UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA


# Vineyards monitoring using convolutional neural networks and multispectral images

Rodrigo Manso Teixeira Basílio Ferreira

**Mestrado em Engenharia Informática**

Dissertação orientada por:
Nuno Ricardo da Cruz Garcia
João Pedro Leal Abalada de Matos Carvalho

2024

# Agradecimentos

The development of this thesis was only possible with the unconditional support of several people. Therefore, I would like to express my gratitude to all those who, in some way, contributed to the completion of this project.

My big thank you to Professor Nuno Cruz Garcia for the knowledge shared, the technical guidance, and the valuable advice that guided me through each step of this work. His dedication and availability were fundamental to the conclusion of this thesis. Also to Professor João Matos Carvalho, who provided the data used in this project.

Furthermore, to the University, for the resources and academic environment provided, and to all the professors and staff who, directly or indirectly, contributed to my education.

Finally, without any doubt, this thesis would not have been possible without the total support of my family, especially my parents, brother, uncles and cousins, as well as Inês and her family, for their unconditional support, patience, and encouragement throughout this academic journey, always providing me the necessary support to achieve my goals. I would also like to thank my friends who, in various ways, contributed to my personal and professional growth, offering enriching discussions and moments of relaxation that made this path lighter.

To all of you, my sincere thanks.

*Dedicatória.*

# Resumo

Este trabalho aborda o desenvolvimento de um sistema de monitoramento agrícola voltado para a deteção de videiras e linhas de plantio utilizando imagens aéreas capturadas por veículos aéreos não tripulados (UAVs) e algoritmos de segmentação semântica. O objetivo principal é aplicar técnicas de Agricultura de Precisão para melhorar a gestão dos vinhedos, otimizando o uso de recursos como água e fertilizantes, promovendo a sustentabilidade e contribuindo para o aumento da produtividade. A pesquisa explora as vantagens oferecidas pela tecnologia de monitoramento automatizado, desde a economia de recursos até o aumento da eficiência operacional e a redução de custos. A Agricultura de Precisão tem ganhado destaque devido à sua capacidade de transformar a gestão agrícola por meio da coleta e análise detalhada de dados sobre as plantações. A viticultura, sendo uma atividade agrícola tradicional e amplamente praticada, enfrenta desafios contínuos, como a variabilidade nas condições do solo, a deteção precoce de doenças e o monitoramento do vigor das plantas. A tecnologia de UAVs, juntamente com técnicas de visão computacional e aprendizado de máquina, permite uma abordagem mais precisa e eficaz para o monitoramento de vinhedos, especialmente em áreas de difícil acesso, onde a inspeção manual seria demorada e custosa. A motivação para este trabalho surge do crescente interesse em combinar as tradições da viticultura com as tecnologias mais recentes da Agricultura de Precisão. A capacidade de detetar videiras de forma automática oferece uma série de vantagens para os produtores de vinho. A primeira vantagem é a economia de recursos, já que a deteção precisa das plantas permite que os agricultores direcionem melhor o uso de água e fertilizantes, reduzindo o desperdício. Em segundo lugar, o monitoramento constante das plantas possibilita a gestão estratégica dos vinhedos, otimizando práticas como poda, irrigação e fertilização, resultando em maior eficiência e menores custos operacionais, além de promover uma viticultura mais sustentável, com menor impacto ambiental.

Os principais objetivos deste projeto incluem o desenvolvimento de um modelo de segmentação binária para detetar videiras a partir de imagens aéreas e a implementação de um sistema para detetar as linhas de plantio usando a Transformada de Hough. O sistema foi projetado para ser simples de usar e escalável, permitindo que os agricultores monitorem seus vinhedos a partir de qualquer local que disponha da tecnologia de UAV. A estrutura do sistema baseia-se na segmentação semântica, especificamente utilizando a arquitetura U-Net, reconhecida pelo seu desempenho em tarefas de segmentação de imagens. O modelo foi treinado com dois conjuntos de dados principais. O primeiro foi obtido a partir de um estudo anterior sobre segmentação multiespectral de vinhedos e o segundo, fornecido pela empresa Beyond Vision, especializada em captura de imagens

com drones. Esses conjuntos de dados incluíam imagens orto mosaicas de alta resolução, essenciais para garantir a precisão das previsões do modelo. As orto mosaicas são imagens de grande escala, georreferenciadas, obtidas pela combinação de várias imagens aéreas capturadas em diferentes ângulos e alturas. Um dos desafios enfrentados durante o desenvolvimento do projeto foi a obtenção de conjuntos de dados adequados. A segmentação semântica exige dados que já tenham máscaras definidas, ou seja, os pixels da imagem precisam estar rotulados, distinguindo videiras de áreas sem vegetação. Dado o custo e a dificuldade de rotular manualmente grandes quantidades de dados, a utilização de conjuntos de dados disponíveis com essas máscaras foi fundamental para o treinamento do modelo. A metodologia deste trabalho segue várias etapas: o pré-processamento dos dados, a construção e o treinamento do modelo de segmentação, a aplicação da Transformada de Hough para a deteção das linhas de plantio e, por fim, a validação dos resultados. No pré-processamento, as imagens orto mosaicas foram divididas em subimagens menores de 240x240 pixels, o que facilitou o treinamento do modelo e reduziu a carga computacional. Essa abordagem modular permitiu um processamento mais eficiente e rápido das imagens, essencial para a análise de grandes áreas de vinhedos. O modelo U-Net, utilizado para segmentação, opera em uma arquitetura de codificação e decodificação, onde a imagem é inicialmente reduzida a uma representação comprimida e, em seguida, ampliada novamente para a resolução original. As conexões de salto entre as camadas do modelo permitem a combinação de detalhes em diferentes níveis da rede, o que aumenta a precisão das previsões, especialmente em áreas de bordas e detalhes finos. O modelo foi treinado com um conjunto de dados específico e avaliado com base em métricas como precisão (accuracy), interseção sobre união (IoU), coeficiente Dice e recall.

Após o treinamento, os resultados iniciais indicaram que o modelo foi capaz de detetar videiras com precisão razoável. No entanto, alguns desafios relacionados à consistência espacial entre as subimagens foram observados. Isso significa que, em certas situações, o modelo não conseguiu alinhar perfeitamente as previsões com as máscaras reais, resultando em pixels incorretamente classificados. Esse problema foi abordado com o uso do algoritmo de Otsu, que ajusta automaticamente os valores dos pixels, garantindo que a segmentação fosse binária (com valores de 0 e 255, representando a ausência e a presença de videiras, respetivamente). Essa correção permitiu uma comparação mais precisa entre as previsões do modelo e as máscaras reais. Um dos aspetos inovadores deste trabalho foi a aplicação da Transformada de Hough para a deteção das linhas cultivo das videiras. A Transformada de Hough é um algoritmo amplamente utilizado para a deteção de formas geométricas, como linhas e círculos, em imagens. No contexto dos vinhedos, foi aplicada para identificar as fileiras de plantas, algo fundamental para a automação de operações como poda e colheita. A combinação da segmentação semântica com a deteção de linhas proporciona uma solução robusta para o monitoramento automatizado dos vinhedos, permitindo uma gestão mais precisa e eficiente. Os resultados obtidos mostraram que o modelo conseguiu segmentar com sucesso as áreas de videiras nas imagens testadas, com a deteção das linhas de plantio sendo satisfatória em grande parte dos casos. As métricas de avaliação indicaram que, embora a precisão global fosse alta, havia margem para melhorias, especialmente em termos de interseção

sobre união (IoU) e coeficiente Dice, particularmente em áreas com menor densidade de videiras. A análise dos resultados revelou que variações nas condições das imagens, como iluminação e alturas de voo dos UAVs, impactaram o desempenho do modelo.

Este trabalho contribui significativamente para o campo emergente da Agricultura de Precisão, demonstrando o potencial das tecnologias avançadas, como UAVs e aprendizado profundo, para otimizar a produção agrícola. O sistema desenvolvido oferece uma abordagem prática para o monitoramento de vinhedos, promovendo uma gestão mais eficiente e sustentável dos recursos. Além disso, o código-fonte do modelo foi disponibilizado publicamente, permitindo que outros pesquisadores e profissionais da área de Agricultura de Precisão possam utilizar e aprimorar o sistema.

Para trabalhos futuros, recomenda-se a ampliação dos conjuntos de dados de treinamento, incorporando uma maior variedade de condições de solo, clima e iluminação, a fim de melhorar a robustez do modelo e sua capacidade de generalização. Além disso, o uso de técnicas de pós-processamento pode melhorar a consistência espacial das previsões, reduzindo o número de falsos positivos e negativos. Finalmente, a integração de sensores adicionais, como câmaras multiespectrais, poderá aumentar a precisão da deteção e permitir a análise de fatores adicionais, como o estado de saúde das plantas. O código e os datasets estão em `https://github.com/rodrigo-99ferreira/Vineyards`

**Palavras-chave:** Agricultura de precisão; Deteção de vinhas; Segmentação binária; Arquitetura U-Net; Deteção de linhas de cultivo

# Abstract

This study presents the development of an agricultural monitoring system designed to detect vineyards and crop lines through the application of binary segmentation techniques. The primary objective is to enhance the efficiency of vineyard monitoring, enabling precise plant detection using aerial imagery captured by unmanned aerial vehicles (UAVs). The system utilizes U-Net architecture for semantic segmentation, which was selected for its ability to effectively differentiate between vine and non-vine areas, promoting resource optimization and sustainable viticulture. Additionally, an algorithm based on the Hough Transform was implemented to accurately detect vineyard crop rows, further supporting precision agriculture practices. The model was trained and validated using datasets obtained from various sources, including publicly available datasets and those provided by industry partners. Evaluation metrics such as accuracy, Intersection over Union (IoU), and Dice Coefficient were employed to assess model performance, with results indicating varying levels of success across different datasets. The research contributes to the growing field of precision agriculture by offering a practical tool for vineyard management, with potential applications in resource allocation, environmental sustainability, and operational efficiency. The system's design and the methodologies employed underscore the feasibility of integrating advanced machine learning models into real-world agricultural contexts.The code and dataset are publicly https://github.com/rodrigo-99ferreira/Vineyards

**Keywords:** Precision Agriculture; Vineyard Detection; Binary Segmentation; U-Net Architecture; Crop Line Detection;

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

As part of the second year of the Master's in Computer Engineering at the Faculty of Sciences of the University of Lisbon, the proposed project involves the development of an agricultural monitoring system. This system will be capable of accurately detecting vineyards and crop lines within the plantation, thus improving the effectiveness of vineyard monitoring.

In this chapter, the motivation for this work is presented in Section 1.1. Then, the objectives defined for this project are discussed in Section 1.2. Finally, the main contributions of the work presented here are outlined in Sections 1.3 and 1.4, along with the structure of the document.

## 1.1 Motivation

Agriculture, regarded as the pillar of human civilization, has undergone exponential evolution over the centuries. Unprecedented challenges confront us today, encompassing the escalating demand for food and the urgent need to conserve natural resources while mitigating adverse environmental impacts. In this context, Precision Agriculture emerges as a promising approach, reshaping agricultural paradigms through the integration of advanced technology and precise data to optimize every aspect of the production cycle.

Viticulture, acknowledged as one of the oldest and most revered agricultural activities, assumes a vital role in the culture and economy of various regions worldwide. Nonetheless, the inherent complexity in managing vineyards and producing high-quality grapes remains a perpetual concern for modern winegrowers. In this context, research into Vineyard Monitoring and Detection has evolved into a constantly progressing field of study, combining the technological advances of Precision Agriculture with the specific needs of contemporary viticulture.

The motivation behind this research arises from the belief that the strategic application of monitoring and detection technology has the potential to revolutionize the management of vineyards. Vineyards, often situated in complex and diverse landscapes, present distinctive challenges, ranging from managing plant vigor to early disease identification, harvest optimization, and plant detection. It is the convergence between the rich tradition of winemaking and the cutting edge of agricultural technology that propels this work forward.

The utilization of Precision Agriculture technologies for plant detection in winemaking presents

several noteworthy benefits for winegrowers. These advantages constitute part of the motivation behind this work:

- Saving resources - Through plant detection, winegrowers can find out how their vines are doing and thus direct their resources, such as water and fertilizers, more effectively. This reduces waste and the associated costs.

- Optimization of Vineyard Management - With information on plant distribution of vigour throughout the vineyard, growers can adjust pruning, irrigation and fertilization to optimize vine growth and development.

- Increased Sustainability - By reducing the use of inputs and promoting more precise management practices, plant detection contributes to more sustainable viticulture, minimizing the environmental impact of wine production.

- Strategic vineyard management - The information obtained through plant detection can be used to make long-term strategic decisions, such as choosing the most suitable grape varieties for certain areas or optimizing the layout of vineyards.

- Improved Operational Efficiency - By avoiding unplanned interruptions, plant detection contributes to a more efficient agricultural operation. This means that farmers can carry out their activities more productively and meet planting and harvesting deadlines.

- Reduced labor costs - It also saves money because it doesn't require detection by people but by machines. This saves money and makes plant detection monitoring more effective and more perceptive.

## 1.2 Goals

The objective of this project is to develop an application that demonstrates the ability to identify plants through the use of binary segmentation and detection the rows crops. Additionally, the goal is to create an easy-to-use application for monitoring vineyards using aerial images captured by UAVs.

This approach is aimed at strengthening precision agriculture, aiming for greater efficiency and effectiveness in various agricultural contexts. By achieving these objectives, the project seeks to contribute significantly to the advancement and optimization of agricultural practices, promoting a positive impact on the sector's productivity and sustainability.

## 1.3 Contributions

This work resulted in the following contributions described below:

- Development of a system capable of detecting plants, specifically vineyards. In addition to this detection through binary segmentation, crop lines were also detected using the Hough Transform algorithm.
- The developed model represents a valuable contribution, as once validated, it can serve as a starting tool for future work or be applied in real-world contexts. The code will be made available in the following repository https://github.com/rodrigo-99ferreira/Vineyards
- To carry out this work, datasets were necessary to validate the developed model. These datasets were provided by [1], and they remain available for future research related to this topic.
- This project contributes to the growing body of work supporting the use of technology in agriculture. It has also led to the creation of a paper that will be submitted to contribute to the advancement of precision agriculture.
- Additionally, this work will aid in the development of precision agriculture, offering ease and savings for farmers through its practical application.

## 1.4 Structure of the document

This document is organised as follows:

- Chapter 1 - Introduction: This chapter presents the motivation behind the development of this work, as well as the objectives outlined to be achieved and the general contributions of the project.
- Chapter 2 - Related Work: In this chapter, related work that contributed to the construction of this thesis will be presented. An introduction to the use of drones in agriculture will be provided, followed by works on semantic segmentation using U-Net. Additionally, studies employing the Hough Transform for crop line detection in agriculture will be discussed. Finally, the evaluation metrics used for the created model will be outlined.
- Chapter 3 - Methodology: The methodology will present the development process of the work. It will cover what was used and how it was implemented, including pre-processing, segmentation, and line detection
- Chapter 4 - Results: The results will be presented and discussed. Comparisons between evaluation datasets will be made, and the remaining datasets will be showcased. A thorough analysis of the work will be conducted, along with a detailed discussion of the results obtained.
- Chapter 5 - Conclusion: This chapter will provide a final summary of the work, along with suggestions and guidelines for future iterations of the system.

# Chapter 2

# Related Work

In this chapter, the main concepts used in this work, which have been similarly employed in other studies, will be presented.

In Section 2.1, the use of UAVs in agriculture will be discussed, representing a significant contribution to precision agriculture. The capture of datasets used in this work is made possible through the use of UAVs. Although not part of this work, it was carried out by others to make datasets available for conducting this study.

In Section 2.2, the objective of this project will be presented, along with the works that contributed to its development and the methodology applied.

Next, Section 2.3 will address line detection using the Hough Transform in agriculture.

Finally, in this chapter, Section 2.4 will explain the evaluation metrics for the model developed in this project.

## 2.1 Remote Sensing - UAV

Before any data analysis and interpretation process, the initial and crucial stage is the collection of information. In the application of vine detection using data processing, this stage takes on even greater importance. The accuracy and comprehensiveness of the data acquired plays a fundamental role in the system's effectiveness. It is therefore essential to employ advanced collection methods that make it possible to obtain detailed information about the vine crop. This can include the use of drones equipped with high-resolution cameras, allowing superior quality images to be captured, even in areas that are difficult to access. In addition, the use of specific sensors to detect key parameters, such as soil moisture levels and sunlight incidence, complements data collection in a comprehensive way. Integrating these technologies into the data collection phase not only improves the accuracy of the information, but also optimizes the vine detection process, providing a solid basis for further analysis.

Based on this well-collected data, the vine detection system will have a robust foundation on which to operate. From here, advanced image processing and machine learning techniques will come into play to identify and map the vines accurately and effectively. This approach not only optimizes the monitoring process, but also has the potential to revolutionize agricultural practice,

enabling precise and punctual interventions, such as the targeted application of nutrients or the identification of areas that require special attention. In this way, data collection proves to be the solid foundation on which to build the effectiveness of the vine detection system through advanced processing, promoting not only efficiency but also sustainability in modern agricultural practices.

Although this work does not focus on data collection, obtaining suitable datasets was the first barrier encountered. This is due to the rarity of available datasets that already include the necessary masks. The inclusion of these masks was mandatory, as the primary goal was to build and train a model using existing datasets and subsequently verify the robustness of this model on different datasets.

After an extensive search, three appropriate datasets were identified. Of these, only two were selected for use, sourced from the study "Multispectral Vineyard Segmentation: A Deep Learning Comparison Study"[5]. The remaining datasets used for validation were provided by the company Beyond Vision[1].

Beyond Vision[1] is a Portuguese company specializing in the application of drone technology across various industrial sectors. Founded with the aim of offering innovative solutions, Beyond Vision uses drones equipped with high-resolution cameras and advanced sensors. These drones are employed to provide services that optimize processes, increase efficiency, and enhance safety in various operations.

The datasets provided by Beyond Vision played a crucial role in advancing this work. The quality and precision of the data, obtained through advanced drone technologies, allowed for more detailed analysis and validation of the developed model. Additionally, the partnership with Beyond Vision exemplifies the importance of collaboration between academia and industry for technological progress and practical innovation.

Thus, overcoming the initial barrier of obtaining datasets with suitable masks was fundamental to the project's success. The use of data from reliable and advanced sources ensured the robustness and applicability of the developed models, demonstrating the potential of drone technologies in research and development.

## 2.2 Semantic Segmentation

In the previous section, the starting point of this project in data collection was presented. Although it was not part of this project, it is indispensable because, without data, the realization of this project would not have been possible. This section presents the works that helped shape and support the purpose of this project. The work that contributed the most was "Multispectral Vineyard Segmentation: A Deep Learning Comparison Study"[5] , as it provided training datasets for our model and guidelines for our objectives.

This work was essential for our research, as it offers a comprehensive comparison of different deep learning approaches applied to vineyard segmentation using vineyard datasets. The main objective of the study was to evaluate the effectiveness of various deep learning techniques in the segmentation of vineyards in multispectral images. Accurate segmentation is crucial for ap-

plications such as plant health monitoring, irrigation management, and selective harvesting. The research compared several convolutional neural network (CNN) architectures, including U-Net, SegNet, and FCN, to determine which offers the best performance in terms of accuracy, speed, and robustness.

Multispectral segmentation is an advanced technique that uses different wavelengths of light to obtain detailed information about vegetation. In the context of vineyards, this technique allows for precise analysis of plant conditions, facilitating the detection of problems such as diseases, nutritional deficiencies, and water stress.

The methodology used encompasses the following points:

- Data Collection and Preprocessing: Multispectral images captured from vineyards using advanced sensors installed on drones were used. These images underwent a rigorous preprocessing process, including normalization to correct luminosity variations and noise removal to improve data quality. Additionally, the images were spatially aligned to ensure segmentation accuracy.

- Model Development and Training: Three different convolutional neural network architectures were developed and trained with the preprocessed data:

    1. U-Net: This architecture is widely known for its ability to achieve precise segmentation in biomedical images. In the study, it was adapted for vineyard segmentation, leveraging its U-shaped structure that facilitates the combination of features at different scales.
    2. SegNet: With an encoder-decoder architecture, SegNet was evaluated for its efficiency in semantic segmentation. SegNet's ability to preserve high-resolution information during the decoding process was a key point in the analysis.
    3. FCN (Fully Convolutional Networks): Evaluated for its ability to perform dense segmentation in high-resolution images, FCN proved promising in the analysis of large vineyard areas.

- Performance Evaluation and Comparison: The models were evaluated using a comprehensive set of performance metrics. The primary metrics included Accuracy, Precision and Recall, F1-Score, Processing Time, and Robustness. Some of these metrics will be used in the evaluation of the model developed in this project.

The study results revealed valuable insights into the performance of different convolutional neural network architectures. The detailed analysis demonstrated that U-Net was the architecture that showed the best overall performance in terms of accuracy and robustness. U-Net was able to segment vineyards with high accuracy, proving particularly effective in identifying fine details in the images. SegNet stood out for its efficiency in terms of processing time. SegNet was able to segment images quickly, which is a significant advantage in real-time applications. Although FCN showed good results, it fell short compared to the other two architectures under the specific

conditions of the multispectral vineyard data. The main limitation observed was lower accuracy in segmenting complex areas of the vineyards.

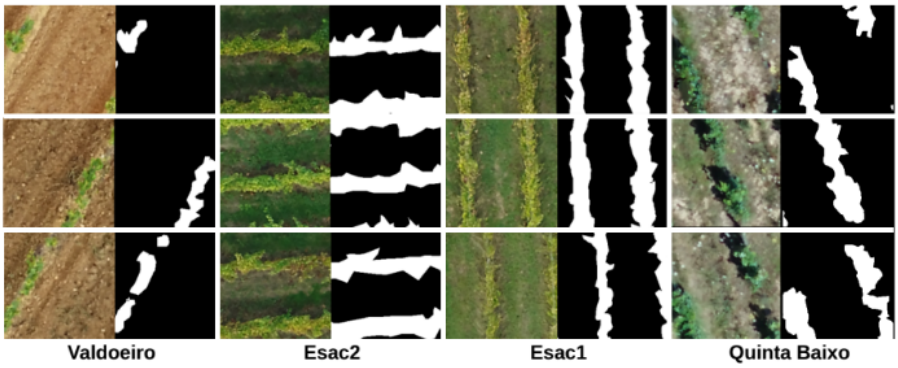



Figure 2.1: Segmentation in ”Multispectral Vineyard Segmentation: A Deep Learning Comparison Study”[5]

The comparative analysis provided a comprehensive view of the advantages and disadvantages of each approach. It highlighted the importance of choosing the appropriate network architecture based on the specific application requirements, such as the need for detailed accuracy versus real-time processing demands. Therefore, based on the results of this project, the U-Net architecture was chosen for segmentation in the developed project. This choice was motivated by U-Net's high accuracy and robustness, essential characteristics for the specific needs of vineyard monitoring.

The conclusions obtained in the study ”Multispectral Vineyard Segmentation: A Deep Learning Comparison Study”[5] were fundamental in guiding the development of this work.

According to the results of the paper[5], the U-Net architecture demonstrated the best performance. Therefore, further research was conducted on works that used U-Net for segmentation with aerial images.

For example, the article titled ”Deep learning for identifying salt bodies in seismic images”[22] discusses the application of deep learning techniques to identify salt bodies in seismic images. The study explores the use of convolutional neural networks (CNNs), with a particular focus on architectures such as U-Net, to automate the process of detecting and segmenting salt structures. The article highlights the advantages of deep learning over traditional methods, including improvements in accuracy and efficiency. The methodology involves training the neural networks on large datasets of annotated seismic images and evaluating their performance against standard benchmarks.

This research demonstrates how modern deep learning techniques can transform seismic interpretation, offering faster and more accurate solutions for identifying salt bodies, which are critical in hydrocarbon exploration.

As observed, U-Net proves to be effective and enhances accuracy and efficiency in aerial images.

## 2.3 Line Detection

In this section, the works that contributed to the accurate and effective detection of vineyard crop rows are presented. We will look at some examples of works that used the Hough Transform algorithm for crop row detection.

The article titled ”Crop Row Detection on Tiny Plants with the Pattern Hough Transform”[21] presents an innovative approach for detecting crop rows in UAV (Unmanned Aerial Vehicle) imagery of small plants, utilizing the Hough Transform. The authors propose an adapted version of the Hough Transform specifically for early-stage crops, where the plants are small and sparse, making row detection more challenging.

The methodology developed by the authors enhances the robustness and accuracy of crop row detection by incorporating advanced pattern recognition techniques. These techniques allow for better management of variability and noise present in UAV imagery, which are common characteristics in images of plants at early growth stages.

The study includes a series of experiments demonstrating the effectiveness of the proposed method. The experimental results show that the approach based on the Hough Transform significantly outperforms traditional methods in terms of crop row detection accuracy. These results indicate that the new approach is more effective for identifying crop rows in scenarios where the plants are small and spaced apart.

The implications of this work are significant for precision agriculture. Accurate and efficient crop row detection is crucial for agricultural monitoring and management. With this new technique, it is possible to improve informed decision-making and optimize agricultural production. The developed technology can be used to enhance real-time monitoring and crop management practices, contributing to increased operational efficiency and better resource management.

Another paper, titled ”Crop-row detection algorithm based on Random Hough Transformation” [12], presents an innovative algorithm for crop row detection using the Random Hough Transformation. This method is designed to improve the accuracy and efficiency of identifying crop rows in agricultural images, especially in scenarios where the rows may be obscured or not clearly defined.

The Random Hough Transformation is a variation of the traditional Hough Transform that reduces computational complexity by randomly sampling data points instead of considering all points in an image. This process allows for faster and more effective detection of crop rows, even in images with significant noise or with crops in early growth stages, where the plants are small and spaced apart.

The authors conducted a series of experiments to test the performance of the proposed algorithm. The results show that the Random Hough Transformation is capable of detecting crop rows with high accuracy, outperforming traditional methods in terms of speed and robustness. Additionally, the algorithm proved effective under various environmental and lighting conditions, making it a versatile tool for use in different agricultural contexts.

This work highlights the potential of the Random Hough Transformation as a powerful tech-

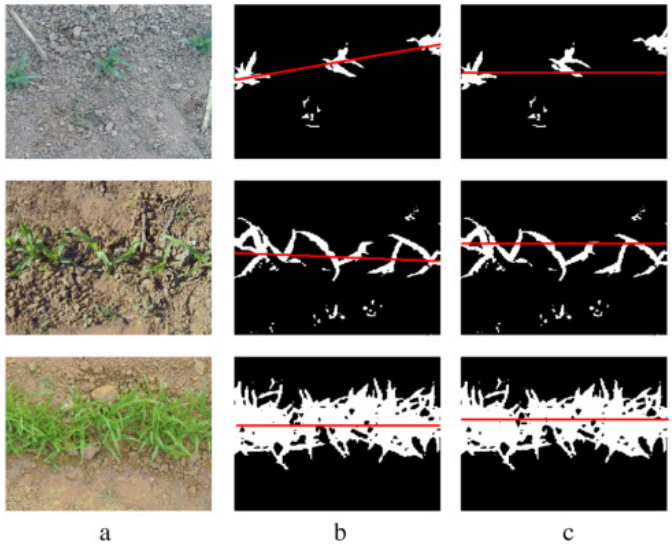



Figure 2.2: The center line of crop-row detection results by gradient-based RHT and HT in ”Crop-row detection algorithm based on Random Hough Transformation”[12]

nique for precision agriculture, facilitating crop monitoring and management. The implementation of this algorithm can help farmers optimize agricultural practices, improve operational efficiency, and increase crop productivity.

Therefore, it can be concluded that the Hough Transform is a highly capable and effective algorithm for crop row detection.

## 2.4 Evaluation Metrics

In the field of computer vision, binary segmentation is an essential technique that allows the differentiation between two distinct classes of pixels in an image, usually representing the background and the object of interest. To ensure the effectiveness and accuracy of segmentation models, it is crucial to use evaluation metrics that enable a rigorous analysis of their performance. These metrics provide a quantitative means of measuring how well the model segments an image, correctly identifying the pixels that belong to each class. Therefore, the careful selection of evaluation metrics is vital to ensure reliable results and the continuous improvement of binary segmentation algorithms. So in this section, the metrics that will be used to evaluate the model will be described.

### Accuracy

In binary segmentation tasks, accuracy is a measure that quantifies how correctly a model distinguishes between two classes: the foreground (the object of interest) and the background. Specifically, it is the ratio of the number of correctly classified pixels to the total number of pixels in the image. It provides an overall assessment of the model’s performance in terms of the proportion of pixels that were correctly classified.

Mathematically, the accuracy for binary segmentation is defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

Where:

- **TP** (True Positives): The number of pixels that are correctly classified as belonging to the positive class.
- **TN** (True Negative): The number of pixels correctly predicted as belonging to the background.
- **FP** (False Positives): The number of pixels incorrectly predicted as positive.
- **FN** (False Negatives): The number of pixels incorrectly predicted as negative.

While accuracy is a simple and intuitive metric, it can be misleading in cases where the dataset is imbalanced (i.e., when the number of background pixels significantly outweighs the number of foreground pixels). In such cases, a model that predicts all pixels as the majority class (e.g., all background) may still achieve a high accuracy despite failing to segment the object of interest.

This calculation is useful when evaluating the performance of image segmentation models, particularly in tasks such as medical segmentation (e.g., tumor detection) and object segmentation in computer vision, among others. Accuracy is a global metric that is easy to understand; however, it can be complemented by other metrics, which will be described in the following sections, such as IoU, Dice, precision, and recall, for a more robust evaluation.

### Precision

Precision, also known as positive predictive value, is a metric used to evaluate the performance of a binary segmentation model, particularly in terms of how accurately it identifies the pixels of the target class (e.g., the foreground or the object of interest). It measures the proportion of pixels that are correctly predicted as the target class among all the pixels predicted as the target class. It answers the question: ”Of all the pixels that the model predicted to be part of the positive class, how many of them are actually correct?”

Mathematically, precision is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

Where:

- **TP** (True Positives): The number of pixels that are correctly classified as belonging to the positive class.
- **FP** (False Positives): The number of pixels incorrectly predicted as belonging to the positive class.

Precision is particularly important in contexts where false positives are costly or problematic. A high precision ensures that most of the detected areas are truly present, minimizing unnecessary alarms or treatments.

### Recall

Recall in the context of binary segmentation measures the proportion of actual positive pixels that have been correctly predicted as positive by the model. It answers the question: ”Of all the pixels that truly belong to the positive class, how many did the model correctly identify?”

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

Where:

- **TP** (True Positives): The number of pixels correctly predicted as belonging to the positive class.
- **FN** (False Negatives): The number of pixels incorrectly predicted as negative.

Recall is important in contexts where false negatives are costly or undesirable. High recall ensures that most, if not all, of the actual disease cases are identified, even if it means accepting some false positives. Balancing recall with precision is often necessary to ensure a model is both sensitive (high recall) and specific (high precision).

### Dice Coefficient

The Dice Coefficient is a widely used metric for evaluating the performance of image segmentation models, particularly in binary segmentation tasks. It is designed to measure the similarity between two sets: the predicted segmentation (output from the model) and the ground truth (the actual, correct segmentation).

$$\text{Dice Coefficient} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{2.4}$$

Where:

- **TP** (True Positives): The number of pixels correctly predicted as belonging to the positive class.
- **FP** (False Positives): The number of pixels incorrectly predicted as positive.
- **FN** (False Negatives): The number of pixels incorrectly predicted as negative.

The Dice Coefficient measures the overlap between the predicted segmentation and the ground truth. The formula can be interpreted as twice the number of overlapping pixels (true positives) divided by the total number of pixels in both the predicted and ground truth positive sets.

The multiplication by 2 ensures that the Dice Coefficient gives equal importance to both precision and recall:

- Precision measures the proportion of correctly predicted positive pixels out of all the pixels predicted as positive.

- Recall measures the proportion of correctly predicted positive pixels out of all the actual positive pixels in the ground truth.

By balancing these two components, the Dice Coefficient provides a single, comprehensive metric for evaluating the accuracy of segmentation models.

The Dice Coefficient ranges from 0 to 1. A value of 1 indicates perfect agreement between the predicted segmentation and the ground truth — every pixel that should be labeled as positive is correctly predicted as positive, and no negative pixels are incorrectly predicted as positive. A value of 0 indicates no overlap between the predicted positive pixels and the ground truth.

The Dice Coefficient is particularly well-suited for image segmentation tasks because two aspects. First, it combines both precision (how many of the predicted positives are true positives) and recall (how many of the actual positives are detected), making it a balanced measure that penalizes both false positives and false negatives. Second, unlike metrics like accuracy, which can be misleading in the case of highly imbalanced classes (e.g., where the positive class is much smaller than the negative class), the Dice Coefficient focuses specifically on the positive class overlap.

The Dice Coefficient is a powerful and widely-used metric for binary segmentation tasks, providing a comprehensive measure of how well the predicted segmentation aligns with the ground truth. It is particularly useful in applications where both false positives and false negatives must be minimized, such as in medical imaging, autonomous driving, and environmental monitoring.

By incorporating both precision and recall into a single score, the Dice Coefficient offers a balanced evaluation that helps to assess and improve segmentation models effectively.

### Intersection over Union

The Intersection over Union (IoU) is another popular metric used for evaluating the performance of image segmentation models. It measures the overlap between the predicted segmentation (output of the model) and the ground truth (actual segmentation). The IoU is particularly useful for understanding how well the model captures the region of interest.

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{2.5}$$

Where:

- **TP** (True Positives): The number of pixels correctly predicted as belonging to the positive class.
- **FP** (False Positives): The number of pixels incorrectly predicted as belonging to the positive class.
- **FN** (False Negatives): The number of pixels incorrectly predicted as belonging to the negative class.

The IoU ranges from 0 to 1. A value of 1 indicates perfect overlap between the predicted segmentation and the ground truth. High IoU means the model's predictions closely match the ground truth, with minimal errors. A value of 0 indicates no overlap at all. Low IoU suggests that the model's predictions diverge significantly from the actual segmentation.

The IoU metric is critical for several reasons:

- Measures Overlap Quality: It directly measures the quality of the overlap between the predicted and actual segmentations, making it a useful metric for assessing how accurately the model captures the region of interest.
- Robust to Class Imbalance: Like the Dice Coefficient, IoU is less affected by class imbalance than metrics such as accuracy, since it focuses specifically on the positive class overlap.
- Commonly Used in Benchmarks: IoU is a standard metric in many segmentation challenges and benchmarks, making it an essential metric for comparing different models' performances.

The Intersection over Union (IoU) is a powerful and interpretable metric for evaluating the performance of binary segmentation models. It provides a direct measure of how well the predicted segmentation aligns with the ground truth, making it essential for assessing model performance, particularly in domains like agriculture and any other field requiring precise segmentation of objects.

## 2.5 Summary

This chapter covers the work and concepts utilized in the development of this project. Additionally, a literature review was conducted, providing a detailed exploration of various topics, including segmentation models and their architectures, as well as crop line detection.

A description of the metrics used to evaluate the developed model was also provided.

# Chapter 3

# Methodology

In this chapter, the work conducted is presented and comprehensively analyzed. The foundation of this system lies in its capability to differentiate between vine and non-vine areas based on images captured by UAVs, constituting the initial segment. The subsequent part necessitates the system's proficiency in detecting planting lines.

Section 3.1 will address the datasets employed for training and testing the model. Section 3.2 will delve into the model and algorithms employed for image segmentation from drone-captured imagery. Finally, in Section 3.3, the focus will be on the detection of lines within the plantation, followed by a summarization of Chapter 3 in Section 3.4.

## 3.1 Datasets

In this section, an overview of the datasets employed in the development of this work will be provided. Procuring these datasets entailed considerable effort and research, as the creation of the datasets themselves—specifically, the task of capturing images of the vineyards using an Unmanned Aerial Vehicle (UAV) was beyond the scope of this project. This is in alignment with the primary focus of this work, which centers around image processing.

While there is an abundance of literature on the subject, it is exceedingly uncommon to come across papers that openly share their datasets. However, in the case of the [5] paper, datasets for three vineyards were fortunately available. The noteworthy advantage of utilizing these datasets is their pre-existing masks, greatly facilitating the model training process. The challenge encountered lies in the fact that not all datasets were in ideal conditions for use, as some files had defects. Therefore, it was possible to utilize only two sets for training the model, while the third was reserved for verifications.

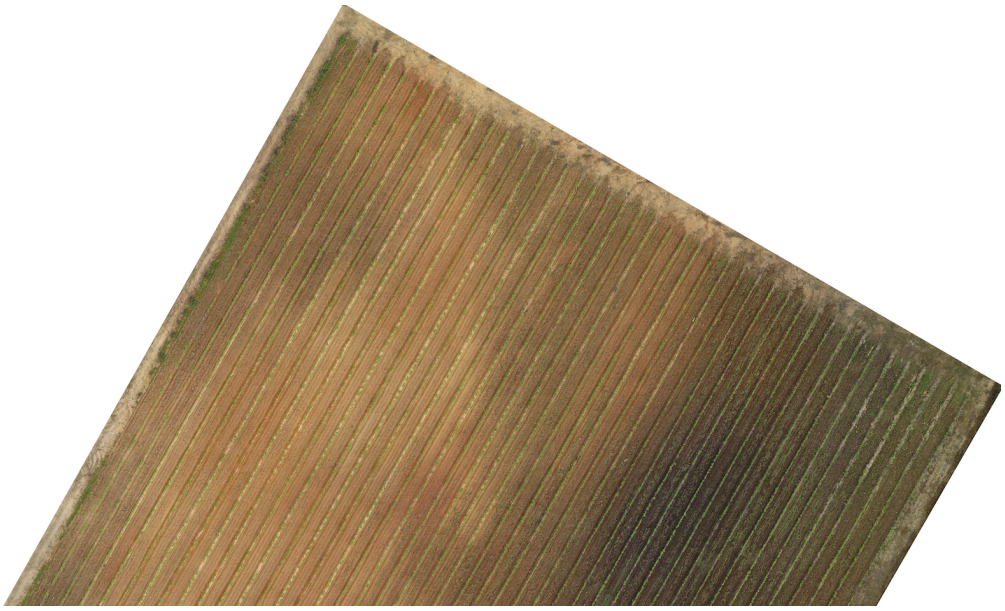


Figure 3.1: Dataset from [5]

To evaluate the system, another set of data provided by Beyond Vision[1] was utilized. This company specializes in the development of Unmanned Aerial Vehicles (UAVs) and maintains a division dedicated to image capture, encompassing data sets related to vineyards, which will be employed in this project.

Both the training and verification datasets are orthomosaics, visually representing a unified and georeferenced terrestrial area. These orthomosaics are generated by merging various aerial images, undergoing geometric rectification to correct distortions caused by variations in terrain, altitude, or camera tilt during capture. The result is a composite image that accurately reproduces the Earth's surface in terms of correct scale and position.

In this manner, Beyond Vision's datasets represent a valuable font of information, contributing significantly to the progress of this undertaking.

| Dataset | #Image Size | #Pixels | Channels | Flight Height(m) |
|---|---|---|---|---|
| Dataset 1[5] | 15956x14162 | 225968872 | RGB | 120 |
| Dataset 2[5] | 23419x8820 | 206555580 | RGB | 60 |
| Dataset 3[5] | 22400x23500 | 526400000 | RGB | 60 |
| Dataset 4[1] | 6383x13782 | 87970506 | RGB and multispectral | 60 |
| Dataset 5[1] | 6659x11156 | 74287804 | RGB and multispectral | 60 |
| Dataset 6[1] | 7788x11769 | 91656972 | RGB and multispectral | 60 |
| Dataset 7[1] | 23396x9124 | 213465104 | RGB and multispectral | 60 |
| Dataset 8[1] | 28796x11659 | 335732564 | RGB and multispectral | 60 |
| Dataset 9[1] | 6912x11435 | 79038720 | RGB and multispectral | 60 |

Table 3.1: Table with information about the datasets; Image Size: number of pixels high X number of pixels wide; Channels; Fligth Height (meters)

## 3.2 Plant Detection

In this section, we will begin to describe the work carried out and its development process. We'll start by discussing the semantic segmentation technique, which is used in plant detection. In this context, we will detail the model used, its architecture, as well as the processing and segmentation procedures. We will also explain some additional algorithms that were used to help perform the task.

In the next section, we will present the approach adopted for detecting the lines within the plantation.

### 3.2.1 Semantic Segmentation

Semantic segmentation is a computer vision technique wherein each pixel in an image is labeled to identify distinct meaningful regions. This task essentially involves dividing an image into different segments, with each segment corresponding to a region carrying semantic significance.

It can be asserted that this technique holds fundamental importance for computer vision systems that require comprehending and interpreting the visual content of images in a manner akin to human perception.

#### Model architecture

The U-Net architecture adopts a cascade encoding-decoding model, where the encoder progressively reduces information to a smaller representation. Subsequently, the decoder reverses this process, returning the information to the original dimensions of the image. This gives the architecture a general "U" shape, hence the name U-Net.

One of the most distinctive aspects of the U-Net architecture are the so-called "skip connections," which allow the transfer of information from the encoder side to the decoder side. This

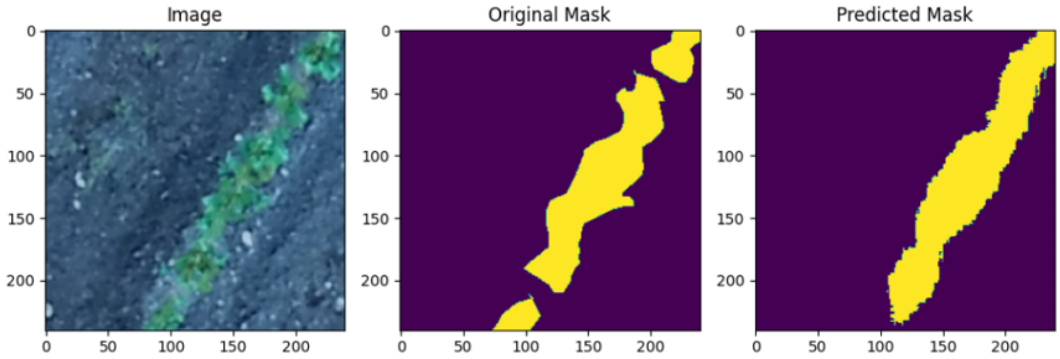



Figure 3.2: Example of segmentation

mechanism enables the model to make more accurate predictions.

As we delve deeper, the encoder processes information at increasingly higher levels of abstraction. Simply put, in the initial layers, the encoder's feature maps capture low-level details like textures and object contours. As we progress, these features start to contain more abstract information about shapes and object categories.

It is important to emphasize that both low-level and high-level information are crucial for segmenting objects in an image. For instance, variations in texture between objects and details at the edges can assist in identifying the boundaries between different objects. On the other hand, high-level information about the class to which a particular object shape belongs can be essential for correctly segmenting the corresponding pixels into the object classes they represent.

Therefore, to incorporate both sources of information during predictions, the U-Net architecture implements skip connections between the encoder and the decoder. This allows us to extract intermediate information from feature maps at various depths on the encoder side and integrate them on the decoder side, thereby enabling more accurate and refined predictions.

### Model created

In this section, a detailed description of the accomplished tasks will be provided. Firstly, a thorough examination of the model's structure (U-Net) will be conducted. Following this, the training process will be outlined. Subsequently, a discussion on how to adapt it for utilization with new datasets will be presented.

The model comprises an encoder class and a decoder class. The encoder gradually diminishes the spatial dimensions to compress information. It also augments the number of channels, signifying the quantity of feature maps in each phase. This enables the model to capture diverse details and characteristics within the image. Conversely, the decoder takes the final representation from the encoder and incrementally enlarges the spatial dimensions while reducing the number of channels. This ultimately yields a segmentation mask with the same spatial dimensions as the input image. Furthermore, a Block module is defined as the fundamental component of our encoder and

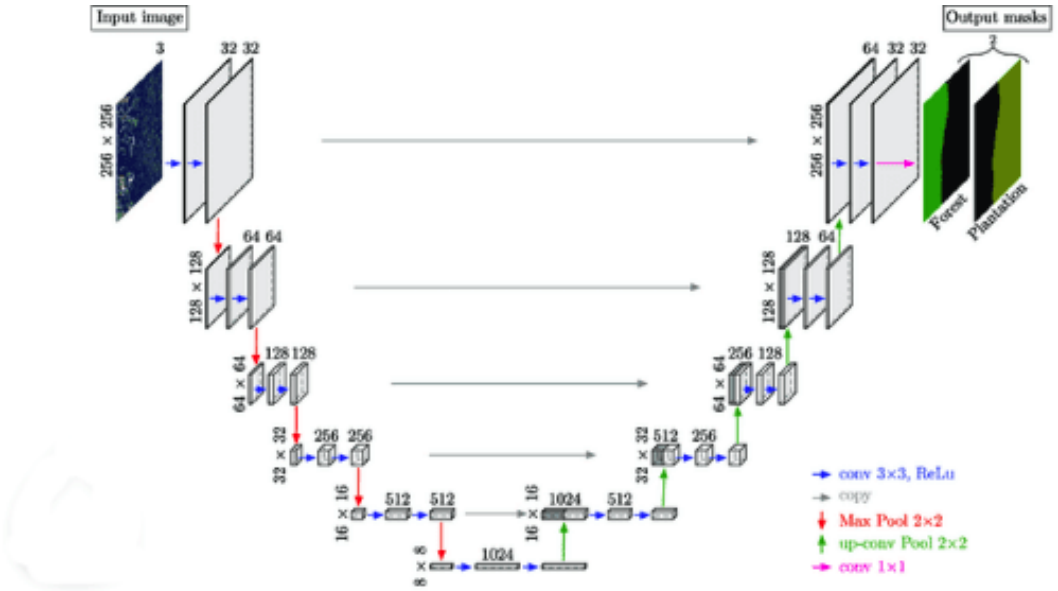



Figure 3.3: Architecture of the U-Net image segmentation model.

decoder architecture.

The Block class is defined to perform specific operations. It takes an input feature map with a specified number of channels (inChannels), applies two convolution operations with a ReLU activation in between, and produces an output feature map with a specified number of channels (outChannels). In the init constructor, two parameters, inChannels and outChannels, are accepted. These parameters determine the number of channels in the input and output feature maps. The class initializes two convolution layers along with a ReLU activation. The forward function is also defined. This function takes a feature map as input, executes the sequence of operations, and returns the resulting output feature map.

Following, the definition of the Encoder class. The class constructor, denoted by the init method, accepts a tuple (referred to as ”channels”) specifying the dimensions of channels. It is important to note that the initial value represents the number of channels in the input image, with subsequent values progressively doubling the channel dimension.

The process commences with the initialization of a list of blocks for the encoder (referred to as self.encBlocks), facilitated by PyTorch’s ModuleList feature. Each Block module takes the input channels from the preceding block and doubles the channels in the resulting output feature map. Additionally, a MaxPool2d() layer is initialized, which reduces the spatial dimensions (i.e., height and width) of the feature maps by a factor of 2.

Ultimately, the forward function for the encoder is defined. This function takes an image as input. An empty list named blockOutputs is initialized to store the intermediate outputs from the encoder blocks. This ensures that these outputs can later be passed to the decoder for further processing with the decoder feature maps.

The subsequent step involves iterating through each block in the encoder, processing the input feature map through the respective block, and appending the output to the blockOutputs list. The

max pooling operation is then applied to the block output. This process is repeated for each block in the encoder. Finally, the blockOutputs list is returned.

The definition of the Decoder class follows a similar structure as the encoder. The *init* method of the decoder, like the encoder, accepts a tuple denoted as ”channels” indicating the channel dimensions. The distinction lies in the fact that in the decoder, the channels gradually decrease by a factor of 2 as opposed to increasing, which is characteristic of the encoder.

The initialization process encompasses setting the number of channels. Additionally, a list of upsampling blocks is defined. These blocks employ the ConvTranspose2d layer to increase the spatial dimensions (height and width) of the feature maps by a factor of 2. This operation also reduces the number of channels by a factor of 2.

Furthermore, a list of blocks for the decoder is established, mirroring the structure present in the encoder.

The forward function is then defined, taking as input a feature map (x) and the list of intermediate outputs from the encoder (encFeatures). Within this function, a loop iterates through the number of channels, performing a series of operations:

Upsample the input to the decoder using the i-th upsampling block. Concatenate the i-th intermediate feature map from the encoder (encFeatures[i]) with the current output from the upsampling block. It's essential to ensure that the spatial dimensions of encFeatures[i] and x match, achieved through the crop function. Concatenate the cropped encoder feature maps (encFeat) with the current upsampled feature map along the channel dimension. Pass the concatenated output through the i-th decoder block. Upon completion of the loop, the final decoder output is returned.

The crop function is also defined, which takes an intermediate feature map from the encoder (encFeatures) and a feature map output from the decoder and spatially crops the former to match the dimensions of the latter. This is accomplished by obtaining the spatial dimensions of x (height H and width W), cropping encFeatures to match these dimensions using the CenterCrop function, and subsequently returning the cropped output.

The U-net class is constructed with the initiation of the *init* constructor method, which accepts as input the following parameters:

- encChannels: This tuple delineates the progressive augmentation in channel dimensions as the input traverses through the encoder. It initiates with 3 channels (representing RGB) and subsequently undergoes a doubling of channel count.
- decChannels: This tuple outlines the gradual reduction in channel dimensions as the input proceeds through the decoder. The channels are diminished by a factor of 2 in each step.
- nbClasses: This parameter establishes the quantity of segmentation classes, corresponding to the number of channels in the output segmentation map. Each class typically corresponds to one channel.
- retainDim: This flag indicates whether there is a desire to maintain the original output dimension.

- outSize: This parameter governs the spatial dimensions of the output segmentation map, set to mirror the dimensions of the input image.

To conclude the model, the final function within the U-Net class is the forward function.

The process initiates by forwarding the input x through the encoder, yielding the list of encoder feature maps (encFeatures). It's noteworthy that this list encompasses all the feature maps, starting from the initial encoder block output to the ultimate one, as previously discussed. Consequently, the order of feature maps in this list is reversed: encFeatures[::-1].

Subsequently, the encFeatures[::-1] list contains the feature map outputs in reverse sequence (from the last to the first encoder block). This reversal holds significance, as on the decoder side, we will be utilizing the encoder feature maps, commencing from the last encoder block output to the first.

Moving forward, we transmit the output of the final encoder block (encFeatures[::-1][0]) and the feature map outputs of all intermediate encoder blocks (encFeatures[::-1][1:]) to the decoder. The resulting output from the decoder is stored as decFeatures.

Finally, we pass the decoder output to our convolution head to derive the segmentation mask.

**Train Model**

In order to train the model, datasets from Coimbra were used, which already had the masks. The use of these masks was crucial, as manually labeling the datasets would have been much more time-consuming. Labeling images, that is, assigning categories to different features in the images, can be a challenging task for various reasons, especially in the context of computer vision. Here are some reasons for this:

- Large volume of data: Despite having a large number of images, it does not imply that there is a substantial amount of labeled data. Consequently, this can become a significant expenditure of time and resources.
- Visual complexity: Some images may contain overlapping or partially hidden objects, or they may be in challenging visual conditions, making the precise identification and labeling of these objects difficult.n.
- Ambiguity: Some images can be ambiguous, making it challenging to determine the correct category with certainty. The lack of context or additional information can lead to misinterpretations.
- Need for large labeled datasets: Creating large and accurate labeled datasets to train machine learning models is a challenge in itself. Obtaining correct and representative labels is a time-consuming process.

Therefore, the datasets were used to train the model. Both the datasets and the masks were divided into sub-images of 240 pixels by 240 pixels to generate more data and make the training process easier and faster.

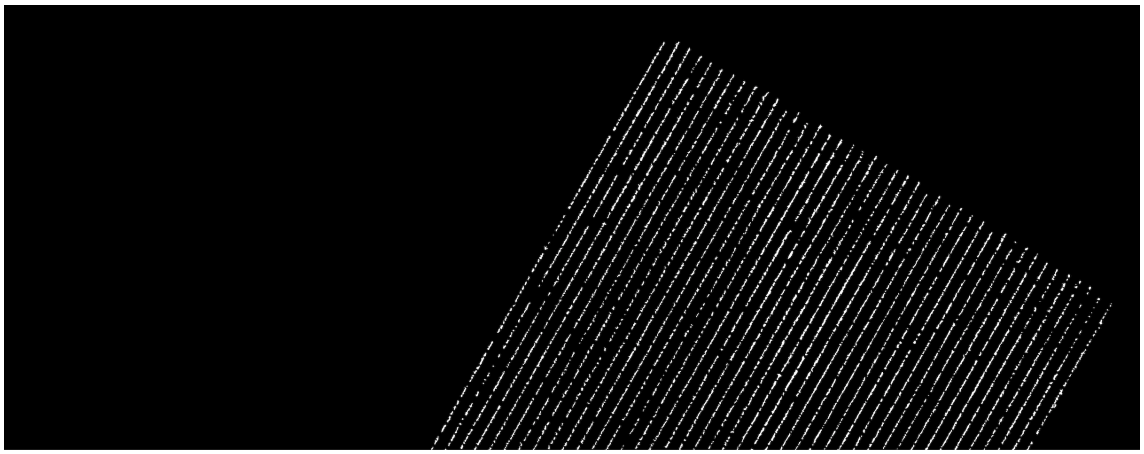


Figure 3.4: Mask of a dataset used for training the model

The explanation of the model training process will now be provided.

Firstly, two lists (imagePaths and maskPaths) are defined to store the paths of all images and their corresponding segmentation masks, respectively. The dataset is then partitioned into training and test sets using scikit-learn's train_test_split.

Transformations are defined for loading input images, including ToPILImage(), Resize(), and ToTensor(). These transformations are consolidated using the Compose function. ToPILImage() is necessary as OpenCV is used to load images, but PyTorch expects input samples in PIL format. Resize() is applied to resize images to a specific input dimension accepted by the model. ToTensor() is used to convert input images to PyTorch tensors and normalize the input PIL Image range from [0, 255] to [0, 1].

Subsequently, the train and test images and their corresponding masks are passed to the custom SegmentationDataset to create the training and test datasets. Training and test dataloaders are created using the PyTorch DataLoader class directly.

The U-Net model and training parameters are initialized. The loss function and optimizer are defined, with the Adam optimizer taking the model parameters and learning rate as input.

During the training process, the time function is used to track elapsed time. The number of epochs (NUM_EPOCHS) is defined in the training loop. Before training begins, the model is set to train mode. The training loop involves the following steps:

- Data samples are moved to the training device.
- Input image samples are passed through the U-Net model to get the output.
- The loss between the model prediction (pred) and the ground-truth label is computed.
- Backpropagation is performed to update the model parameters. This involves clearing gradients, computing gradients of the loss, and updating parameters using the optimizer.
- The training loss is tracked by adding the loss for the step to the totalTrainLoss variable.

This process is repeated until all dataset samples have been iterated through once.

### Image partition

In this section, the essential data preparation for the segmentation process will be addressed. The input comprises a dataset of vineyards, which can be either an image of the entire vineyard or a subsection captured by an Unmanned Aerial Vehicle (UAV). The desired output is the mask resulting from the detection of plants in the scene.

The inputs will be orthomosaics, which constitute a unified and georeferenced visual representation of a terrestrial area. These orthomosaics are created by merging various aerial or satellite images and are geometrically rectified to correct distortions caused by variations in terrain, altitude, or camera tilt during capture. The result is a composite image that accurately depicts the Earth's surface in the correct scale and position.

Orthomosaics find wide applications in various fields such as mapping, precision agriculture, environmental monitoring, and urban planning, providing a detailed and accurate view of large geographical areas. Their geometric accuracy is crucial for applications requiring precise measurements and spatial analyses.

However, orthomosaics are large and arbitrary data structures, making them unsuitable for direct feeding into deep learning (DL) based approaches, which typically rely on convolutional neural networks (CNN) optimized for grid-based inputs of fixed size. Additionally, the computational demands of CNNs increase proportionally with input size, making it computationally expensive to directly feed orthomosaics into DL networks.

To overcome this limitation, this work adopts the OrthoPredict approach, involving the following steps: receiving orthomosaics as inputs; dividing these orthomosaics into sub-images; feeding the sub-images into the segmentation network, which generates prediction sub-masks. Finally, the sub-masks are reconstructed into an orthomosaic mask.

The image splitting strategy was developed to divide orthomosaics from all bands into smaller sub-images with a fixed size of 240×240 pixels, providing a more manageable computational load for deep learning segmentation networks. The splitting process starts at the top-left corner of the orthomosaic, progressing to the right, creating sub-images every 240 pixels. After completing a row, a new row is defined 240 pixels below. The process is repeated until the entire orthomosaic is processed.

### Segmentation

In this section, it will be detailed how the trained model will be used to automate plant detection, considering the dataset already divided into subimages, as presented in the previous section. A function is defined that takes as input the path of the subimage and the trained segmentation model responsible for performing the segmentation.

The process begins by setting the model to evaluation mode and deactivating PyTorch gradient computation. The test image is loaded from the imagePath using OpenCV, converted to RGB format, and normalized to the pixel value range [0, 1], aligning with the model's training data.

Subsequently, the image is resized to match the standard dimensions accepted by our model.

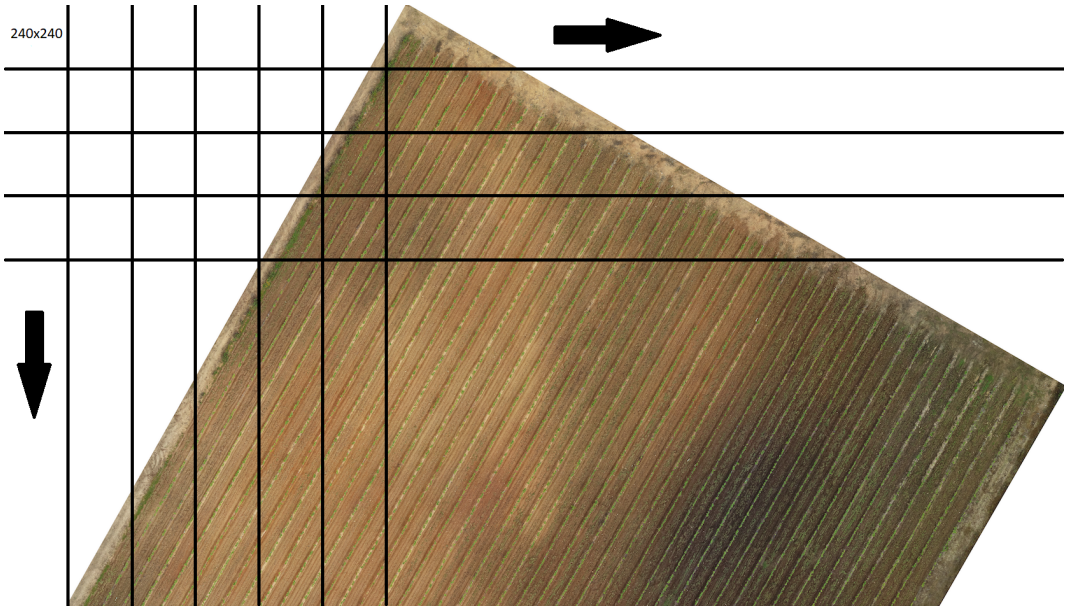



Figure 3.5: Orthomosaic splitting approach. The splitting begins at the upper left corner and proceeds to the right until the end of the row. The process is repeated until the bottom

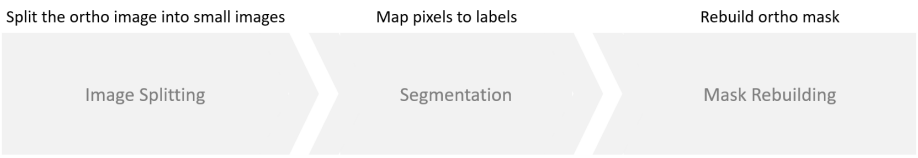



Figure 3.6: Segmentation pipeline with the following modules: image splitting, which splits the orthomosaics into sub-images; DL segmentation, which predicts sub-masks using a DL-based segmentation approach; and mask rebuilding, which uses the sub-masks to build a mask;

Currently, the image has a shape of [240, 240, 3]. However, the segmentation model requires four-dimensional inputs in the format [batch_dimension, channel_dimension, height, width].

To achieve this, the image is transposed into channel-first format ([3, 240, 240]). An additional dimension is added using the **expand_dims** function in numpy, resulting in a four-dimensional array ([1, 3, 240, 240]). The first dimension indicates the batch dimension, set to one since a single image is processed at a time. The image is then converted to a PyTorch tensor using **torch.from_numpy()** and moved to the device where the model is located.

Finally, the image is processed by passing it through the model, and the output image is saved. The sigmoid activation is applied to obtain predictions in the range [0, 1]. Since the segmentation task involves classifying pixels into two discrete classes, the config.THRESHOLD is used to binarize the output. Pixels with values greater than the threshold are assigned the value 1, while others are assigned 0. Multiplying the thresholded output by 255 ensures that the final pixel values in the output image are either 0 (representing black color) or 255 (representing white color). White

pixels indicate regions where the model has detected plants, while black pixels correspond to areas without vineyards.

Lastly, the image is saved for later reconstruction of the complete mask.

## 3.3 Otsu's Algorithm

This section was initiated due to the fact that the model outputs, which should have pixels set to 0 (black color - corresponding to areas without vineyards), were returning 30 (a dark gray instead of black). Similarly, pixels that should be white were returning a value of 225 instead of the expected 255 (white color - corresponding to areas with vineyards).

This discrepancy is attributed to incorrect values in the segmentation, where the values should ideally be 0 and 255. To address this problem and enable the comparison of the results with the ground truth mask for accuracy assessment, the Otsu algorithm will be used to adjust the obtained mask values to 0 and 255.

In image processing, the application of thresholding involves binarizing an image based on pixel intensities. This algorithm typically takes a grayscale image and a specified threshold as input, producing a binary image as output.

When the intensity of a pixel in the input image exceeds the threshold, the corresponding output pixel is designated as white (foreground). Conversely, if the input pixel intensity is less than or equal to the threshold, the output pixel location is marked as black (background).

A drawback of simple thresholding lies in the manual specification of the threshold value. While one can experiment with different values, this process is labor-intensive and may not be suitable for real-world scenarios.

The Otsu algorithm[16], developed by Nobuyuki Otsu in 1979, is an image segmentation method that aims to automatically find the optimal threshold to separate two classes of pixels in a grayscale image. The threshold is a value that determines whether a pixel belongs to the background or object class in the binarized image.

The central idea of the Otsu algorithm is to maximize the variance between the two classes of pixels, considering the distribution of grayscale levels in the image. Variance is a measure of the spread of pixel values around the threshold. The higher the variance between the two classes, the more effective it is, considering that, as observed in section 3.2.1, the images have already undergone segmentation. This process is occurring solely for the pixel values to match those of the mask, allowing for comparison.

Here is a detailed explanation of the Otsu algorithm:

- Histogram: Calculate the histogram of the grayscale image. A histogram is a graphical representation of the distribution of pixel intensities.

- Pixel Probabilities: Normalize the histogram by dividing each pixel count by the total sum of pixels in the image to obtain the probabilities of occurrence for each grayscale level.

- Cumulative Probability: Calculate the cumulative probability for each grayscale level by summing the pixel probabilities up to that point.
- Global Mean and Global Variance: Calculate the global mean and global variance of the image.
- Iterative Loop: Start an iterative loop over all possible thresholds (grayscale levels). For each threshold, calculate the cumulative probability, mean, and variance for both classes (background and object classes) using normalized pixel probabilities.
- Maximization of Interclass Variance: Calculate the variance between the two classes using Otsu's interclass variance formula: $\sigma^2 = \rho(class1) \cdot \rho(class2) \cdot (\mu(class1) - \mu(class2))^2$. The threshold that maximizes this variance is chosen as the optimal threshold.
- Binarization: Binarize the image using the optimal threshold, where pixels below the threshold are assigned to the background class, and pixels above the threshold are assigned to the object class.

The image is binarized using the optimal threshold, where pixels below the threshold are assigned to the background class, and pixels above the threshold are assigned to the object class. The Otsu algorithm is highly effective for images with a clear bimodal distribution, that is, images where pixel intensities are clustered around two distinct values, as is the case.

## 3.4 Crop Row Detection

In this section, the detection of vineyard planting lines is discussed. The cultivation line detection will be applied to the images obtained from segmentation. As depicted in Figure 3.7, the blue line corresponds to the cultivation line. Hough Transform was employed for this cultivation line detection, and its functioning will be elucidated in Section 3.4.1.

### 3.4.1 Hough Transform

The Hough Transform serves as an algorithm for identifying geometric shapes, like lines, circles, and ellipses, within images. Proposed by Paul Hough, the algorithm has found applications across various domains in image processing and computer vision. This explanation will outline the workings of the Hough algorithm, particularly in line detection because it was used.

Line detection employing the Hough Transform relies on a parametric representation of lines. In this representation, lines are characterized by polar parameters: each coordinate (x, y) within the image corresponds to a line defined by (rho, theta), where rho ($\rho$) signifies the distance from the point (x, y) to the origin, measured perpendicularly to the line, and theta ($\theta$) represents the angle formed by the line perpendicular to the point (x, y) and the x-axis.

The process involves constructing an Accumulator Matrix, where a sinusoidal curve is delineated in the (rho, theta) space for every edge point detected in the image. Each point along these

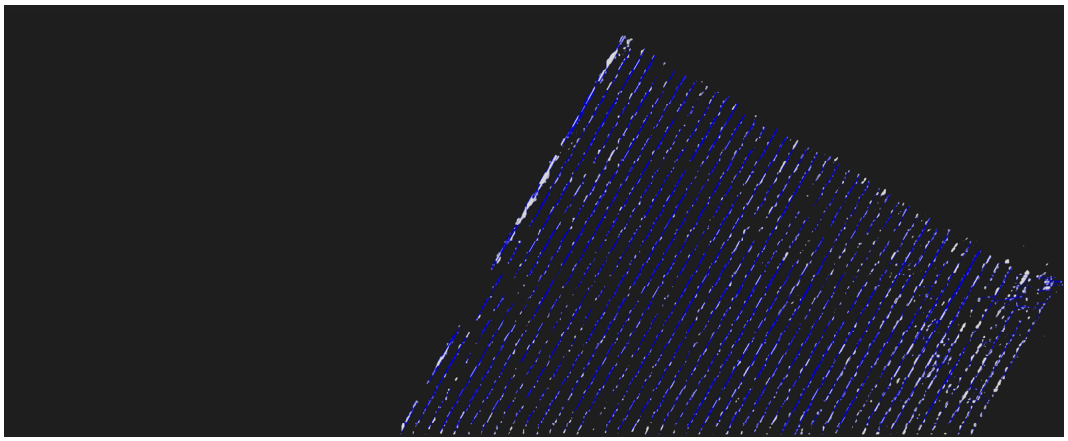


Figure 3.7: Image already with Hough Transform aplication for the crop row detection

curves contributes to the accumulator matrix, accumulating votes for lines specified by the parameters (rho, theta). The intersections within this matrix highlight values of (rho, theta) that have amassed a notable number of votes, with each intersection indicating a detected line in the image.

Following this, thresholding and post-processing steps are undertaken. Thresholding is applied to determine which intersections are to be considered as detected lines. Furthermore, additional parameters such as minLineLength and maxLineGap can be utilized to filter lines based on length and distance between segments.

In practical implementation, the process begins with edge detection, typically employing techniques like the Canny operator. Multiple lines are then drawn in the parameter space (rho, theta) for each edge point, and the accumulator matrix is updated with votes from all lines represented by edge points. Subsequent analysis of the accumulator matrix identifies significant intersections, which are indicative of detected lines. Post-processing involves applying a threshold, grouping nearby lines, and filtering based on additional criteria.

The Hough Transform demonstrates resilience to noise and the ability to detect lines even in scenarios with complex intersections. However, it is noteworthy that its computational intensity can fluctuate based on image size and parameter space resolution.

## 3.5 Summary

This chapter described the datasets used for training and validating the model. The developed model was also detailed, including the architecture used and the training process. Additionally, the segmentation approach was outlined.

Following that, the algorithms employed, such as the Otsu algorithm and the Hough Transform for crop line detection, were explained.

# Chapter 4

# Results

In this chapter, the outcomes of the research efforts and associated work detailed in Chapter 2 and the work done in chapter 3.

Section 4.1 will be dedicated to the presentation of the segmentation results. Initially, a comparison will be made between the datasets used to train the model, distinguishing between those with real masks and those with masks predicted by the trained model. This section will also include a breakdown of the phases undertaken to achieve the best possible values in the evaluation metrics.

Subsequently, Section 4.2 will provide a detailed exposition of the results of crop line detection obtained from the datasets used in the study. Finally, Section 4.3 will present a summary of this chapter.

## 4.1 Segmentation

### 4.1.1 Evaluation

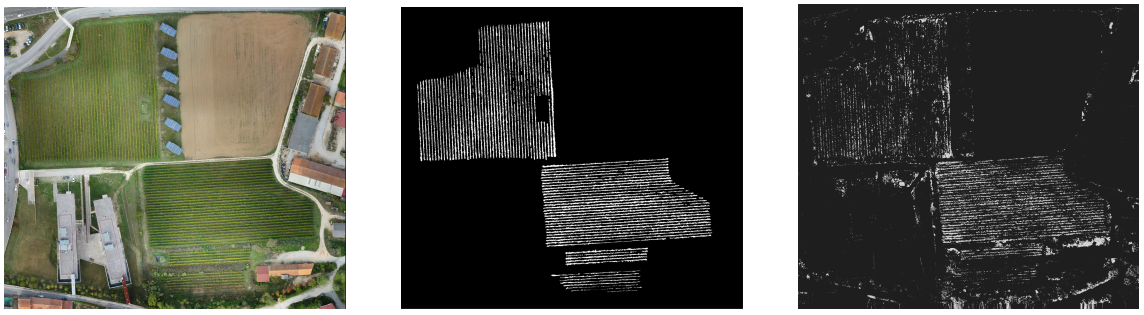
#### 1º Phase

After training the model, it is necessary to validate and test it. This phase involves evaluating and testing the model using different datasets. For evaluation, metrics such as accuracy, precision, and intersection over union, among others, were used. These metrics were explained in Section 2.4. In this section, the interpretation of the results and the process to achieve the best possible outcomes will be discussed.

Initially, the datasets were segmented to validate the model's performance after training. However, although the visual results appeared reasonable, a comparison with the real mask using the selected metrics revealed that the results were far from satisfactory. This led to an investigation into the reasons behind the model's poor performance.

The first issue was that when comparing the real mask with the predicted mask, the accuracy was 0.0. The reason for this was the difference in pixel values. This issue required the use of the Otsu algorithm to automatically regularize the predicted masks, ensuring that both the predicted and real masks had the same values. The pixel values differed because the trained model was producing results where pixels that should have been defined as 0 (black color - corresponding to

areas without vineyards) were returning a value of 30 (a dark gray instead of black). Similarly, pixels that should have been white were returning a value of 225 instead of the expected value of 255 (white color - corresponding to areas with vineyards).



(a) Dataset (b) Real mask (c) Predict mask in phase 1

Figure 4.1: Different values of pixels between (b) and (c)

The algorithm was then applied to standardize the values, converting all pixel values of 30 to 0 and all values of 225 to 255. This adjustment enabled a correct comparison between the predicted and real masks, allowing the metrics to return values. However, some metrics still exhibited very low values. Nonetheless, reasonable accuracy values were obtained, as shown in the following table.

| Dataset | Accuracy |
|---|---|
| Dataset 1 | 77.84 |
| Dataset 2 | 95.22 |

Table 4.1: Accuracy in phase 1

**2º Phase**

In Phase 1, there were significant improvements; however, some metrics remained very low. The second improvement involved preparing the initial datasets with dimensions that were multiples of 240 pixels. The dataset was divided into 240-pixel sub-images for segmentation, but the last row often did not align with a multiple of 240. When the model performed segmentation, it added pixels that affected the results. By ensuring that the dataset dimensions were multiples of 240, false negatives were eliminated. To address this, an algorithm was implemented to check the dataset size to determine if it was a multiple of 240. If not, the algorithm divided the height and width of the image by whole numbers to obtain the ideal dataset size, effectively removing pixels to make the dimensions multiples of 240.

The table 4.2 presents the datasets after pre-processing, showing the new sizes, pixel count, and the number of sub-images. This pre-processing step allowed for the achievement of the best possible results.

| Dataset | #Processed image | #Pixels | #Number of images |
|---|---|---|---|
| Dataset 1[5] | 15840x14160 | 224294400 | 3894 |
| Dataset 2[5] | 23280x8640 | 201139200 | 3492 |
| Dataset 3[5] | 22320x23280 | 519609600 | 9021 |
| Dataset 4[1] | 6240x13680 | 85363200 | 1482 |
| Dataset 5[1] | 6480x11040 | 71539200 | 1242 |
| Dataset 6[1] | 7680x11760 | 90316800 | 1568 |
| Dataset 7[1] | 23280x9120 | 21231360 | 3686 |
| Dataset 8[1] | 28560x11520 | 329011200 | 5712 |
| Dataset 9[1] | 6720x11280 | 75801600 | 1316 |

Table 4.2: Table with data from datasets after being transformed for better segmentation

**Results**

**Dataset 1 and 2:**

Figure 4.2: Dataset 1

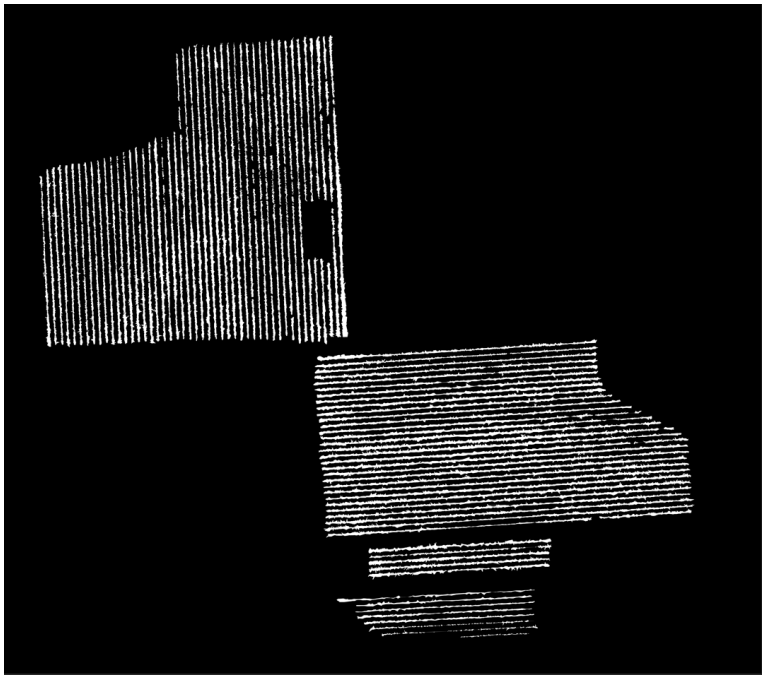


Figure 4.3: Mask Dataset 1

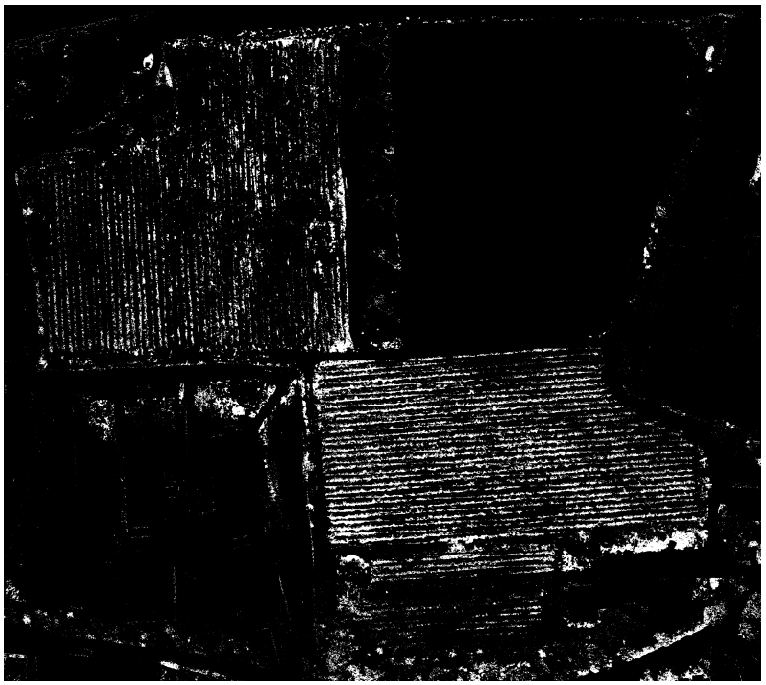


Figure 4.4: Segmentation of Dataset 1

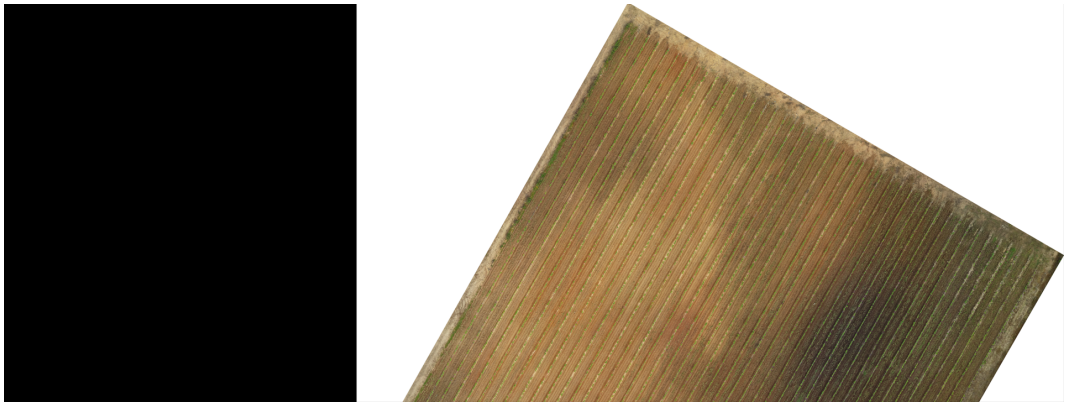


Figure 4.5: Dataset 2

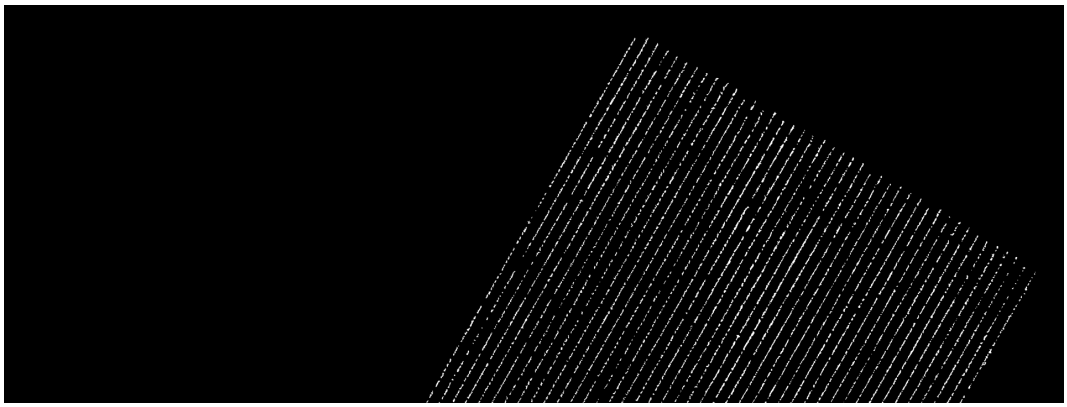


Figure 4.6: Mask Dataset 2

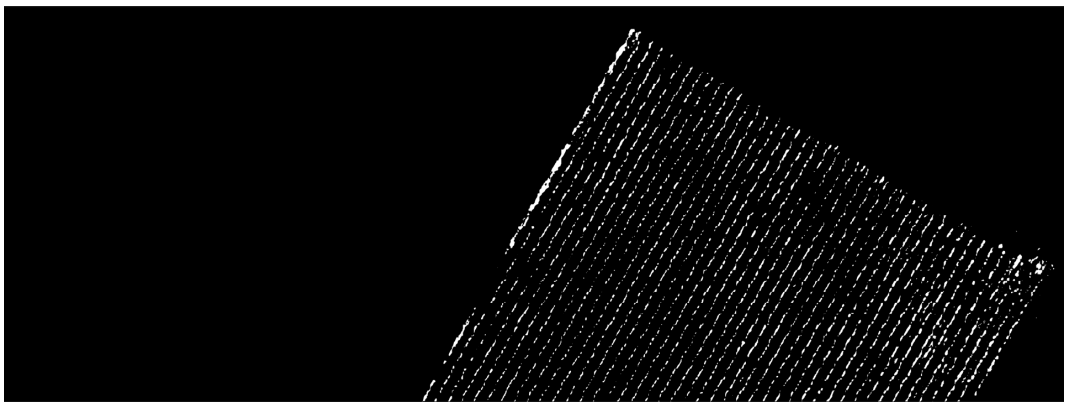


Figure 4.7: Segmentation of Dataset 2

| Dataset | Accuracy | Iou | Dice Coefficient | Precision | Recall |
|---|---|---|---|---|---|
| Dataset 1 | 90.39 | 0.2 | 0.32 | 0.42 | 0,27 |
| Dataset 2 | 97.64 | 0.33 | 0.49 | 0.47 | 0.52 |

Table 4.3: Model results on datasets 1 and 2

Analyzing the results from the two datasets, it appears that the model performs quite differently on each. Let us evaluate this in more detail.

Dataset 1:

- Accuracy (90.39%): Although this seems high, accuracy is not always the best indicator for segmentation models, especially when there is class imbalance.
- IoU (0.2): This value is quite low, suggesting that the overlap between the prediction and the true label is small relative to their union. This indicates that the model struggles to correctly segment the vineyard areas.
- Dice (0.32): The Dice coefficient is also low, further reinforcing that the model has difficulty accurately capturing the vineyard areas.
- Precision (0.42): The model has a low precision for predicting ”vineyards,” with many false positives.
- Recall (0.27): Recall is particularly low, indicating that the model is ”missing” many areas that should be identified as vineyards.

Dataset 2:

- Accuracy (97.64%): Significantly higher accuracy than dataset 1, but again, this may mask class imbalances.
- IoU (0.33): Although still not ideal, the IoU is significantly better compared to dataset 1, suggesting a more accurate segmentation.
- Dice (0.49): The Dice coefficient is close to 0.5, indicating an improvement in the correspondence between predicted and actual vineyard areas.
- Precision (0.47): A slight increase in precision, meaning the model is generating fewer false positives than in dataset 1.
- Recall (0.52): This is a positive sign, as the model is able to correctly identify more vineyard areas compared to the first dataset.

However, when visually comparing the datasets, especially dataset 2, it becomes apparent that while the results are quite similar to the ground truth, there are still some false negatives. Despite this, one would expect better overall performance.

Upon analyzing the images, it becomes evident that the segmentation does not align perfectly at the pixel level, meaning there is spatial variation in the segmentation, as seen in the images.

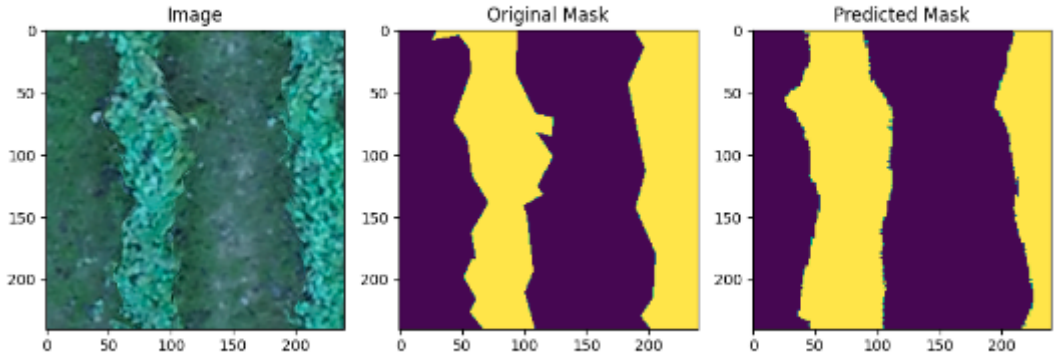



Figure 4.8: Comparison of subimages in segmentation to better see the classification of pixels in dataset 1

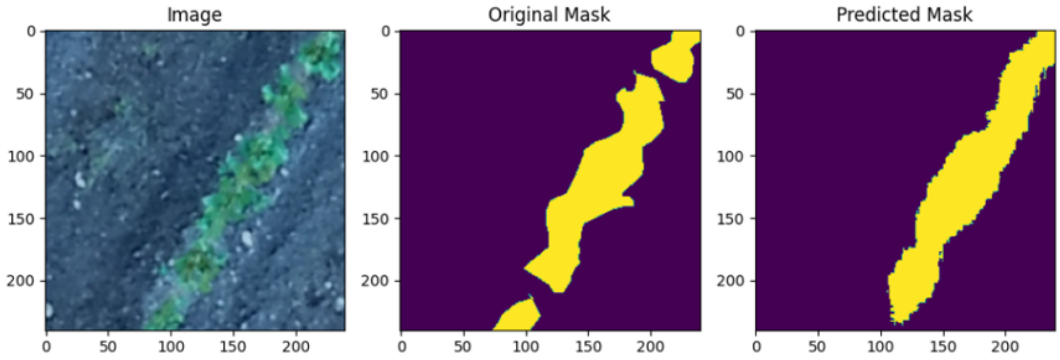



Figure 4.9: Comparison of subimages in segmentation to better see the classification of pixels in dataset 2

This issue is known as spatial inconsistency or spatial imprecision in segmentation. It occurs when the model fails to precisely align the segmented areas with the regions of interest (in this case, the vineyards) from one image to another. This can result in predictions that are misaligned or inconsistent, especially in high-resolution images or when there is variation in the size and shape of the objects being segmented (such as leaves or branches).

The most plausible causes for this issue include:

- Data variation: Differences in lighting, perspective, or even noise within the dataset may confuse the model, leading to inconsistent predictions across similar images.
- Lack of spatial resolution: Segmentation models often work with reduced resolutions during the downsampling/upsampling process (in networks like U-Net or FCN), which can result

in a loss of spatial details. This, in turn, affects the precision of the segmentation and can cause misalignment between the predicted masks and the actual vineyard areas.

Addressing these issues may require improvements in the dataset, such as normalizing the lighting conditions or adding more diverse training examples, as well as refining the model architecture to preserve spatial resolution throughout the segmentation process. However, the availability of datasets with annotated masks is significantly limited and difficult to obtain.

### 4.1.2 Model Validation

In this section, the original datasets will be presented along with the predicted masks generated by the developed model.

**Dataset 3**

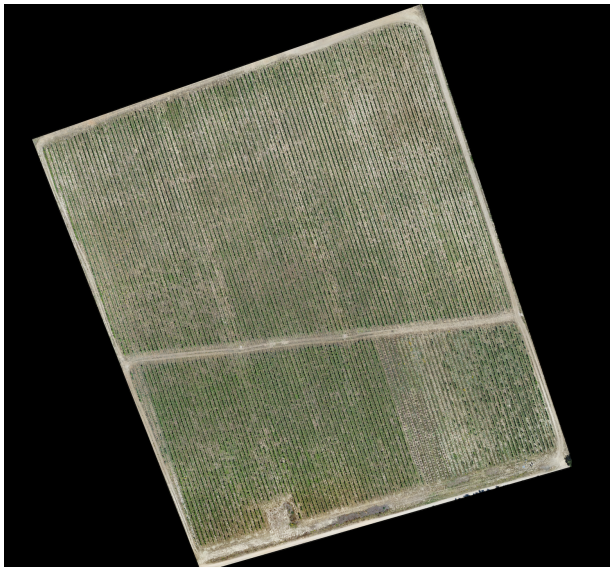


Figure 4.10: Dataset 3

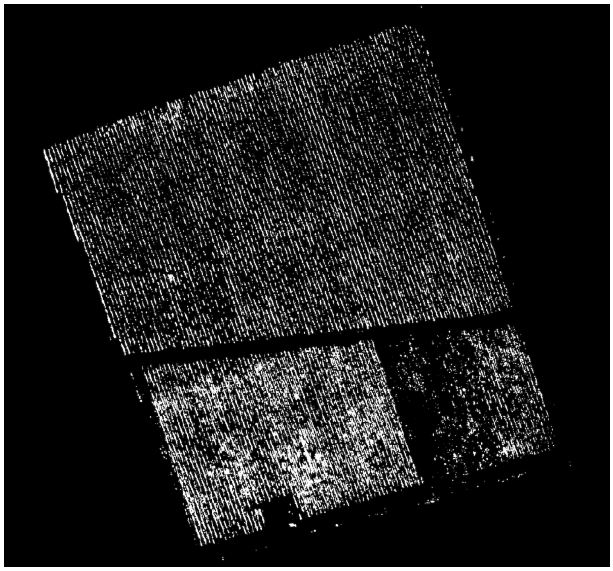


Figure 4.11: Segmentation of Dataset 3

**Dataset 4**

Figure 4.12: Dataset 4

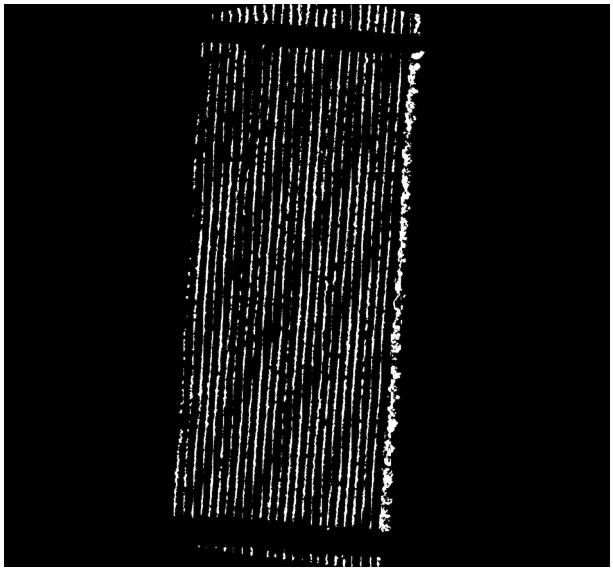


Figure 4.13: Segmentation of Dataset 4

**Dataset 5**

Figure 4.14: Dataset 5

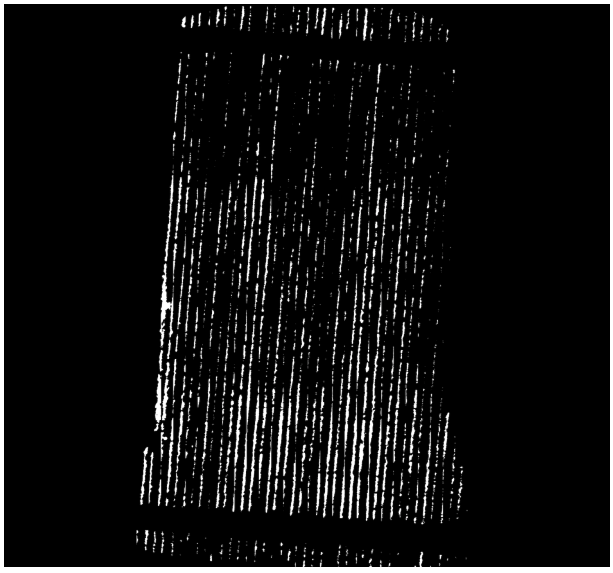


Figure 4.15: Segmentation of Dataset 5

**Dataset 6**

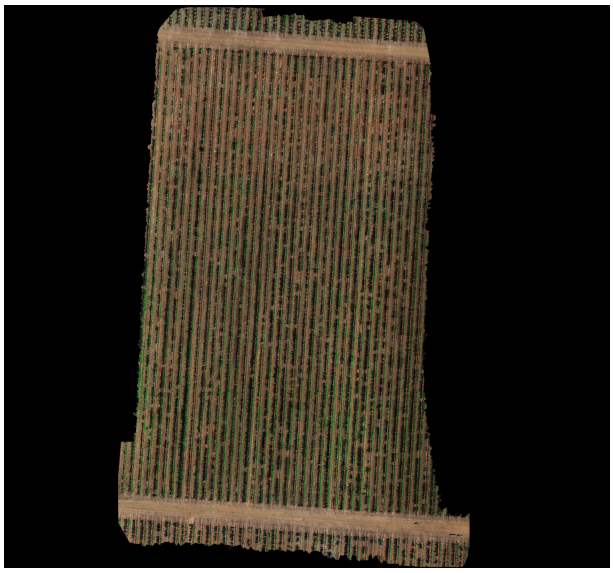


Figure 4.16: Dataset 6

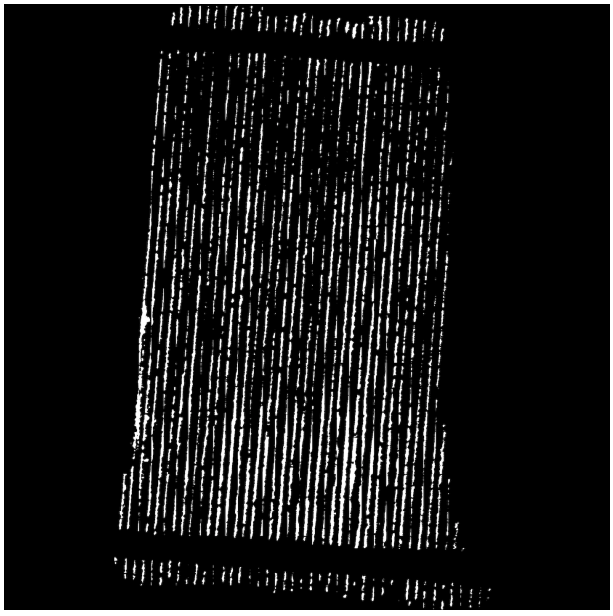


Figure 4.17: Segmentation of Dataset 6

**Dataset 7**

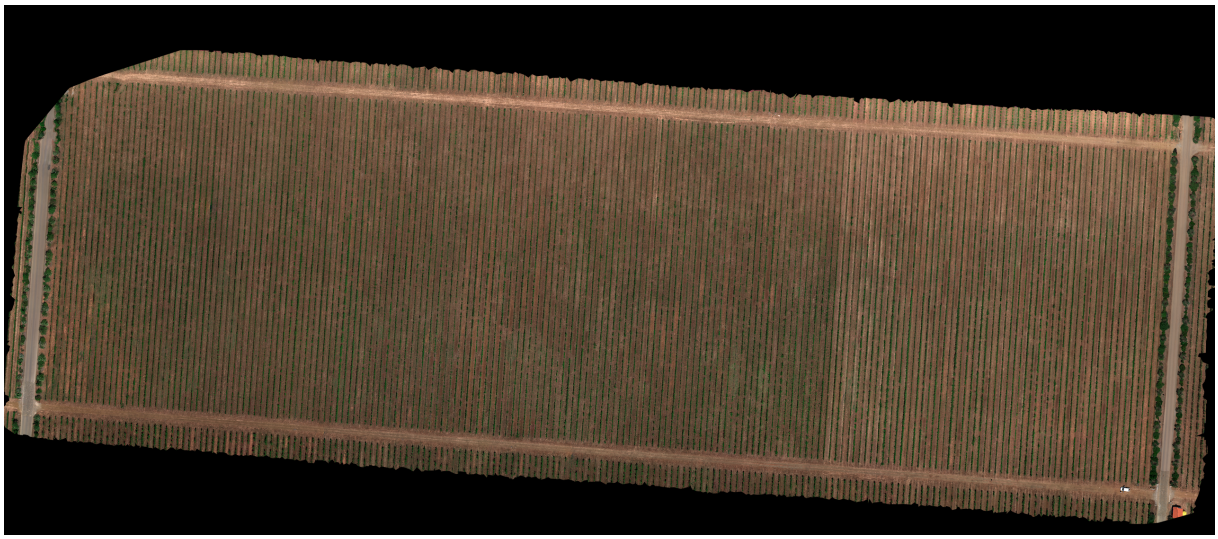


Figure 4.18: Dataset 7

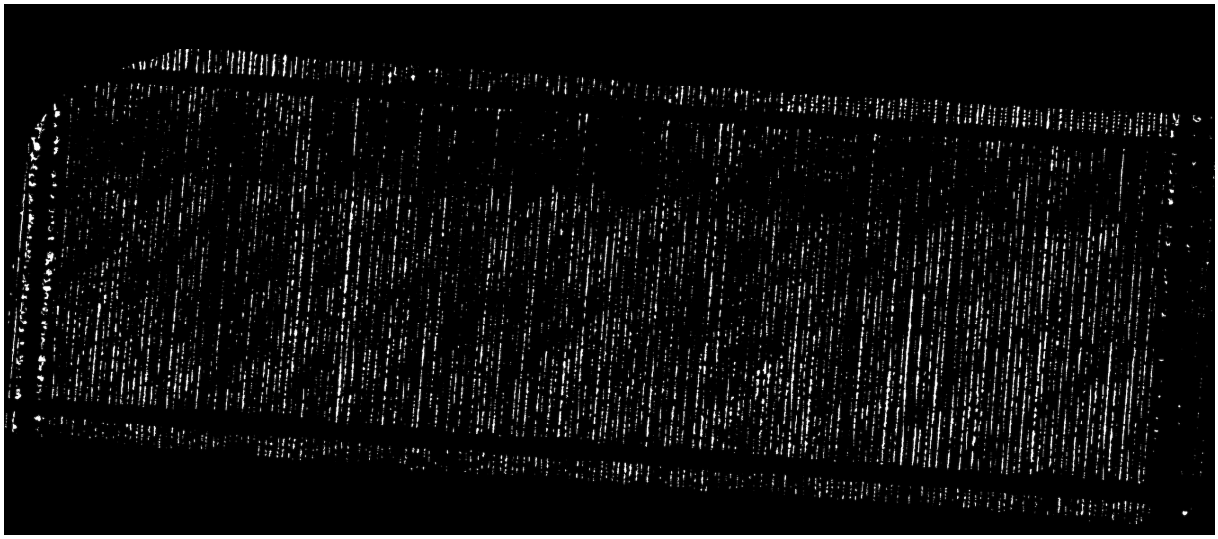


Figure 4.19: Segmentation of Dataset 7

### Dataset 8

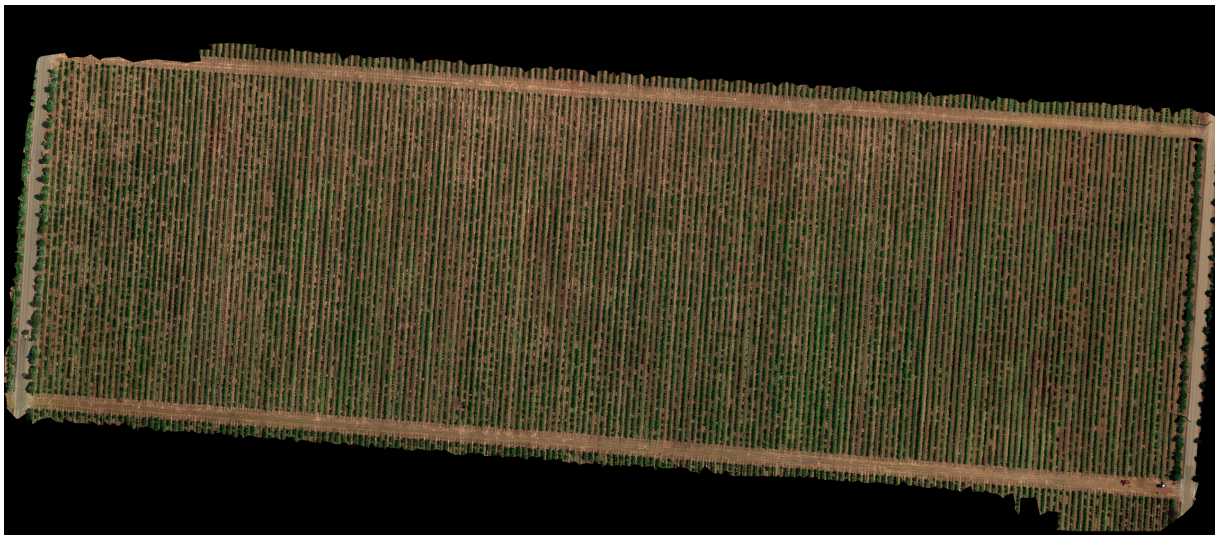


Figure 4.20: Dataset 8

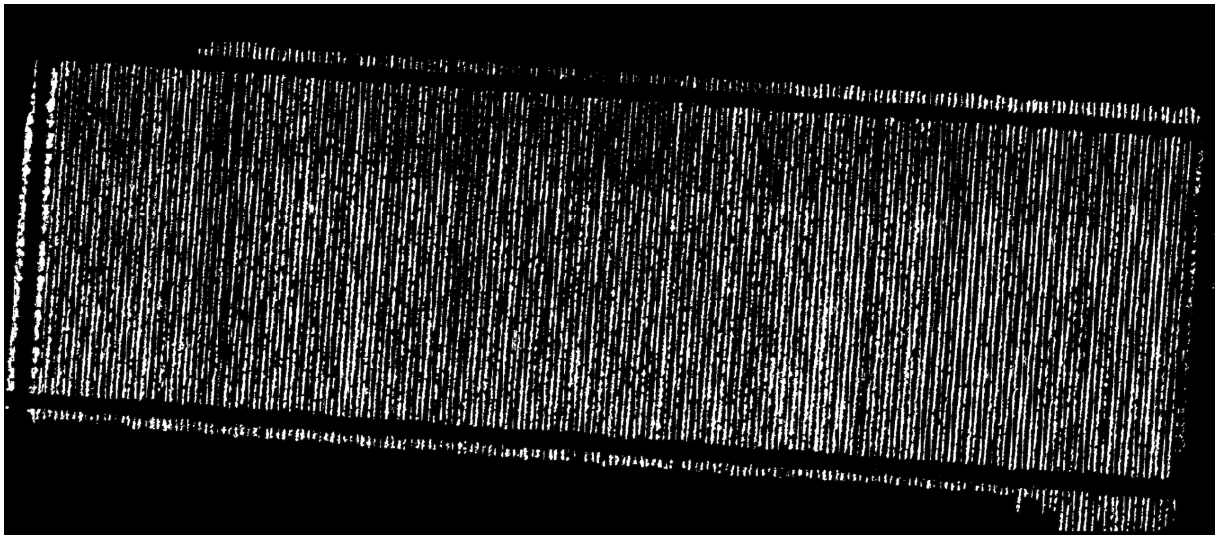


Figure 4.21: Segmentation of Dataset 8

**Dataset 9**

Figure 4.22: Dataset 9

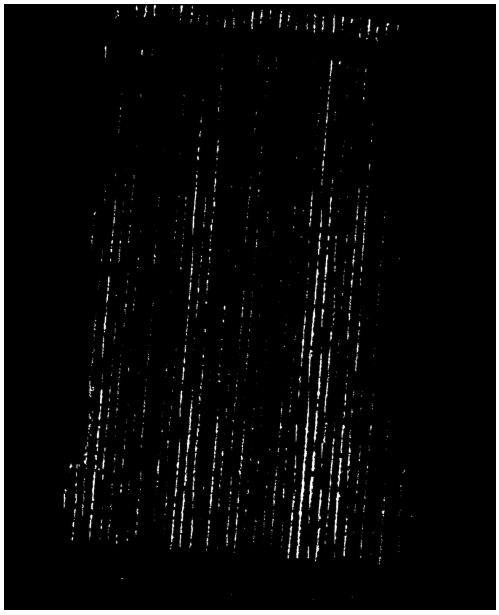


Figure 4.23: Segmentation of Dataset 9

## 4.2 Crop Row Detection

In this section, the datasets with the detected crop lines will be presented. Line detection was performed using the Hough Transform on the predicted masks generated by the developed model.

### Dataset 1

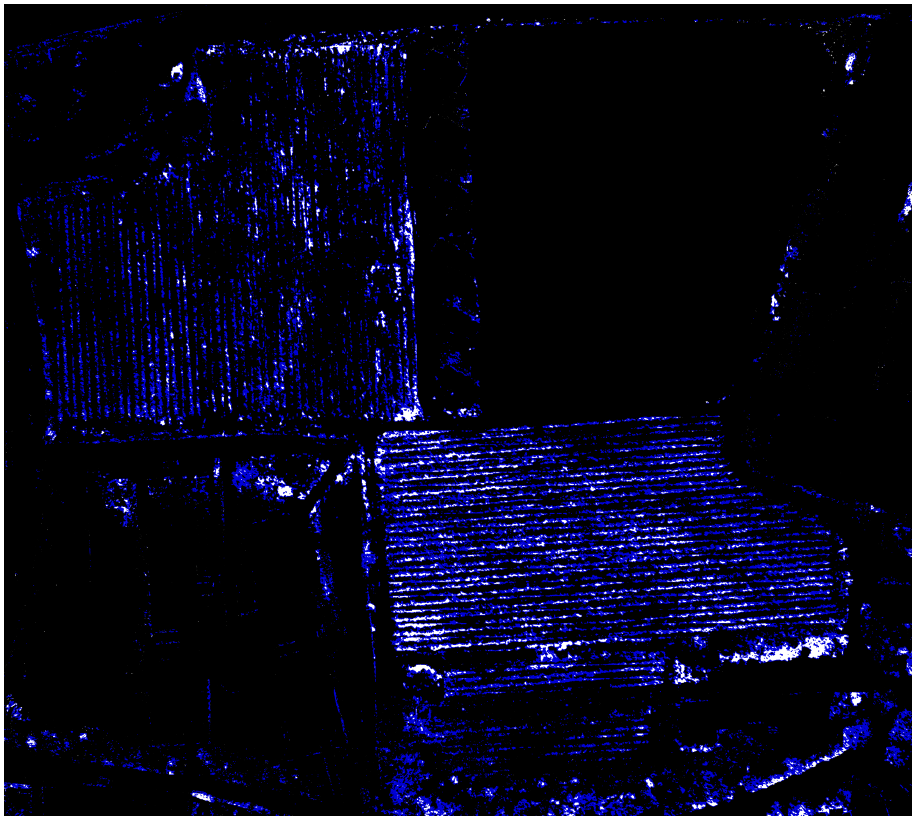


Figure 4.24: Crop row detection of Dataset 1

### Dataset 2

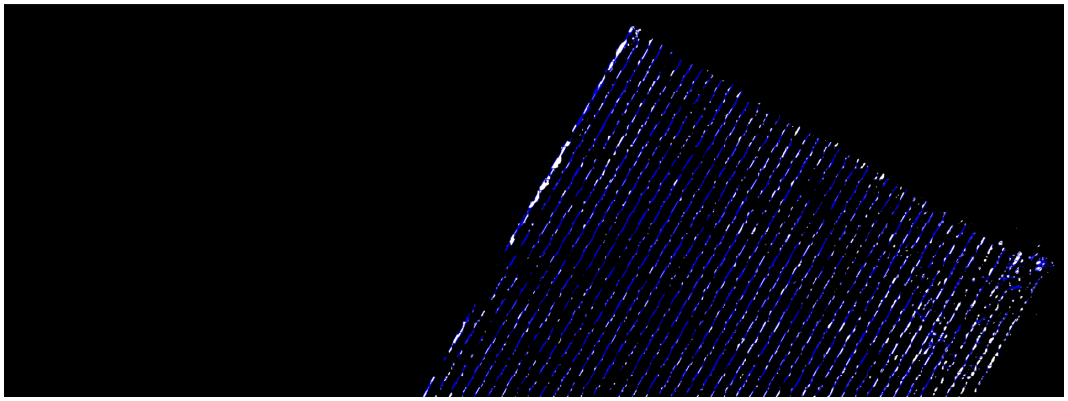


Figure 4.25: Crop row detection of Dataset 2

### Dataset 3

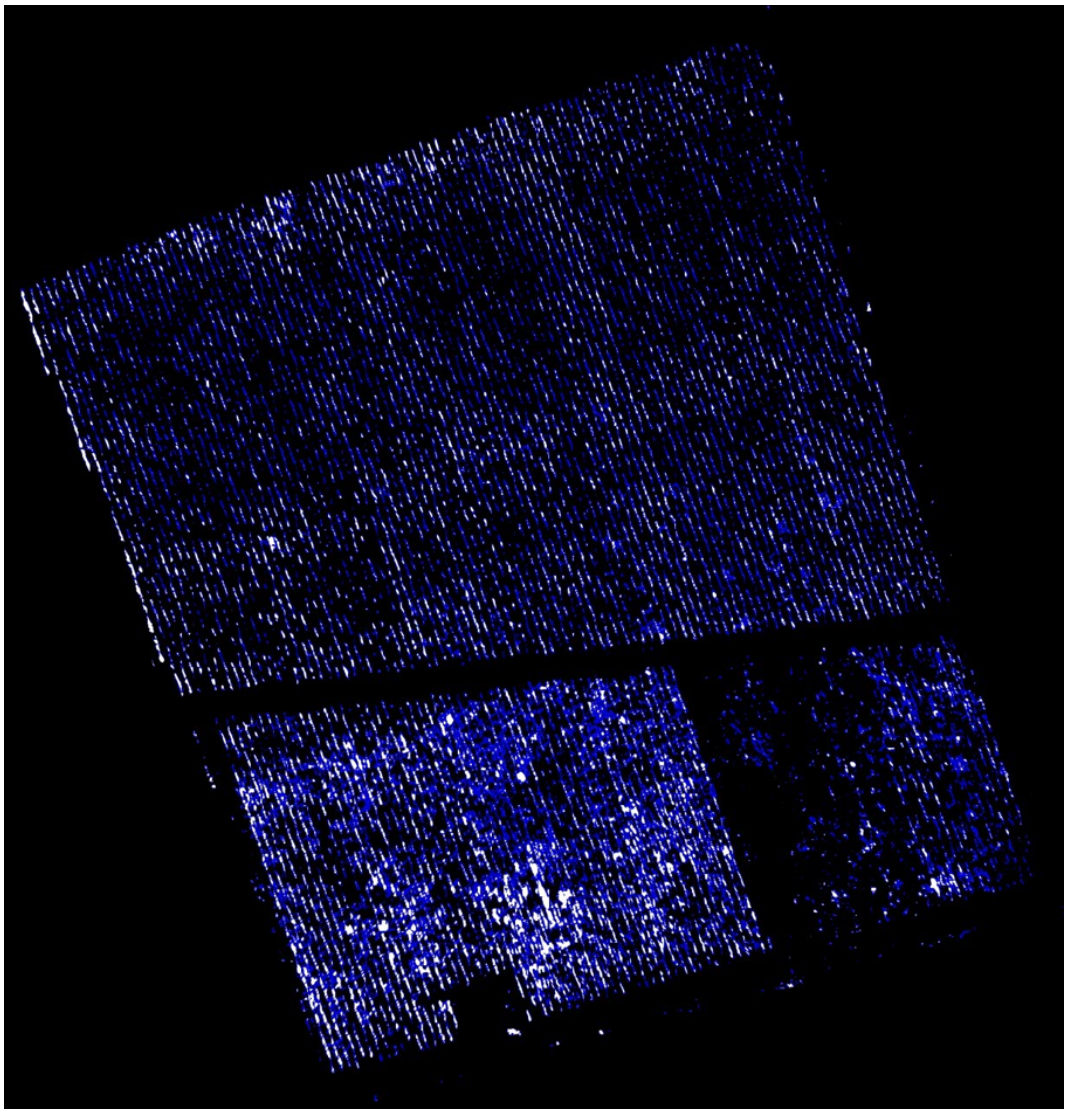


Figure 4.26: Crop row detection of Dataset 3

### Dataset 4

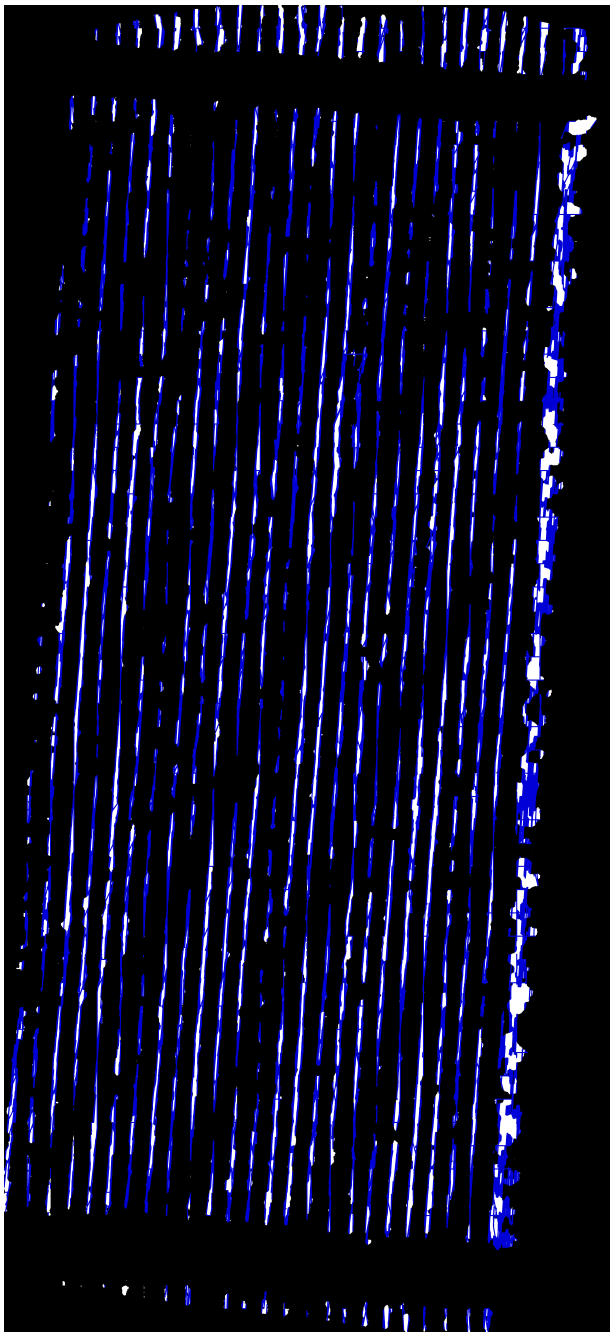


Figure 4.27: Crop row detection of Dataset 4

### Dataset 5

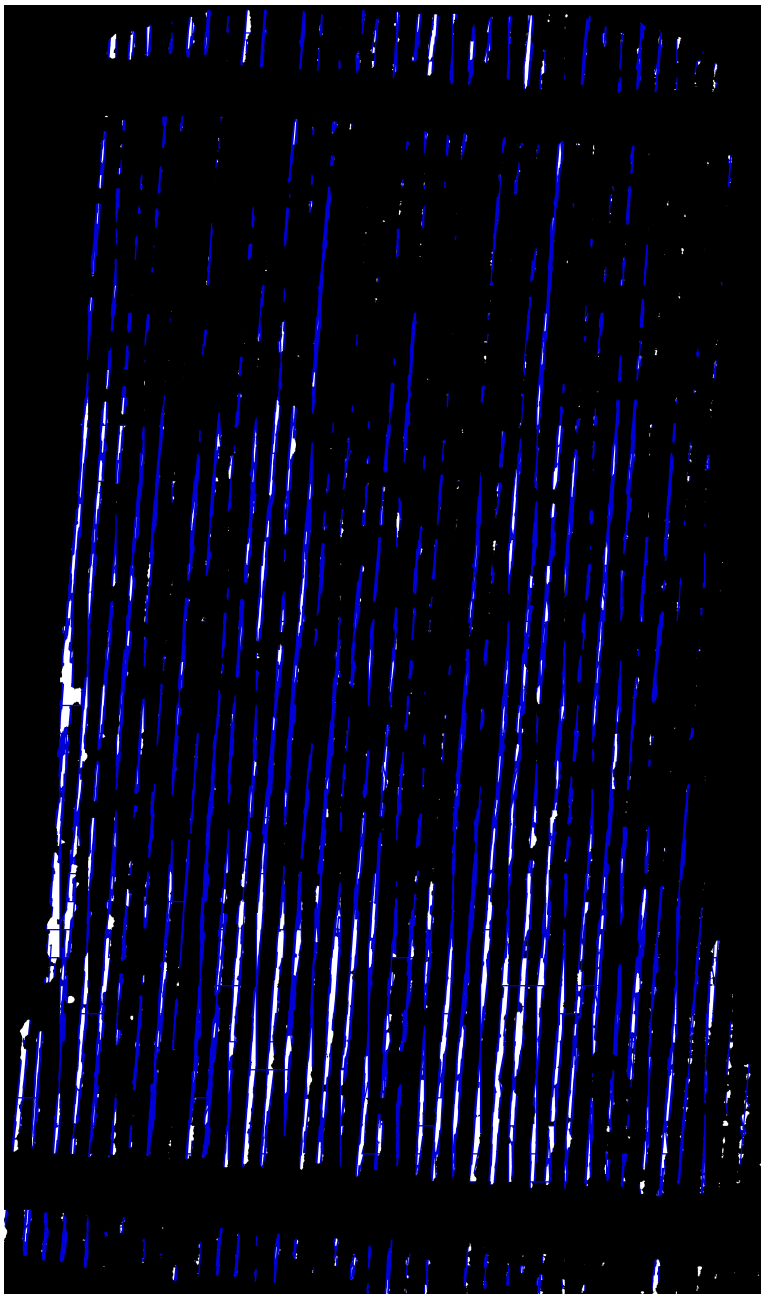


Figure 4.28: Crop row detection of Dataset 5

### Dataset 6

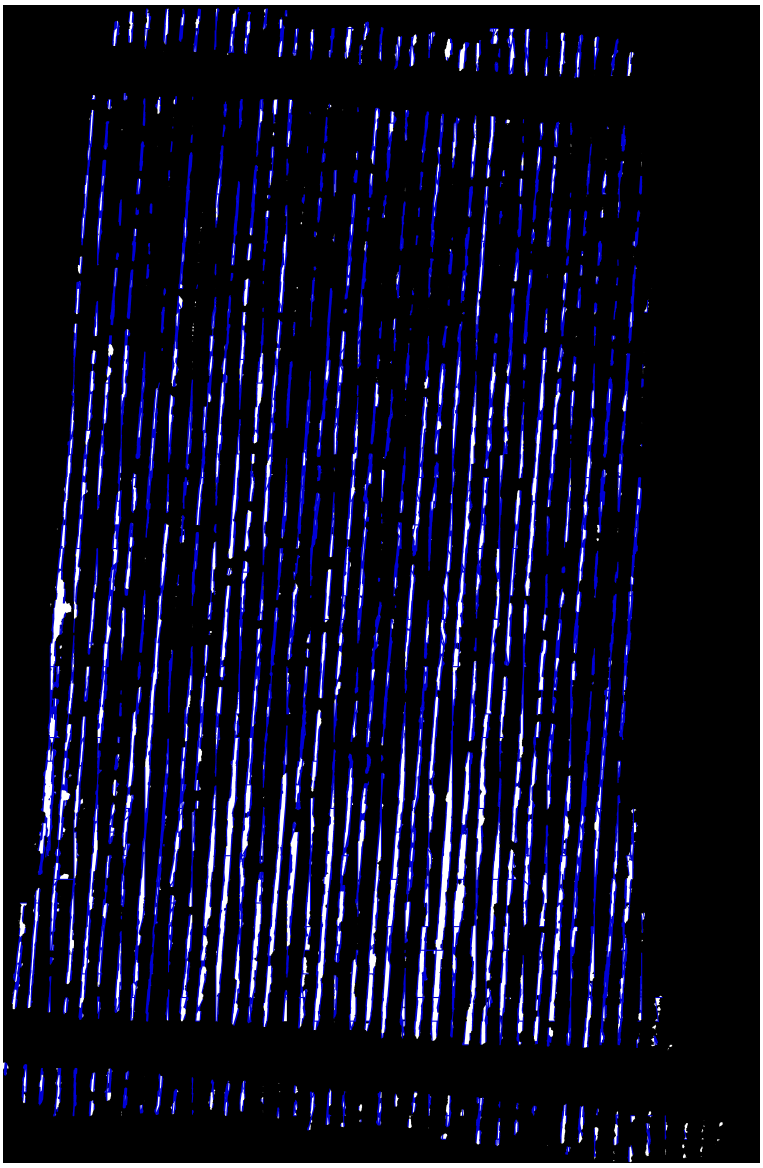


Figure 4.29: Crop row detection of Dataset 6

### Dataset 7

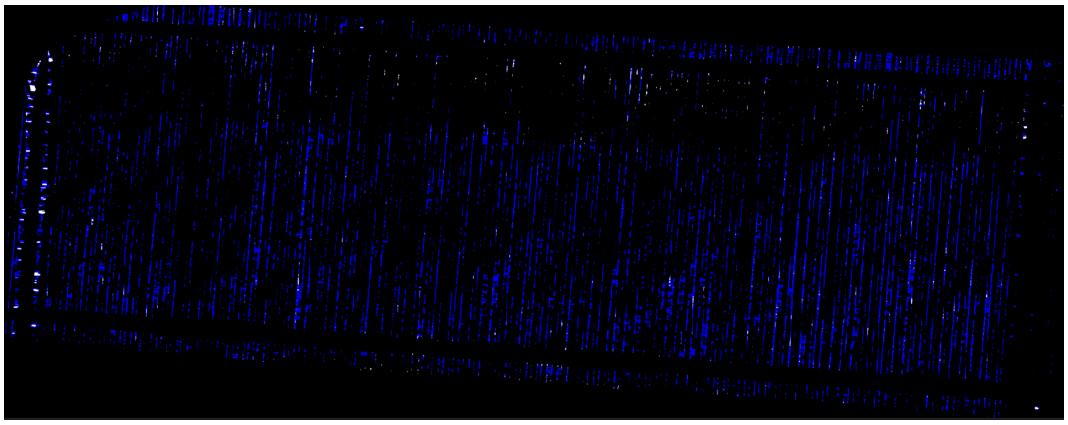


Figure 4.30: Crop row detection of Dataset 7

### Dataset 8

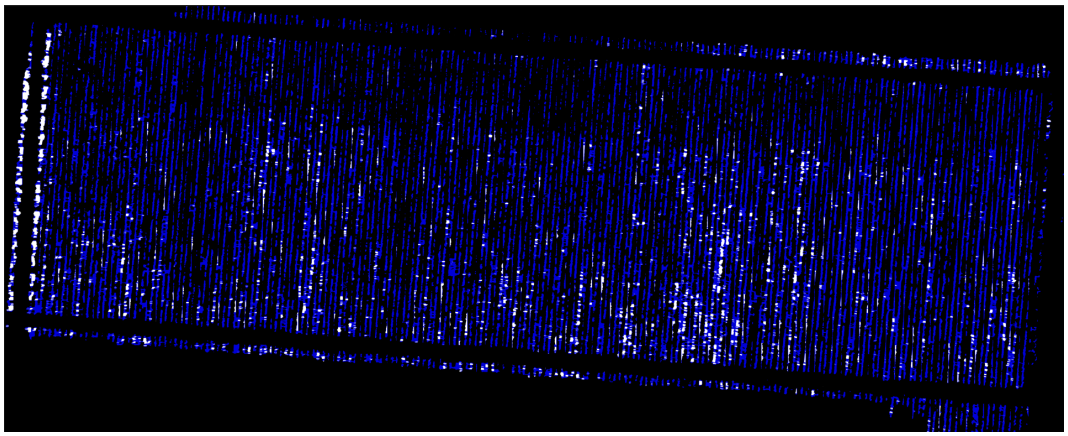


Figure 4.31: Crop row detection of Dataset 8

### Dataset 9

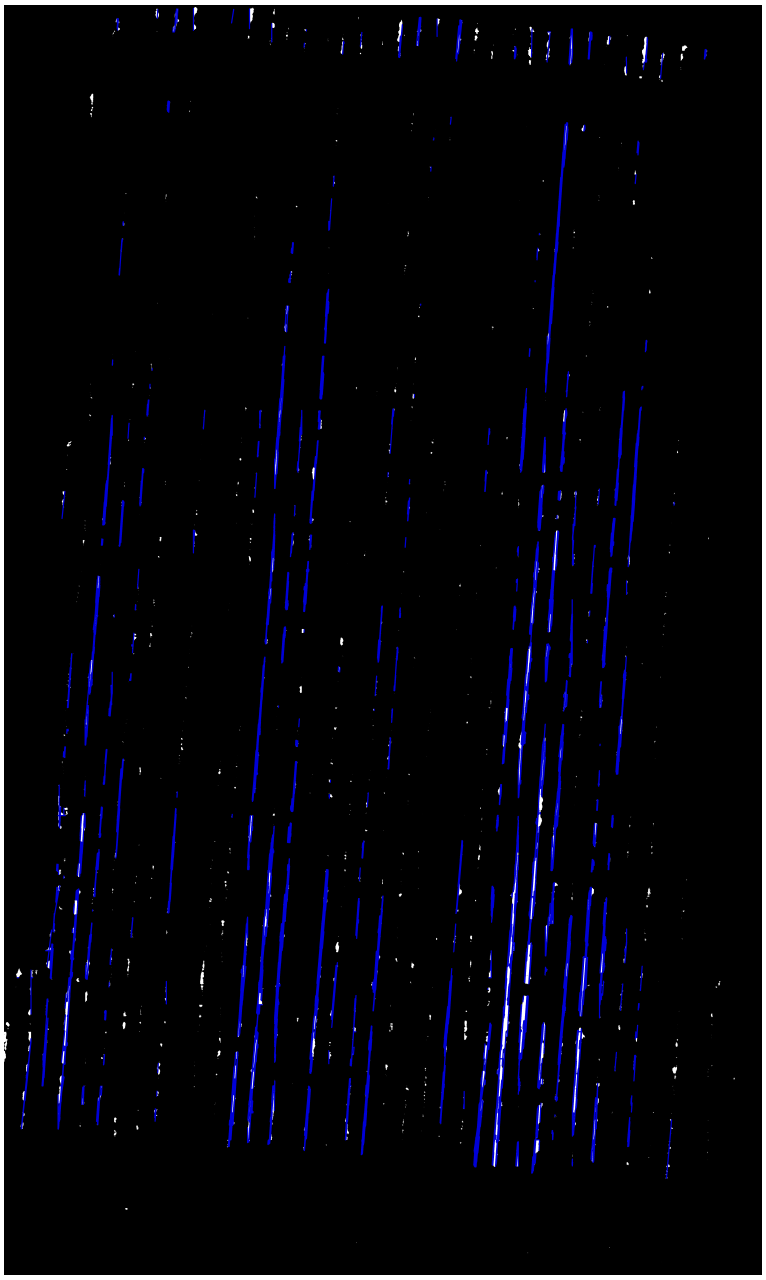


Figure 4.32: Crop row detection of Dataset 9

## 4.3 Summary

This chapter addressed the initial issues encountered with the developed model and the solutions implemented to resolve them. The evaluation metrics results were also presented, along with an explanation for why the outcomes were somewhat below expectations.

All validation datasets with their respective segmentations were shown, followed by the next section, which includes the detection of crop lines.

# Chapter 5

# Conclusion

This chapter presents an overall assessment of the project, highlighting the main contributions and offering suggestions for future improvements in this segmentation and crop line detection project.

Throughout this project, a model was developed using the U-Net architecture for vineyard detection. Additionally, an algorithm based on the Hough Transform was created for detecting vineyard crop lines.

The first contribution of this project was the assembly of new datasets, which can be used in the future to improve this model or for the development of other projects. Finding suitable datasets was a particularly challenging task, and it was only made possible thanks to Beyond Vision, which provided datasets that contributed to the model's validation. Furthermore, the University of Coimbra made its datasets publicly available, which were used to train the model. Since these datasets already included real masks, the model training process was significantly expedited, as there was no need to manually label or classify the data.

The second significant contribution was the creation of the model that detects vineyards, specifically distinguishing between plant and non-plant areas. This detection has various potential applications, such as resource optimization. By identifying plants, vineyard managers can better assess the state of their vineyards. If large gaps in the vineyard rows are detected, they can allocate resources like water and fertilizers more effectively, reducing waste and associated costs. Additionally, the model contributes to increased sustainability by promoting more precise management practices. Accurate plant detection supports more sustainable viticulture, minimizing the environmental impact of wine production, as farmers gain a clearer understanding of their vineyard's needs. For instance, if significant gaps are identified, farmers can better estimate the quantities needed for treatments such as spraying, reducing unnecessary use of chemicals.

Finally, the third contribution was the integration of several algorithms to improve the segmentation of the datasets. This enhancement made the line detection process more accurate. Such detection has practical applications, especially in modern intensive vineyards, where it can be integrated with machine GPS systems to ensure precise navigation, helping to avoid significant losses for the farmer.

## 5.1 Future works

In this section, future work that could be developed, using this study as a starting point, will be discussed.

Dataset labeling is one area that could be improved. Creating new annotations for the datasets would allow for testing whether the model would achieve better results using the same segmentation method. Another possibility is the automation of mask correction, implementing automated techniques to improve segmentation masks during post-processing. This adjustment could eliminate errors and inconsistencies in the predictions, and deep learning methods for refining pixel boundaries in segmented regions could be explored.

Improving the model's accuracy and robustness is another focus. Although the results were satisfactory, performance can be enhanced in terms of segmentation accuracy and consistency. More advanced or hybrid architectures, such as attention networks, could increase the model's ability to capture more complex details in the vineyards, especially in challenging scenarios such as seasonal changes or soil variations.

Expanding the dataset would be a fundamental initiative to make the model more robust and generalizable. Including images of vineyards from different regions and under various climatic conditions would increase the diversity of the training data. Additionally, creating or obtaining new segmentation masks with more precise annotations would help improve metrics such as recall and IoU, thus raising the overall performance of the model.

Another avenue for future work is the integration with multispectral sensors. In addition to RGB images, the inclusion of multispectral or hyperspectral data could enhance the detection of physiological characteristics of the vineyards, such as vigor and water stress. Investigating how this integration with different sensor types could improve the efficiency and accuracy of agricultural monitoring is a promising field to explore.

Furthermore, the system developed can be adapted for application to other agricultural crops. Although the focus was on viticulture, future research could extend the model to crops such as intensive olive groves, adjusting segmentation and line detection as needed for specific agricultural contexts.

Finally, the development of an interface for end-users could make the system more accessible and practical for vineyard managers and agronomists. A software application that allows for the visualization of segmentation and line detection results, along with the generation of automated reports, would add significant value to the system and facilitate its use in the field.

# Bibliography


[1] https://beyond-vision.pt/.

[2] Simon Afonso. Utilização de algoritmos de visão computacional para diagnóstico de doenças da vinha. 2019.

[3] Yiannis Ampatzidis and Victor Partel. Uav-based high throughput phenotyping in citrus utilizing multispectral imaging and artificial intelligence. *Remote Sensing*, 11(4):410, 2019.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[5] Tiago Barros, Pedro Conde, Gil Gonçalves, Cristiano Premebida, Miguel Monteiro, Carla Sofia Santos Ferreira, and Urbano J Nunes. Multispectral vineyard segmentation: A deep learning comparison study. *Computers and Electronics in Agriculture*, 195:106782, 2022.

[6] Alessia Cogato, Franco Meggio, Cassandra Collins, and Francesco Marinello. Medium-resolution multispectral data from sentinel-2 to assess the damage and the recovery time of late frost on vineyards. *Remote Sensing*, 12(11):1896, 2020.

[7] Ana I De Castro, Francisco M Jiménez-Brenes, Jorge Torres-Sánchez, José M Peña, Irene Borra-Serrano, and Francisca López-Granados. 3-d characterization of vineyards using a novel uav imagery-based obia procedure for precision viticulture applications. *Remote Sensing*, 10(4):584, 2018.

[8] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[9] Mulham Fawakherji, Ali Youssef, Domenico Bloisi, Alberto Pretto, and Daniele Nardi. Crop and weeds classification for precision agriculture using context-independent pixel-wise segmentation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 146–152, 2019.

[10] Pierre-Antoine Ganaye, Michaël Sdika, and Hugues Benoit-Cattin. Semi-supervised learning for segmentation under semantic constraint. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image*

*Computing and Computer Assisted Intervention – MICCAI 2018*, pages 595–602, Cham, 2018. Springer International Publishing.

[11] José Miguel Guerrero, Gonzalo Pajares, Martín Montalvo, Juan Romeo, and María Guijarro. Support vector machines for crop/weeds identification in maize fields. *Expert Systems with Applications*, 39(12):11149–11155, 2012.

[12] Ronghua Ji and Lijun Qi. Crop-row detection algorithm based on random hough transformation. *Mathematical and Computer Modelling*, 54(3-4):1016–1020, 2011.

[13] Aleem Khaliq, Lorenzo Comba, Alessandro Biglia, Davide Ricauda Aimonino, Marcello Chiaberge, and Paolo Gay. Comparison of satellite and uav-based multispectral imagery for vineyard variability assessment. *Remote Sensing*, 11(4):436, 2019.

[14] Aleem Khaliq, Lorenzo Comba, Alessandro Biglia, Davide Ricauda Aimonino, Marcello Chiaberge, and Paolo Gay. Comparison of satellite and uav-based multispectral imagery for vineyard variability assessment. *Remote Sensing*, 11(4):436, 2019.

[15] Lucas Prado Osco, Mauro dos Santos De Arruda, José Marcato Junior, Neemias Buceli Da Silva, Ana Paula Marques Ramos, Érika Akemi Saito Moryia, Nilton Nobuhiro Imai, Danillo Roberto Pereira, José Eduardo Creste, Edson Takashi Matsubara, et al. A convolutional neural network approach for counting and geolocating citrus-trees in uav multispectral imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 160:97–106, 2020.

[16] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.*, 9:62–66, 1979.

[17] Yan Pang, Yeyin Shi, Shancheng Gao, Feng Jiang, Arun-Narenthiran Veeranampalayam-Sivakumar, Laura Thompson, Joe Luck, and Chao Liu. Improved crop row detection with deep neural network for early-season maize stand count in uav imagery. *Computers and Electronics in Agriculture*, 178:105766, 2020.

[18] Maria Pérez-Ortiz, JM Peña, Pedro Antonio Gutiérrez, Jorge Torres-Sánchez, César Hervás-Martínez, and Francisca López-Granados. A semi-supervised system for weed mapping in sunflower crops using unmanned aerial vehicles and a crop row detection method. *Applied Soft Computing*, 37:533–544, 2015.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[20] Jinya Su, Xiaoyong Zhu, Shihua Li, and Wen-Hua Chen. Ai meets uavs: a survey on ai empowered uav perception systems for precision agriculture. *Neurocomputing*, 2022.

[21] Wera Winterhalter, Freya Veronika Fleckenstein, Christian Dornhege, and Wolfram Burgard. Crop row detection on tiny plants with the pattern hough transform. *IEEE Robotics and Automation Letters*, 3(4):3394–3401, 2018.

[22] Yu Zhang, Wei Zhang, Xuefeng Li, Jinwen Ma, and Yilin Liu. Deep learning for identifying salt bodies in seismic images. *Journal of Applied Geophysics*, 179:104054, 2020.