish, debris and marine growth on the fishnets.

The peak filtering and value tuning processes enhanced the accuracy of peak detection, and the algorithm effectively selected the correct basis vectors from the FFT images. The algorithm showed good performance in detecting fishnets within a range of 0.9 meters to 4 meters, with potential for detection up to 5 meters in real scenarios.

The junction point detection algorithm successfully identified junction points in fishnets, although its accuracy varied depending on factors such as fishnet cell size and occlusions. Further refinements and optimizations are recommended to improve its performance, specially for multi-layered fishnets and considering an adaptive approach for selecting the clustering distance value based on the distance to the net.

The developed algorithm has significant implications for underwater robotics, particularly in fish farming operations. It can be integrated into existing AUV systems to enable autonomous fishnet detection and tracking, enhancing the AUV's navigational capabilities and facilitating decision-making and information gathering. The algorithm's performance evaluation and the insights gained from real-world scenarios provide valuable feedback for further improvements and optimizations.

Overall, this master thesis has contributed to enhancing AUV capabilities in fish farming operations, contributing to the sustainability and efficiency of the aquaculture industry. The research opens doors for future advancements in underwater robotics and makes a valuable contribution to the broader field of marine technology.

Bibliography

[1] Food and A. O. of the United Nations, *The State of World Fisheries and Aquaculture 2022: Towards Blue Transformation*. Rome: FAO, 2022. [Online]. Available: https://doi.org/10.4060/cc0461en.

[2] H. M. Føre and T. Thorvaldsen, 'Causal analysis of escape of atlantic salmon and rainbow trout from norwegian fish farms during 2010–2018', *Aquaculture*, vol. 532, p. 736 002, 2021, ISSN: 0044-8486. DOI: https://doi.org/10.1016/j.aquaculture.2020.736002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0044848620315684.

[3] H. V. Bjelland, M. Føre, P. Lader *et al.*, 'Exposed aquaculture in norway', in *OCEANS 2015 - MTS/IEEE Washington*, 2015, pp. 1–10. DOI: 10.23919/OCEANS.2015.7404486.

[4] M. Føre, K. Frank, T. Norton *et al.*, 'Precision fish farming: A new framework to improve production in aquaculture', *Biosystems Engineering*, vol. 173, pp. 176–193, 2018, Advances in the Engineering of Sensor-based Monitoring and Management Systems for Precision Livestock Farming, ISSN: 1537-5110. DOI: https://doi.org/10.1016/j.biosystemseng.2017.10.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1537511017304488.

[5] V. Fernandes, A. Neto and D. Rodrigues, 'Pipeline inspection with auv', Jul. 2015. DOI: 10.1109/RIOAcoustics.2015.7473607.

[6] M. Saghafi and R. Lavimi, 'Optimal design of nose and tail of an autonomous underwater vehicle hull to reduce drag force using numerical simulation', *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, vol. 234, no. 1, pp. 76–88, 2020. DOI: 10.1177/1475090219863191. eprint: https://doi.org/10.1177/1475090219863191. [Online]. Available: https://doi.org/10.1177/1475090219863191.

[7] J. S. Willners, I. Carlucho, T. Łuczyński *et al.*, *From market-ready rovs to low-cost auvs*, 2021. DOI: 10.48550/ARXIV.2108.05792. [Online]. Available: https://arxiv.org/abs/2108.05792.

[8] C. Schellewald, A. Stahl and E. Kelasidi, 'Vision-based pose estimation for autonomous operations in aquacultural fish farms**this work was financed by the research council of norway through the project: Development of technology for autonomous, bio-interactive and high-quality data acquisition from aquaculture net cages (cagereporter, project number 269087).', *IFAC-PapersOnLine*, vol. 54, no. 16, pp. 438–443, 2021, 13th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2021, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.10.128. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896321015305.

[9] A. Duda, J. Schwendner, A. Stahl and P. Rundtop, 'Visual pose estimation for autonomous inspection of fish pens', in *OCEANS 2015 - Genova*, 2015, pp. 1–6. DOI: 10.1109/OCEANS-Genova.2015.7271392.

[10] J. Qin, M. Li, D. Li, J. Zhong and K. Yang, 'A survey on visual navigation and positioning for autonomous uuvs', *Remote Sensing*, vol. 14, p. 3794, Aug. 2022. DOI: 10.3390/rs14153794.

[11] L. Paull, S. Saeedi, M. Seto and H. Li, 'Auv navigation and localization: A review', *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2014. DOI: 10.1109/JOE.2013.2278891.

[12] R. Szeliski. 'Computer vision algorithms and applications'. (2011), [Online]. Available: http://dx.doi.org/10.1007/978-1-84882-935-0.

[13] K. Simek, *Dissecting the camera matrix, part 3: The intrinsic matrix*, https://ksimek.github.io/2013/08/13/intrinsic/, (Accessed on 04/28/2023), Aug. 2013.

[14] J. W. Woods, *Multidimensional Signal, Image, and Video Processing and Coding*, 1st. Burlington, MA, USA: Academic Press, 2006, ISBN: 9780120885169.

[15] S. Gruppetta, *2d fourier transform in python: Create any image using only sine functions*, https://thepythoncodingbook.com/2021/08/30/2d-fourier-transform-in-python-and-fourier-synthesis-of-images/, (Accessed on 04/14/2023), Aug. 2021.

[16] D. Fortun, P. Bouthemy and C. Kervrann, 'Optical flow modeling and computation: A survey', *Computer Vision and Image Understanding*, vol. 134, pp. 1–21, 2015, Image Understanding for Real-world Distributed Video Networks, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2015.02.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314215000429.

[17] S. T. Haider and X. Xiang, 'Traditional and modern strategies for optical flow: An investigation', *SN Applied Sciences*, vol. 3, 2021, Image Understanding for Real-world Distributed Video Networks, ISSN: 1077-3142. DOI: https://doi.org/10.1007/s42452-021-04227-x. [Online]. Available: https://link.springer.com/article/10.1007/s42452-021-04227-x.

[18] B. K. Horn and B. G. Schunck, 'Determining optical flow', *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981, ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(81)90024-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370281900242.

[19] B. D. Lucas and T. Kanade, 'An iterative image registration technique with an application to stereo vision', in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, ser. IJCAI'81, (Accessed on 05/10/2023), Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981. [Online]. Available: http://dl.acm.org/citation.cfm?id=1623264.1623280.

[20] J.-Y. Bouguet, *Pyramidal implementation of the lucas kanade feature tracker description of the algorithm*, (Accessed on 05/10/2023), 2000. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_tracking.pdf.

[21] J. Shi and Tomasi, 'Good features to track', in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[22] G. Farnebäck, 'Two-frame motion estimation based on polynomial expansion', in *Image Analysis*, J. Bigun and T. Gustavsson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370, ISBN: 978-3-540-45103-7. DOI: https://doi.org/10.1007/3-540-45103-X_50.

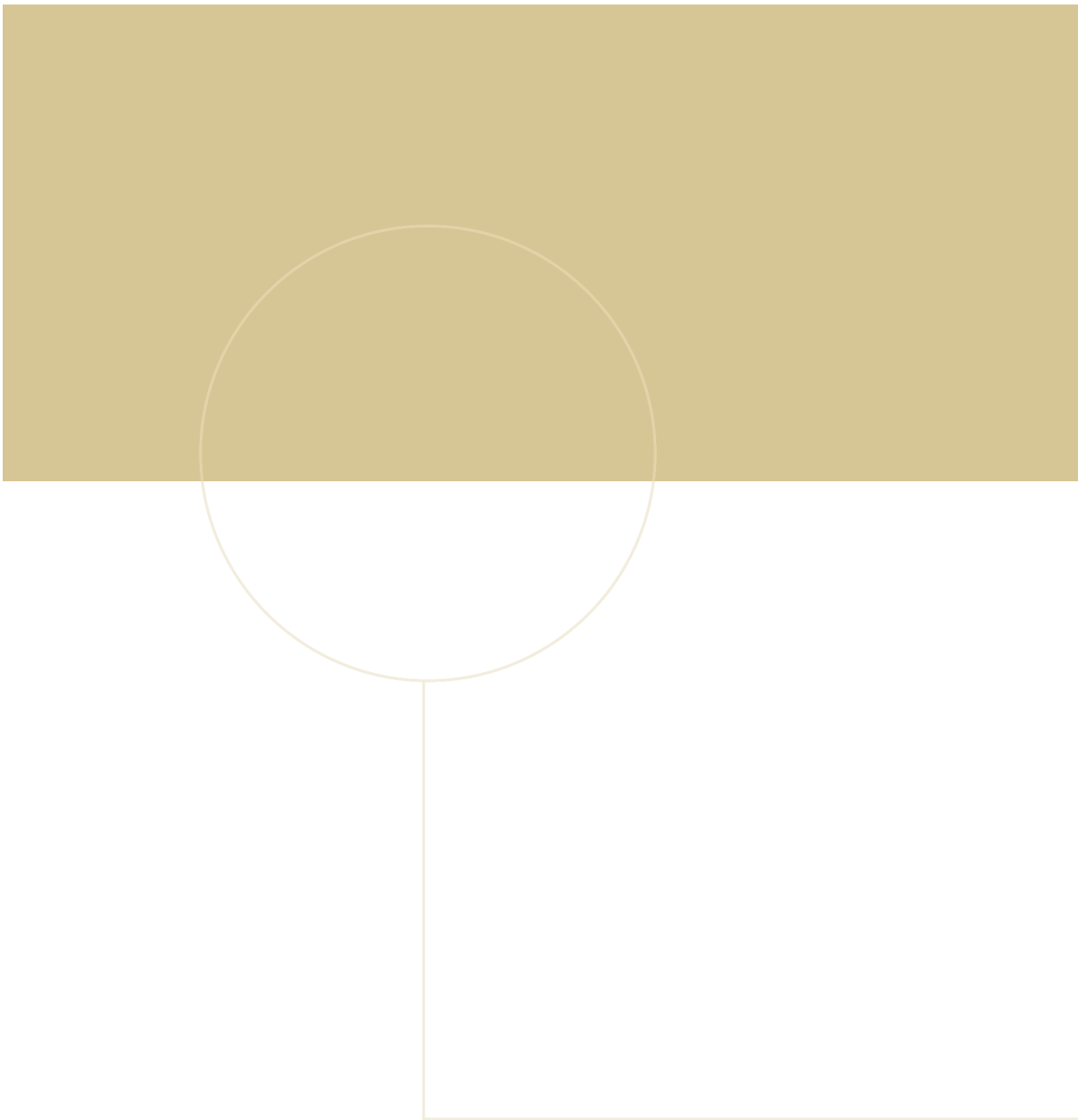
[23] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, *Robotics: Modelling, Planning and Control* (Advanced Textbooks in Control and Signal Processing), 1st ed. Springer London, 2009, pp. XXIV, 632, ISBN: 978-1-84628-641-4. DOI: 10.1007/978-1-84628-642-1. [Online]. Available: https://doi.org/10.1007/978-1-84628-642-1.

[24] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control 2nd Edition*. Wiley-Blackwell, 2021.

[25] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, 'A density-based algorithm for discovering clusters in large spatial databases with noise', in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[26] V. Klema and A. Laub, 'The singular value decomposition: Its computation and some applications', *IEEE Transactions on Automatic Control*, vol. 25, no. 2, pp. 164–176, 1980. DOI: 10.1109/TAC.1980.1102314.

[27] S. Jiang, D. Campbell, Y. Lu, H. Li and R. Hartley, *Learning to estimate hidden motions with global motion aggregation*, 2021. arXiv: 2104.02409 [cs.CV].



NTNU
Norwegian University of
Science and Technology

mohn
technology