



HOKKAIDO UNIVERSITY

Title	Development of machine vision-based autonomous navigation systems for agricultural vehicles
Author(s)	SAHA, Sristi
Degree Grantor	北海道大学
Degree Name	博士(農学)
Dissertation Number	甲第16091号
Issue Date	2024-09-25
DOI	https://doi.org/10.14943/doctoral.k16091
Doc URL	https://hdl.handle.net/2115/93391
Type	doctoral thesis
File Information	SAHA_Sristi.pdf



Development of machine vision-based autonomous navigation systems for agricultural vehicles

(マシンビジョンによる農用車両の自律走行
システムの開発)

Hokkaido University Graduate School of Agriculture
Frontiers in Production Sciences Doctor Course

Sristi Saha

Table of Contents

TABLE OF CONTENTS.....	II
ACKNOWLEDGMENTS	VI
LIST OF FIGURES	VII
LIST OF TABLES	XI
LIST OF SYMBOLS	XII
LIST OF ABBREVIATIONS	XV
CHAPTER 1. INTRODUCTION	1
1.1. RESEARCH BACKGROUND	1
1.1.1. Agricultural labor shortage in Japan	1
1.1.2. Automation in agriculture	2
1.1.3. Current navigation technologies and their limitations: RTK-GNSS and IMU	3
1.1.4. Advances in machine vision for autonomous navigation	4
1.1.4.1. Traditional computer vision techniques	5
1.1.4.2. Deep learning models and applications.....	6
1.2. RESEARCH OBJECTIVES	7
1.3. ORGANIZATION OF THESIS	9
CHAPTER 2. RESEARCH PLATFORM	10
2.1. HARDWARE	10
2.1.1. Overview of autonomous agricultural vehicle	10
2.1.1.1. Robot tractor.....	10
2.1.1.2. Robot electric vehicle.....	11
2.1.2. Features of the Vision Sensor (Camera)	12
2.1.3. Specifications of the Processing Computer	14
2.2. SOFTWARE	15
2.2.1. Python programming language	15
2.2.2. Computer Vision Annotation Tool (CVAT).....	17

2.2.3. YOLOv8 model.....	18
CHAPTER 3. VISION-BASED ROAD DETECTION SYSTEM FOR AN AUTONOMOUS TRACTOR	21
3.1. INTRODUCTION.....	21
3.2. SYSTEM DESIGN AND ARCHITECTURE.....	21
3.2.1. Camera setup.....	22
3.2.2. Preprocessing: Camera distortion correction	23
3.2.3. Road detection algorithm	26
3.2.3.1. Capturing the image frames and removing distortion from images	27
3.2.3.2. Image processing.....	28
3.2.3.3. Selecting the region of interest.....	34
3.2.3.4. Road edge detection: Sliding window method	36
3.2.3.5. Auto-adjustment of threshold values used in image processing	38
3.2.3.6. Displaying the final result	39
3.2.3.7. Calculation of lateral error	41
3.3. EXPERIMENTAL SETUP	42
3.3.1. Description of test field.....	43
3.3.2. Data collection method	44
3.4. RESULTS AND DISCUSSION	45
3.4.1. Real-time field testing - Unpaved Road.....	45
3.4.1.1. Daytime experiment.....	45
3.4.1.2. Nighttime experiment.....	48
3.4.2. Real-time field testing - Paved Road	50
3.5. CONCLUSION	53
CHAPTER 4. AUTONOMOUS NAVIGATION OF ELECTRIC VEHICLES IN VINEYARDS USING MACHINE VISION AND YOLOV8	54
4.1. INTRODUCTION.....	54
4.2. SYSTEM DESIGN AND ARCHITECTURE.....	54

4.2.1. Camera setup	55
4.2.2. Data acquisition.....	56
4.2.3. Preprocessing: Camera calibration.....	58
4.2.3.1. Fisheye lens calibration.....	58
4.2.3.2. Determining the co-relation between pixel dimensions and real-world distances	59
4.2.4. Dataset preparation.....	61
4.2.5. YOLOv8 model training	63
4.2.5.1. YOLOv8 model training on Vineyard Dataset I.....	63
4.2.5.2. YOLOv8 model training on Vineyard Dataset II.....	64
4.2.6. Path calculation algorithm	65
4.2.6.1. Capturing the image frames and removing distortion from images	65
4.2.6.2. Application of trained YOLOv8 model for detection and localization	66
4.2.6.3. Calculation of the reference path line	66
4.2.6.4. Calculation of lateral error	68
4.2.6.5. Calculation of heading error.....	70
4.2.6.6. Calculation of steering angle.....	71
4.2.7. Tree row end detection algorithm	72
4.3. EXPERIMENTAL SETUP	73
4.3.1. Description of the vineyard test environment	73
4.3.2. Data collection methods.....	75
4.4. RESULTS AND DISCUSSION	76
4.4.1. YOLOv8 model training results.....	76
4.4.2. Real-time field testing	78
4.4.2.1. Cloudy weather conditions.....	79
4.4.2.2. Variable weather conditions.....	81
4.4.2.3. Starting one meter ahead of the vine rows	83
4.4.2.4. Tree row end detection tests.....	85
4.5. CONCLUSION	88

CHAPTER 5. CONCLUSION	89
5.1. SUMMARY OF THE FINDINGS	89
5.2. CONTRIBUTIONS TO THE FIELD.....	91
5.3. FUTURE RESEARCH	92
REFERENCES.....	94

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my supervisor, Dr. Noboru Noguchi, for his continuous support and guidance throughout the development of this work. Additionally, I extend my heartfelt appreciation to Dr. Kazunobu Ishii and Dr. Ricardo Ospina for their invaluable assistance and support in this research.

I would also like to acknowledge the help and support from all the students and staff members of the Laboratory of Vehicle Robotics. I am particularly thankful to Dr. Tsuyoshi Morita and Dr. Yoshitomo Yamasaki for their constant help with the experiments.

I am immensely grateful to the Ministry of Education, Culture, Sports, Science and Technology for providing me with a scholarship during this course. I also appreciate the support from the student affairs section of the Graduate School of Agriculture for their continuous assistance and support during the doctoral course.

Finally, I would like to thank my family for their unwavering support throughout my studies as a doctoral student. Without their love and encouragement, it would have been impossible for me to continue living in a foreign country and pursue my doctoral course. Finally, I would like to thank all of my friends, both in India and Japan, for their continuous love and support. I could not have achieved this without you.

LIST OF FIGURES

Figure	Title	Page
2.1.	Kubota MR1000A Tractor : (a) Front View, (b) Side View	11
2.2.	Robot Electric Vehicle (EV)	12
2.3.	KODAK PIXPRO 4KVR360 camera : (a) camera body (b) 197° Ultra-Wide Mode (front lens)	13
2.4.	Images captured with the KODAK PIXPRO 4KVR360 camera using the fisheye lens : (a) Agricultural dirt road, (b) Vineyard environment	14
2.5.	YOLOv8 architecture	19
3.1.	Overview of the whole road detection system architecture	21
3.2.	Camera setup on the tractor	22
3.3.	9x6 grid chessboard used to find the camera intrinsic	25
3.4.	Removing camera distortion : (a) Image with distortion, (b) Image without any distortion	25
3.5.	Overview of the road detection algorithm step by step	26
3.6.	Flowchart illustrating the process of image processing under 'general use' condition	28
3.7.	HSV color space	30
3.8.	HSL color space	31
3.9.	Morphological transformation : (a) 'Erosion' operation, (b) 'Opening' operation	32
3.10.	Absence of the left edge in an unpaved dirt road inside the Hokkaido University experimental farm	37
3.11.	Flowchart showing the process of auto-adjusting the threshold values	39
3.12.	Display of the final result of road detection : (a) Unpaved (dirt) road, (b) Paved (asphalt) road	40
3.13.	Calculation of lateral error from the calculated centerline and tractor's position : (a) Manual calculation of real-world distances	41

Figure	Title	Page
	from the chessboard, (b) Lateral error estimation	
3.14.	Road edge data recorded by using RTK-GNSS : (a) Unpaved road (bird's eye view), (b) Paved road (bird's eye view), (c) Unpaved road edge data, (d) Paved road edge data	43
3.15.	Experimental results (daytime) obtained from experiments done on an unpaved road in the Hokkaido University experimental farm : (a), (b) Lateral error obtained from the machine vision system at two different speeds (3.6 km/h and 5.8 km/h respectively), (c), (d) Distance between the tractor's position and actual centerline at two different speeds (3.6 km/h and 5.8 km/h respectively)	46
3.16.	Experimental results (nighttime) were obtained from experiments done on an unpaved road in the Hokkaido University experimental farm : (a), (b) Lateral error obtained from the machine vision system at two different speeds (3.0 km/h and 5.0 km/h respectively), (c), (d) Distance between the tractor's position and actual centerline at two different speeds (3.0 km/h and 5.0 km/h respectively)	49
3.17.	Condition of the experimental paved road inside the Hokkaido University campus: Presence of shadows, potholes, and fallen leaves on the road surface	51
3.18.	Experimental results were obtained from experiments done on a paved road inside the Hokkaido University campus : (a) Lateral error obtained from the machine vision system at a speed of 3.2 km/h, (b) Distance between the tractor's position and the actual centerline	51
4.1.	Overview of the whole vineyard row navigation system architecture	55
4.2.	Robot Electric Vehicle : (a) Camera positioning, (b) Two types of computers integrated into the EV	56

Figure	Title	Page
4.3.	Variability in vineyard conditions across grape production seasons under different weather conditions	57
4.4.	Fisheye lens calibration : (a) The original image displaying curvature, (b) The image post-lens calibration, devoid of curvature	59
4.5.	Mapping pixel dimensions to real-world distances : (a) Illustration of the process for measuring pixel and real-world dimensions using markers, (b) Schematic representation of the ground coordinates with respect to the EV, (c) Results obtained after fitting models to the manually measured data	60
4.6.	Dataset preparation : (a) Vineyard Dataset I with a single annotated class, (b) Vineyard Dataset II with two annotated classes	61
4.7.	Schematic representation of key elements required for determining the reference path line	67
4.8.	Schematic representation of essential factors involved in the calculation of lateral error and heading error	70
4.9.	Tree row end detection and estimating the distance to the endpoint of the row	72
4.10.	Vineyard test area : (a) in bird's eye view, (b) in UTM Coordinates	74
4.11.	Experimental results visualization for Experiment 1: (a) Detected bounding boxes and (b) Calculated edge lines and reference path line	79
4.12.	Experimental Results for Experiment 1: (a) Comparison between the calculated lateral error and the actual distance of the EV from the centerline, (b) Visualization of the path traversed by the EV within path 10, and (c) Analysis of the heading error determined by the MV system during the experimental procedure	79
4.13.	Experimental results visualization for Experiment 2: (a) and (c) Detected bounding boxes under sunny and cloudy weather	81

Figure	Title	Page
	conditions respectively, (b) Calculated edge lines and reference path line during sunny and cloudy weather conditions respectively	
4.14.	Experimental Results for Experiment 2: (a) Comparison between the calculated lateral error and the actual distance of the EV from the centerline, (b) Visualization of the path traversed by the EV within path 7, and (c) Analysis of the heading error determined by the MV system during the experimental procedure	82
4.15.	Experimental results visualization for Experiment 3: (a) Detected bounding boxes and (b) Calculated edge lines and reference path line	83
4.16.	Experimental Results for Experiment 3: (a) Comparison between the calculated lateral error and the actual distance of the EV from the centerline, (b) Visualization of the path traversed by the EV within path 7, and (c) Analysis of the heading error determined by the MV system during the experimental procedure	84
4.17.	Experimental results obtained from stationary tests	85
4.18.	Experimental results: (a) Real-time classification of vine ends (class '0') and line posts (class '1') by the 'YOLOv8m-vine-classes' model, (b) Real-time distance measurement within path 7	85
4.19.	Comparison of experimental results obtained from driving tests within (a) path 10, and (b) path 7	86

LIST OF TABLES

Table No.	Title	Page
2.1.	Specifications of KODAK PIXPRO 4KVR360 (front lens)	13
2.2.	Specifications of the processing computer 14	14
4.1.	Features of Vineyard Dataset I and II	62
4.2.	Performance Metrics of YOLOv8 Training	77

LIST OF SYMBOLS

Symbol	Description	Unit
$\mathbf{P}$	Point in 3D space	--
$\mathbf{X}$	The coordinates of point P in the world reference frame	--
$\mathbf{X}_c$	The coordinate vector of point P in the camera reference frame	--
$\mathbf{R}$	The rotation matrix corresponding to the rotation vector, used to transform coordinates from the world reference frame to the camera reference frame	--
$\mathbf{T}$	The translational matrix, used to transform coordinates from the world reference frame to the camera reference frame	--
x, y, z	The three coordinates of the point P in the camera reference frame (components of $\mathbf{X}_c$)	--
$\mathbf{a}, \mathbf{b}$	The pinhole projection coordinates of point P	--
r	The radial distance of the point in the image plane from the optical center	--
θ	The angle corresponding to the radial distance r , used in fisheye lens distortion correction	Radian
θ_d	The distorted angle due to fisheye lens distortion	Radian
k_1, k_2, k_3, k_4	The distortion coefficients for the fisheye lens model	--
x', y'	The distorted coordinates after applying the fisheye distortion correction	--
u, v	The final pixel coordinates in the image plane	--
f_x, f_y	The focal lengths of the camera in the x and y directions, respectively	Pixel
α	Percentage of the image kept after undistorting the image	Percentage
c_x, c_y	The coordinates of the principal point (the point where the optical axis intersects the image plane) in the x and y directions, respectively	--

Symbol	Description	Unit
$\mathbf{K}$	3x3 floating point camera intrinsic matrix	--
$\mathbf{D}$	Output vector of distortion coefficient	--
n_a, n_b	Binary numbers	--
i	Index	--
$\&$	The bitwise AND operator	--
$ $	The bitwise OR operator	--
t_i	Value of the transformation matrix at index i	--
x'_i, y'_i	Coordinates of quadrangle vertices in the source image	--
x_i, y_i	Coordinates of the quadrangle vertices in the destination image	--
src	Coordinates of the quadrangle vertices in the source image	--
dst	Coordinates of the corresponding quadrangle vertices in the destination image	--
$g(x)$	Resulting blended image	--
α'	Blending factor	--
$f_0(x)$	Undistorted original image	--
$f_1(x)$	The image containing the detected features	--
δ	Steering angle	Degree
$\Delta\epsilon$	Lateral error calculated by MV system for autonomous navigation of robot tractor	--
$\Delta\phi_I$	Heading error provided by IMU for autonomous navigation of robot tractor	--
α_s, β_s	Proportional gains	--
m	Slope of regression line	--
b	Intercept of regression line	--
n	Number of edge points in the array	--
Σ	Summation symbol	--
x_e, y_e	Coordinates of the edge points	--

Symbol	Description	Unit
ix, iy	Coordinates of the intersection point	--
b_l, b_r	Intercepts of the left and right edge lines	--
p_b	Bottommost point of left or right edge lines	--
h	Height of the image frame	Pixel
d_{pix}	Lateral error calculated by MV system for autonomous navigation of robot EV	Pixel
w	Width of the image frame	Pixel
l	Look-ahead distance	Meter
x_{la}	Horizontal coordinate of the look-ahead point	--
d	Lateral error calculated by MV system for autonomous navigation of robot EV	Meter
X_a	Actual real-world width	Meter
X_s	The ratio of the actual real-world width X_a to the pixel width w	Meter per Pixel
$\Delta\phi$	Heading error calculated by MV system for autonomous navigation of robot EV	Degree
π	A mathematical constant that is the ratio of a circle's circumference to its diameter, approximately equal to 3.14159	--
P'	Model evaluation metrics: Precision	--
R'	Model evaluation metrics: Recall	--
n_c	Number of classes	--

LIST OF ABBREVIATIONS

Legend	Description
2D	Two-dimensional
3D	Three-dimensional
AP	Average precision
CMOS	Complementary Metal Oxide Semiconductors
COM	Communication
CVAT	Computer Vision Annotation Tool
DL	Deep learning
EfficientDet	Scalable and Efficient Object Detection
EIS	Electronic Image Stabilization
ELU	Exponential Linear Unit
EV	Electric Vehicle
FN	False Negatives
FPS	Frames per second
FP	False Positives
GNSS	Global Navigation Satellite System
HD	High Definition
HSL	Hue, saturation, lightness
HSV	Hue, saturation, value
Hz	Hertz
IMU	Inertial Measurement Unit
IoU	Intersection over Union
mAP	Mean Average Precision
MV	Machine vision
OpenCV	Open Source Computer Vision Library
PC	Personal computer
RAM	Random Access Memory
R-CNN	Region-Based—Convolutional Neural Network

Legend	Description
RGB	Red Green Blue
RMSE	Root Mean Square Error
ROI	Region of interest
RS	Recommended standard
RTK	Real-time Kinematic
SSD	Single Shot MultiBox Detector
TCP	Transmission Control Protocol
TP	True Positives
USB	Universal Serial Bus
UTM	Universal Transverse Mercator
YOLO	You Only Look Once

CHAPTER 1. INTRODUCTION

1.1. Research Background

1.1.1. Agricultural labor shortage in Japan

According to the Ministry of Agriculture, Forestry and Fisheries (2023), the number of core agricultural workers in Japan, defined as workers whose primary occupation is agriculture, has decreased by 17% over the past five years. Currently, the agricultural workforce comprises approximately 1.16 million individuals. Notably, 70.7% of these core agricultural workers are aged 65 years and above, with the average age of the agricultural working population in Japan being 68.7 years. These statistics highlight two critical issues: the severe lack of young people entering the agricultural sector and the aging of the existing workforce. Consequently, Japan's agricultural sector is experiencing a significant decline.

Vineyard operations, in particular, are highly labor-intensive, necessitating approximately 80 to 250 manual labor hours per acre for full production (McCorkle et al., 2019). This demand for labor is higher than that of many other fruit crops. The shortage of skilled labor exacerbates these challenges, adversely affecting production quality, increasing costs, and reducing global competitiveness.

If this trend persists, Japan's food self-sufficiency rate, which has already been declining, will deteriorate further, posing a serious threat to the future of Japanese agriculture. Addressing this labor shortage is therefore of paramount importance to ensure the sustainability and resilience of the nation's agricultural industry.

1.1.2. Automation in agriculture

Farm automation, often associated with ‘smart farming,’ involves the use of technology to automate various aspects of crop and livestock production, thereby enhancing efficiency and output. The continuous advancements in agricultural automation are aimed at addressing the multifaceted challenges posed by population growth, accelerated urbanization, high competitiveness of high-quality products, environmental preservation, and a diminishing labor force. Recently, automation in agriculture has become a prominent trend, with operations such as cultivation, inspection, spraying, pruning, and harvesting being executed autonomously to mitigate labor shortages (Azimi et al., 2020). Thus, in addressing Japan’s agricultural issues, agricultural automation is anticipated to become an essential solution.

Agricultural robots have the potential to become a benchmark technology for sustaining human society, particularly when systems are appropriately designed to handle unpredictable agricultural environments. These robots can operate continuously throughout the day, be programmed for various tasks, have their efficiency enhanced through optimized algorithms, and prove to be economically efficient in the long term (Roshanianfard et al., 2020).

Several research groups globally have achieved significant advances in agricultural automation (Oliveira et al., 2021). For instance, autonomous tractors capable of operating in open fields for various agricultural tasks have been developed (Barawid et al., 2007; Kise et al., 2001; Noguchi et al., 2001; Takai et al., 2014). Mendez et al. (2023) also highlighted the development of precision agriculture to meet the demands of a growing population by enhancing productivity and reducing manual labor through task automation. This includes the creation of autonomous navigation

algorithms for tasks such as spraying, pruning, fruit picking, weed removal, and field monitoring, which demonstrate the potential of robotic technology to revolutionize vineyard efficiency in the long term. All these research endeavors utilize RTK-GNSS as a primary navigation sensor, often coupled with an IMU (Inertial Measurement Unit).

1.1.3. Current navigation technologies and their limitations: RTK-GNSS and IMU

Gan and Lee (2018) categorize the essential abilities of autonomous agricultural robots into four groups: navigation, detection, action, and mapping. Recognized as the primary step, navigation is of utmost importance and often involves detection and mapping, particularly in challenging environments such as vineyards.

For autonomous navigation, navigation sensors are crucial as they significantly influence the accuracy of the entire navigation system. Precise vehicle location, combined with maps and other navigation sensors, is necessary for effective autonomous driving. GNSS technology, particularly RTK-GNSS, has been integral to the evolution of autonomous driving. Early GNSS systems suffered from low accuracy, limited availability, and a lack of integrity. Despite these limitations, the ability to globally localize on a map was valuable. Over the past 20 years, significant advancements in GNSS technology have led to the development of high-precision GNSS, such as RTK, which can achieve centimeter-level accuracy (Joubert et al., 2020). However, RTK-GNSS still faces several usability challenges.

RTK-GNSS offers two types of solutions: float and fix. The float solution, requiring at least four common satellites, provides an accuracy range of about 20 cm to 1 m. The fixed solution, requiring at least five common satellites, offers accuracy within 5 cm. However, in the presence of tall obstacles like buildings and trees, the RTK fix solution often reverts to the float solution, leading to data jumps (Morales and

Tsubouchi, 2007). Moreover, RTK-GNSS relies on a fast data communication link, which necessitates good internet connectivity. Given that 70% of Japan's land area is mountainous and hilly, with extensive tree canopies in many rural farm areas, maintaining measurement accuracy under such conditions is challenging.

In addition to the limitations of RTK-GNSS, the accuracy of IMUs cannot be guaranteed over extended periods due to drift caused by temperature variations and integration errors. This drift leads to a degradation of IMU performance over time, further complicating the use of IMUs as standalone navigation sensors in autonomous agricultural systems.

In summary, while RTK-GNSS and IMUs are critical components of current navigation technologies, their limitations in terms of accuracy and reliability in certain environments necessitate the development of alternative or supplementary navigation solutions to ensure robust and precise autonomous navigation in agriculture.

1.1.4. Advances in machine vision for autonomous navigation

Machine vision has emerged as a promising technology for enhancing autonomous navigation systems in various fields, including agriculture. By leveraging the power of visual data processing and analysis, machine vision enables autonomous vehicles to perceive and interpret their surroundings, thus facilitating precise navigation without relying solely on external positioning systems like GNSS. In recent years, significant advancements have been made in machine vision techniques and algorithms, contributing to the development of more robust and efficient autonomous navigation systems. This section explores the latest advances in machine vision technology and their applications in the context of autonomous navigation, highlighting their potential

to overcome the limitations of traditional navigation sensors and improve the autonomy and reliability of agricultural vehicles.

1.1.4.1. Traditional computer vision techniques

Various systems based on different sensors have been developed for road surface detection, primarily focusing on paved, well-marked, and well-maintained roads. Among these sensors, monocular vision sensors are widely utilized due to their ability to provide rich information about driving scenes (Sivaraman and Trivedi, 2013). Additionally, when coupled with traditional image processing techniques, vision sensors can often offer higher accuracy in solving specific problems. Algorithms such as color-thresholding and pixel counting are generalizable and exhibit consistent performance across different images (O'Mahony et al., 2020). Using vision sensors, researchers have developed autonomous driving systems capable of tracking lane markings on highways and urban roads (Schneiderman and Nashman, 1992; Wang et al., 2002; Chiu and Lin, 2005; Nguyen et al., 2017).

However, the context of agricultural environments, particularly for tractor navigation, poses distinct challenges. Tractors predominantly operate on unstructured roads, such as rural paths, which deviate from standard road characteristics. These roads may lack lane markings, exhibit spatially fuzzy edges with low-intensity contrast, and feature irregular shapes with varying surface materials, including mud, clay, sand, gravel, and asphalt (Hu et al., 2004). Despite these complexities, vision sensors can still extract valuable features from the road environment. Road shape, area, color, lane markings (if present), as well as surrounding features like grass or shrubs at the road edges, serve as potential identifiers for road detection.

In vineyard environments, researchers have explored the feasibility of utilizing machine vision (MV) techniques for navigation tasks. Sharifi and Chen (2015) demonstrated the use of MV by categorizing Red, Green, Blue (RGB) images captured by a mobile robot in an orchard row. They applied graph partitioning theory and the Hough transform to identify the central path for navigation. Similarly, Mendez et al. (2023) employed MV techniques, including Otsu's method, blob detection, and pixel counting, to locate the center of rows in vineyards. These studies highlight the potential of traditional computer vision techniques in addressing navigation challenges in agricultural settings, paving the way for further advancements in autonomous agricultural vehicles.

1.1.4.2. Deep learning models and applications

Traditional machine vision (MV) techniques, while effective in many contexts, often struggle with challenges such as light variations and the dynamic conditions found in vineyard environments. In contrast, deep learning (DL) methods leverage deep neural networks to extract high-level features, thereby offering superior accuracy in tasks such as image classification, semantic segmentation, and object detection. However, these methods typically require substantial computing resources (Wang et al., 2023).

Hybrid approaches that combine traditional MV techniques with DL offer a practical solution, providing the benefits of both methodologies. These hybrid systems are particularly advantageous in high-performance applications where rapid implementation is crucial (O'Mahony et al., 2020). By integrating traditional MV techniques with the advanced capabilities of DL, these systems can more effectively address the unique challenges posed by vineyard environments.

The advancements in DL models, such as convolutional neural networks (CNNs) and more recent architectures like YOLOv8, have shown significant promise in enhancing automation efficiency in agricultural settings. These models are capable of processing vast amounts of data, learning complex patterns, and making precise predictions, which are essential for tasks such as autonomous navigation, crop monitoring, and yield estimation.

The integration of traditional MV and DL methods highlights a synergistic approach to overcoming the limitations of each technique. Traditional MV provides robust, real-time processing capabilities, while DL offers high accuracy and adaptability to varying conditions. Together, they represent a comprehensive strategy for improving the performance and reliability of autonomous agricultural systems, thereby advancing the field of agricultural automation and contributing to more sustainable farming practices.

1.2. Research Objectives

The first objective of this research is to develop a vision-based road detection system for autonomous tractors. As discussed in section 1.1.3, RTK-GNSS is a primary navigation sensor in autonomous systems, but it has usability issues, particularly in environments with tall obstacles that interfere with signal reception. An autonomous tractor must not only perform various farm operations but also travel autonomously between different locations, such as from the farmer's house or warehouse to the farm, and potentially between multiple farms. Achieving this capability is crucial for full autonomy. Reliable and unobstructed data transmission is critical for the safe navigation of autonomous tractors. Interruptions in RTK-GNSS signals can cause deviations from the predetermined path, posing safety risks. Hence, there is a need for a

backup navigation method to prevent unsafe situations when RTK-GNSS is unavailable. As elaborated in section 1.1.4, machine vision can be employed for autonomous vehicle navigation. This research aims to develop a vision-based road detection system capable of detecting both paved and unpaved roads, whether structured or unstructured. The objective is to create a system that can seamlessly transition between different types of roads without requiring the tractor to stop, ensuring continuous and safe navigation.

The second objective of this research is to integrate traditional machine vision (MV) techniques with deep learning (DL) approaches to develop a robust model for vineyard navigation. As highlighted in section 1.1.4.2, hybrid approaches that combine traditional MV techniques with DL offer a practical solution, and they are particularly advantageous in high-performance applications where rapid implementation is crucial. This study aims to create a model capable of detecting vine trunk ends and calculating the necessary navigation parameters to guide a robot EV through a vine row from start to end. The YOLOv8 model has been selected as the base detection model due to its high performance and versatility, as discussed in section 1.1.4.2. YOLOv8 offers real-time object detection with high accuracy and efficient resource utilization, making it suitable for the demands of this research (Jocher et al., 2023). YOLOv8's customizable and user-friendly framework allows for the fine-tuning of models to meet specific dataset characteristics and detection requirements. This research will leverage these capabilities to ensure seamless integration into the navigation system.

Additionally, the research aims to accurately identify the termination point of a vine row, enabling precise stopping of the robotic EV. This feature is essential for facilitating subsequent operations, such as moving the EV to another row, thereby enhancing the overall efficiency of vineyard management.

1.3. Organization of Thesis

The thesis is organized as follows:

Chapter 1 introduces the research background and objectives. It discusses the agricultural labor shortage in Japan, the potential of automation, and advancements in machine vision and deep learning.

Chapter 2 presents the research platform, describing the hardware components, such as autonomous agricultural vehicles and vision sensors, and the software tools, including Python, computer vision tools, and the YOLOv8 model.

Chapter 3 details the vision-based road detection system for an autonomous tractor. It covers the system's design, image preprocessing techniques, and road detection algorithms, as well as results from real-time field testing.

Chapter 4 describes the machine vision and deep learning-based navigation system for electric vehicles in vineyards. It discusses dataset preparation, YOLOv8 model training, path calculation, and tree row end detection algorithms, along with the field-testing results.

Chapter 5 summarizes the study's findings, contributions, implications, and future work.

CHAPTER 2. RESEARCH PLATFORM

2.1. Hardware

This section provides a detailed overview of the hardware components utilized in the research experiments. It covers the key elements of the research platform, including the autonomous agricultural vehicles, vision sensors, and processing computers. Each component's specifications and functionalities are described to establish a comprehensive understanding of the hardware setup employed throughout the study.

2.1.1. Overview of autonomous agricultural vehicle

This subsection presents an in-depth overview of the autonomous agricultural vehicles utilized in the research. It outlines the characteristics and features of the selected vehicle models, highlighting essential specifications such as engine power, drive type, and dimensions. By providing detailed insights into the design and capabilities of these vehicles, this section lays the groundwork for understanding their role in the experimental setup and their contributions to the research objectives.

2.1.1.1. Robot tractor

The tractor selected for this research belongs to the KUBOTA MR1000A series, with the specific model identified as MR1000AQMAXWUPC3-A (Fig. 2.1). It features a half-track design with four-wheel drive capabilities and is equipped with a 73.5 kW engine. The tractor's dimensions are as follows: length of 4.505 m, width of 2.86 m (1.98 m when the sensor stay is retracted), and height of 2.725 m (2.625 m when the antenna is retracted). A camera is securely mounted on the top of the tractor, positioned at an approximate height of 2.7 meters from the ground following installation.



Figure 2.1. Kubota MR1000A Tractor : (a) Front View, (b) Side View

2.1.1.2. Robot electric vehicle

Numerous companies have introduced EVs tailored for agricultural purposes, and ongoing research in electric agricultural robots is anticipated to advance further (Deere and Company, 2024; Kubota Corp., 2024; Naïo Technologies, 2024).

For this study, a specialized robot EV optimized for agricultural tasks is employed (Fig. 2.2). Derived from a golf cart base, the EV platform is meticulously modified to suit vineyard-specific requirements. Its powertrain incorporates a battery and electric motor sourced from a hybrid car model (Toyota Aqua Hybrid, Japan), ensuring robust power delivery and operational efficiency in field environments (Yamasaki et al., 2021).



Figure 2.2. Robot Electric Vehicle (EV)

2.1.2. Features of the Vision Sensor (Camera)

In this research, an RGB camera (PixPro ORBIT360, Eastman Kodak Company, USA) (Fig. 2.3) is used to capture HD RGB images. This camera features two lenses, with the front lens being used in this research. The specifications of this lens are detailed in Table 2.1.

As shown in Table 2.1, the camera offers multiple resolution options. For this research, resolutions of 1280x720 pixels and 1920x1080 pixels are employed. The camera features electronic image stabilization (EIS), making it robust against vehicle vibrations and consistently producing stable image frames. It operates reliably across a wide range of temperatures and humidity levels, making it suitable for various weather conditions. The camera connects to the processing computer via a USB cable. Images captured with this camera introduce a certain amount of distortion due to its fisheye lens (Fig. 2.4).

Table 2.1

Specifications of KODAK PIXPRO 4KVR360 (front lens)

Specifications	Value
Effective image sensor pixels	20.68 Megapixels
Total image sensor pixels	21.14 Megapixels (1/2.3" BSI CMOS)
Focal length	1.633 mm
F-number	F2.4
Focus	Fix focus
Image stabilization	Electronic Image Stabilization (EIS)
Number of recording pixels	197° 4K Ultra-Wide Mode (front lens)
	(16:9) 3840x2160/30 FPS
	(16:9) 1920x1080/60 FPS
Operating environment	(16:9) 1280x720/60 FPS
	Temperature: -10° – 40°C
ISO sensitivity	Humidity: 0 – 90%
	Auto (100 – 1600) (Video)
USB port	USB 2.0

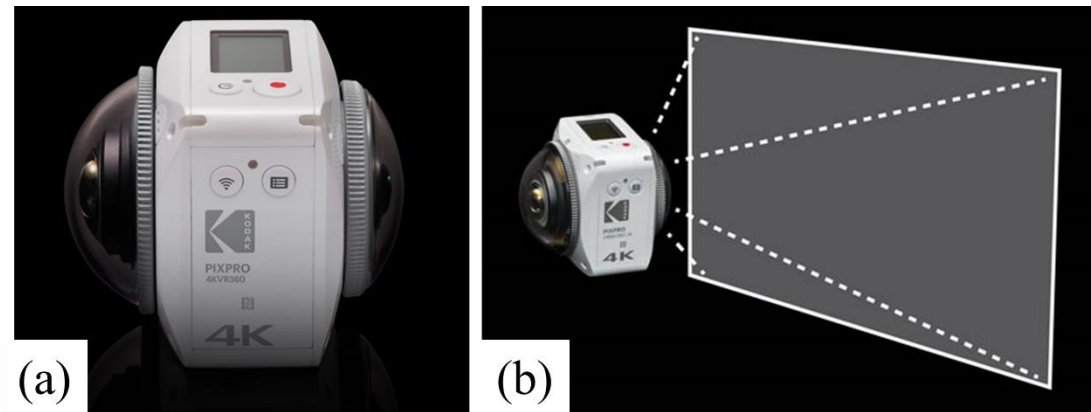


Figure 2.3. KODAK PIXPRO 4KVR360 camera : (a) camera body (b) 197° Ultra-Wide Mode (front lens)

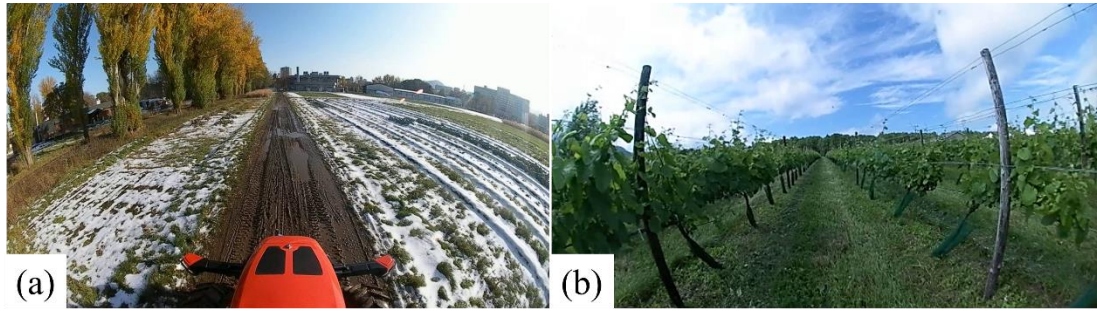


Figure 2.4. Images captured with the KODAK PIXPRO 4KVR360 camera using the fisheye lens : (a) Agricultural dirt road, (b) Vineyard environment

2.1.3. Specifications of the Processing Computer

For image processing, a computer system (GS66 Stealth, MSI Co. Ltd., Taiwan) equipped with an Intel Core i9-10980HK processor, 16 GB RAM, a dedicated graphics card (GeForce RTX 2070 Max-Q, NVIDIA Corp., USA) featuring 8 GB GDDR6 RAM, and the Windows 11 (64-bit) operating system was employed. The specifications of both computers are shown in Table 2.2.

Table 2.2

Specifications of the processing computer

Component	Specifications
Model	GS66 Stealth, MSI Co. Ltd., Taiwan
Processor	Intel(R) Core (TM) i9-10980HK CPU @2.40GHz
Installed RAM	16.0 GB
System type	64-bit operating system
Windows edition	Windows 11 pro
Windows version	23H2
Graphics card	GeForce RTX 2070 Max-Q, NVIDIA Corp., USA
Graphics RAM type	GDDR6
Graphics card RAM size	8 GB

The computer plays a crucial role in this research as it handles all image processing tasks. The high-performance specifications ensure efficient and reliable processing, providing good performance in terms of frames per second (FPS), which is essential for real-time applications.

2.2. Software

This section provides an overview of the software components utilized in the research, detailing the tools and technologies implemented for image processing and autonomous navigation. The software platform is crucial for the effective operation of the vision-based road detection system and the YOLOv8 model for vineyard navigation. Key elements include the Python programming language, selected for its versatility and extensive library support, and the Computer Vision Annotation Tool (CVAT) for dataset preparation and annotation. Additionally, the YOLOv8 model, known for its high performance in object detection. This section outlines the integration of these software components, explaining their roles and interconnections in the overall system architecture.

2.2.1. Python programming language

The programming language used for processing image frames in this research is Python. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It is versatile and applicable in various domains, including data analysis, mathematical computation, and software development. For this study, Python versions 3.7 (Python Software Foundation, 2019) and 3.10 (Python Software Foundation, 2021) are employed within the Anaconda distribution (Anaconda Software Distribution, 2020). Anaconda is a distribution of the Python and R programming

languages designed for scientific computing, simplifying package management and deployment.

Python libraries play a crucial role in developing applications for machine learning, data science, data visualization, and image and data manipulation. The primary libraries used in this research include OpenCV, NumPy, PyTorch, and Joblib:

OpenCV (Open Source Computer Vision Library). A comprehensive library designed for real-time computer vision and image processing. It provides a vast range of functionalities for image manipulation, including filtering, transformation, and feature detection, which are essential for the image processing tasks in this study.

NumPy. A fundamental library for numerical computing in Python. It offers support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is integral for handling and processing the numerical data derived from images.

PyTorch. An open-source machine learning library based on the Torch library. It is used for applications such as computer vision and natural language processing. PyTorch provides robust support for implementing and training deep learning models, which is critical for the development of the YOLOv8 model used in this research.

Joblib. A library for efficient serialization and deserialization of Python objects, primarily used for saving and loading machine learning models and data processing pipelines. It supports transparent disk-caching of functions and objects, enabling efficient handling of large datasets.

These libraries collectively facilitate the development and execution of the image processing and machine learning tasks required for the vision-based road detection system and autonomous navigation in vineyard environments.

2.2.2. Computer Vision Annotation Tool (CVAT)

The Computer Vision Annotation Tool (CVAT) is a free, open-source, web-based tool designed for image and video annotation, primarily used to label data for computer vision algorithms (CVAT.ai Corporation, 2022). Initially developed by Intel, CVAT is tailored for professional data annotation teams, featuring a user interface optimized for various computer vision annotation tasks. CVAT supports the key tasks essential for supervised machine learning: object detection, image classification, and image segmentation. Users can create detailed annotations for each of these tasks, making it a versatile tool for preparing datasets for training machine learning models.

CVAT includes a range of powerful features to enhance the annotation process. One of its notable features is the interpolation of shapes, which allows for the smooth transition of shapes between key frames, reducing the manual effort required for video annotation. Additionally, it offers semi-automatic annotation capabilities by utilizing deep learning models to assist in the annotation process, providing suggestions and automating repetitive tasks. This can significantly expedite the labeling process and improve accuracy. The tool also includes shortcuts for critical actions, offering keyboard shortcuts for the most frequent actions, streamlining the workflow, and increasing efficiency. Furthermore, CVAT features a comprehensive dashboard listing all annotation projects and tasks, enabling easy management and tracking of progress.

These capabilities make CVAT an indispensable tool for generating high-quality annotated datasets, crucial for the development and training of accurate computer vision models. Its integration into this research ensures that the data labeling process is efficient and effective, supporting the broader objectives of image processing and machine learning applications in agricultural automation.

2.2.3. YOLOv8 model

You Only Look Once (Version 8.0.0) or YOLOv8 (Jocher et al., 2023) was selected as the base detection model for this research due to its exceptional performance and versatility, offering real-time object detection with high accuracy while utilizing resources efficiently. The YOLOv8 framework is user-friendly, allowing for the customization and fine-tuning of models to suit specific dataset characteristics and detection requirements, ensuring seamless integration into the research pipeline.

YOLOv8 introduces several significant advancements over its predecessors, including YOLOv5. The new backbone architecture, CSPDarknet-AA, is an advanced version of the CSPDarknet series, known for its efficiency and performance in object detection tasks. This architecture enhances feature extraction capabilities, contributing to improved detection accuracy and speed.

A key technique incorporated in YOLOv8 is multi-scale object detection, enabling the model to detect objects of various sizes within an image. This is particularly beneficial in agricultural settings where object sizes, such as plant features, can vary significantly. Additionally, YOLOv8 utilizes the ELU (Exponential Linear Unit) activation function, which accelerates learning in deep neural networks by mitigating the vanishing gradient problem, leading to faster convergence and more efficient training.

The adoption of GIoU (Generalized Intersection over Union) loss in YOLOv8 marks another significant enhancement. GIoU is a more advanced version of the IoU metric that accounts for the shape and size of bounding boxes, thereby improving the

precision of object localization. This is crucial in applications where accurate object boundary detection is required.

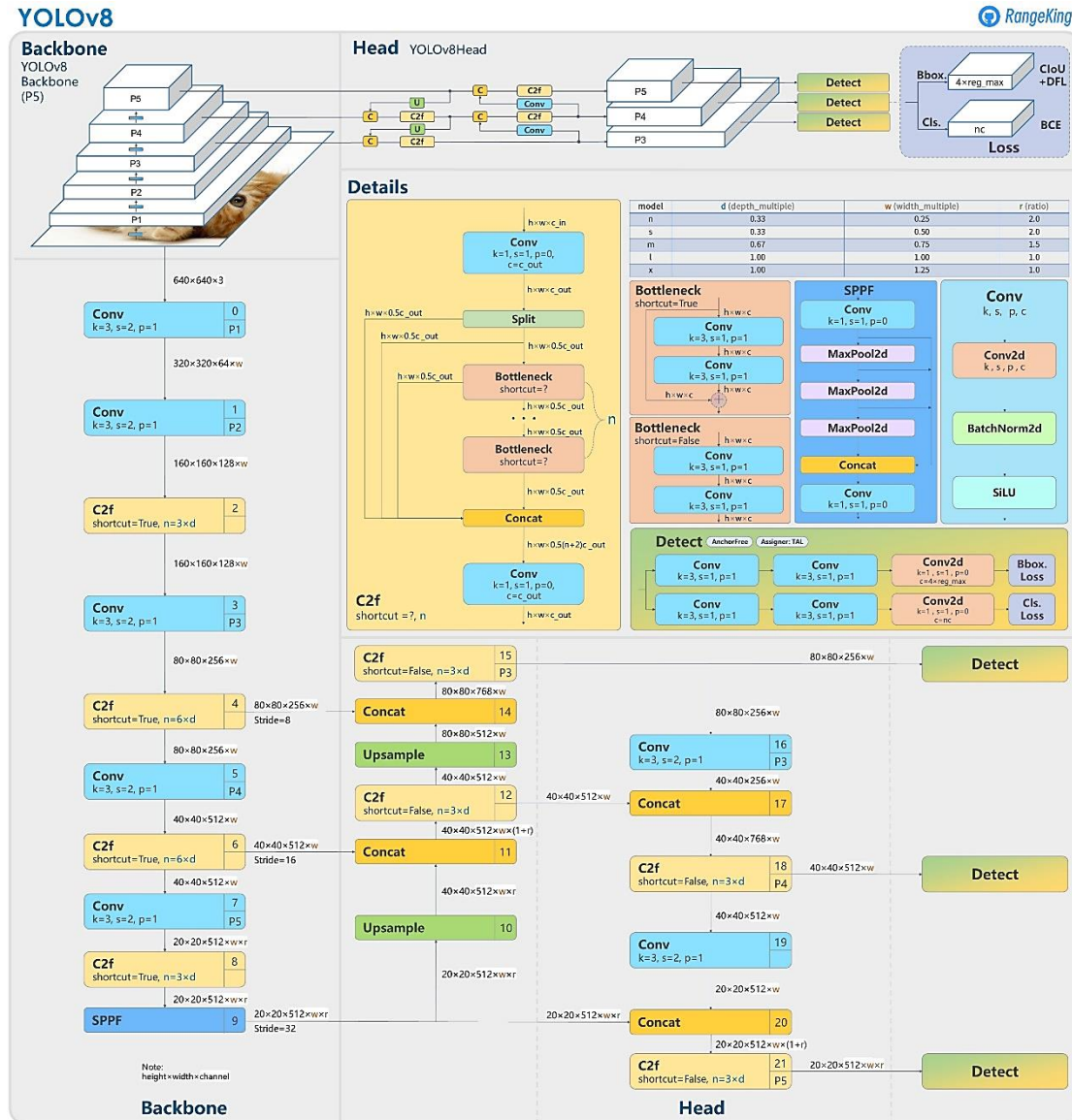


Figure 2.5. YOLOv8 architecture (King, R., 2023)

YOLOv8 achieves a 1.2% increase in average precision (AP) while reducing the model's weight file size by 80.6 MB compared to YOLOv7. This enhanced performance and reduced size make YOLOv8 more efficient and easier to deploy in

resource-constrained environments. YOLOv5, while still effective and widely used, lacks some of the advanced features present in YOLOv8, such as the improved backbone architecture and the use of ELU activation functions.

When comparing YOLOv8 to other deep learning models used in machine vision, such as Faster R-CNN (Region-Based—Convolutional Neural Network), SSD (Single Shot MultiBox Detector), and EfficientDet (Scalable and Efficient Object Detection), several distinctions emerge. Faster R-CNN is known for its high accuracy but suffers from slower inference times, making it less suitable for real-time applications. SSD offers real-time performance but generally with lower accuracy compared to YOLO models. EfficientDet, while efficient and accurate, involves a more complex architecture and requires more tuning for optimal performance.

YOLOv8's primary advantage over these models lies in its balance of speed and accuracy. It provides real-time object detection with high precision, making it suitable for dynamic and resource-constrained environments like agriculture. YOLOv8's design allows for easier customization and integration, which is essential for tailoring the model to specific agricultural tasks, such as detecting vine trunks.

Despite its generalist design aimed at broad applicability, YOLOv8's architecture allows for customization to meet specific task requirements. This adaptability is crucial for achieving high-quality, high-speed detection in non-standard scenarios, such as those encountered in agricultural environments. By customizing the YOLOv8 model to suit the specific needs of this research, the study aims to harness the full potential of this state-of-the-art object detection algorithm for agricultural automation.

CHAPTER 3. VISION-BASED ROAD DETECTION SYSTEM FOR AN AUTONOMOUS TRACTOR

3.1. Introduction

Autonomous tractors traditionally rely on RTK-GNSS to navigate open fields with high precision. However, the availability of RTK-GNSS signals can be inconsistent across different geographical locations, potentially compromising the safety and reliability of autonomous operations. To address this limitation, this research focuses on developing a vision-based agricultural road detection system that operates independently of RTK-GNSS signals. This chapter delves into the core of image processing algorithms utilized in this research, providing an in-depth exploration of their implementation, optimization, and application within the context of autonomous navigation in agricultural settings.

3.2. System Design and Architecture

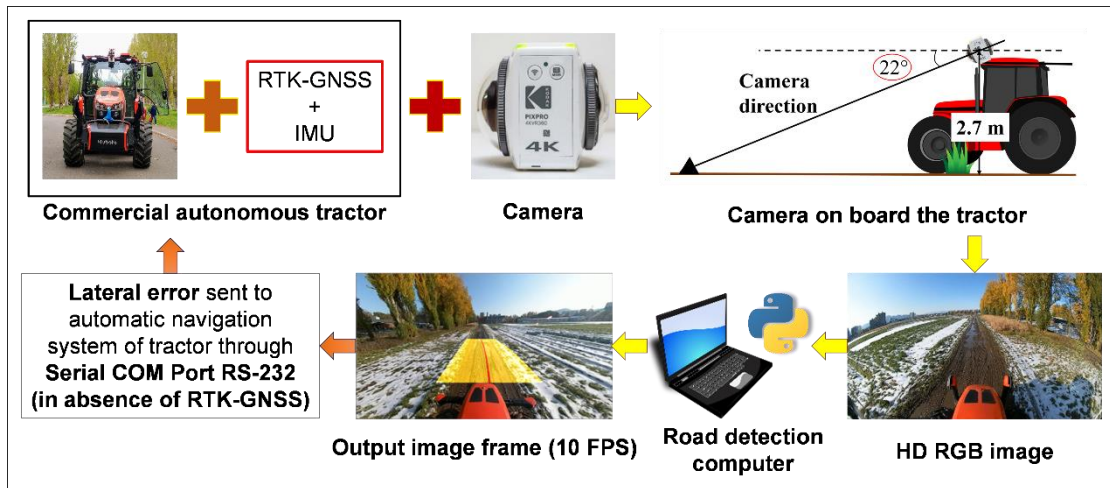


Figure 3.1. Overview of the whole road detection system architecture

In Figure 3.1, we present an overview of the system architecture, illustrating the integration and interaction of various components crucial for the autonomous

navigation of an agricultural tractor. A comprehensive explanation of the camera, tractor, processing computer, and programming language can be found in Chapter 2. The system is developed to process high-definition visual data captured by cameras mounted on the tractor, employing a series of image processing techniques to identify and classify key features within the image frame. The output generated by the road detection algorithm is then relayed to the tractor's automatic navigation system via the serial COM (communication) port RS (Recommended standard) - 232, with an average update rate of 10 FPS or 10 Hz.

Subsequent sections delve into these specific image processing techniques, detailing the selection criteria for each method and the unique challenges they address.

3.2.1. Camera setup

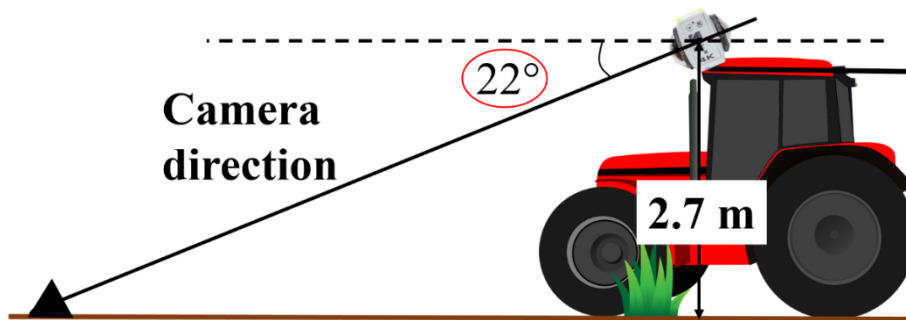


Figure 3.2. Camera setup on the tractor

In this study, the camera utilized for image capture, as detailed in section 2.1.2, is employed for data acquisition purposes. Positioned atop the tractor, the camera is situated at a height of 2.7 meters from the ground, offering an optimal vantage point for comprehensive environmental observation. Additionally, the camera is horizontally tilted at an angle of 22 degrees (Fig. 3.2), ensuring an unobstructed view ahead, with a

clear focus on the surrounding terrain while minimizing the presence of extraneous elements such as the sky or tractor hood within the captured image frame.

3.2.2. Preprocessing: Camera distortion correction

As mentioned in section 2.1.2, the fisheye lens of the KODAK PIXPRO camera introduces distortion, necessitating correction for accurate calculation of 3D points from images. OpenCV's 'fisheye' package is utilized for calibration, refining the camera matrix, and ultimately, undistorting the images. The distortion correction process involves three major steps:

1. Perform camera calibration to obtain the intrinsic camera parameters, including distortion parameters.
2. Refine the camera matrix to control the percentage of unwanted camera pixels in the undistorted image.
3. Apply the refined camera matrix to undistort the image.

The calibration procedure is represented mathematically below-

Let P be a point in 3D of coordinates X in the world reference frame (stored in the matrix X_c). The coordinate vector of P in the camera reference frame is shown by equation 3.1.

$$X_c = RX + T \quad (3.1)$$

Where R is the rotation matrix corresponding to the rotation vector and T is the translational matrix; and x, y, z are the 3 coordinates of X_c :

$$x = X_{c_1} \quad (3.2)$$

$$y = X_{c_2} \quad (3.3)$$

$$z = X_{c_3} \quad (3.4)$$

The pinhole projection coordinates of P is (a, b) , where

$$a = x/z \quad (3.5)$$

$$b = y/z \quad (3.6)$$

$$r^2 = a^2 + b^2 \quad (3.7)$$

$$\theta = \text{atan}(r) \quad (3.8)$$

Fisheye distortion:

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \quad (3.9)$$

The distorted point coordinates are (x', y') , where

$$x' = (\theta_d/r)a \quad (3.10)$$

$$y' = (\theta_d/r)b \quad (3.11)$$

Finally, conversion into pixel coordinates: the final pixel coordinates vector (u, v) , where

$$u = f_x(x' + \alpha y') + c_x \quad (3.12)$$

$$v = f_y y' + c_y \quad (3.13)$$

Here f_x and f_y are the effective focal lengths of the imaging system measured in units of detector pixels in the horizontal and vertical directions and are expected to be equal for square pixels and non-anamorphic optics. c_x and c_y are the pixel coordinates of the center of the perspective projection on the sensor, expected to be close to the detector center. α is the percentage of the image that we want to keep after undistorting the image.

By finding two parameters **K** & **D** with the help of the ‘fisheye’ package and a chessboard (Fig. 3.3), the above procedure can be performed. As shown in equation 3.15, **K** is the output 3x3 floating-point camera intrinsic matrix. **D** is the output vector of distortion coefficients (k_1, k_2, k_3, k_4) .

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 1 & 1 \end{bmatrix} \quad (3.14)$$

$$\mathbf{D} = (k_1, k_2, k_3, k_4) \quad (3.15)$$

After finding the camera intrinsic parameters, $\mathbf{K}$ & $\mathbf{D}$ is applied to the original image with distortion (Fig. 3.4a), and the undistorted image (Fig. 3.4b) is obtained. The effectiveness of the distortion correction is visually evident, as depicted in Figure 3.4b, where the undistorted image presents a straight horizon and road, contrasting with the distorted image's curvilinear appearance. This distortion correction process is essential for ensuring the accuracy of subsequent image processing tasks, contributing to the overall reliability of the system architecture.



Figure 3.3. 9x6 grid chessboard used to find the camera intrinsic parameters

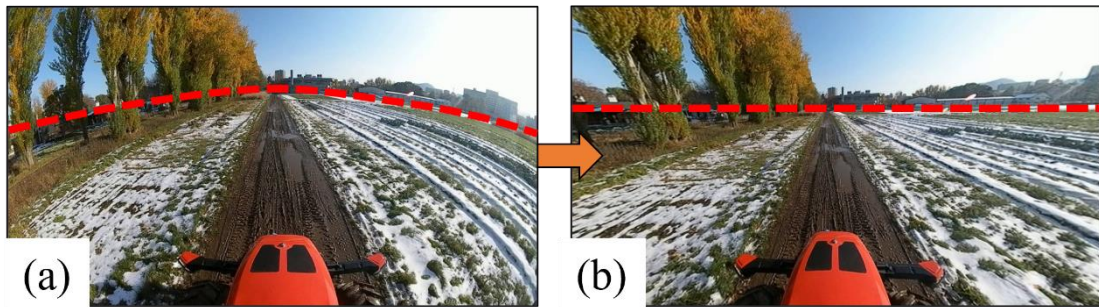


Figure 3.4. Removing camera distortion : (a) Image with distortion, (b) Image without any distortion

3.2.3. Road detection algorithm

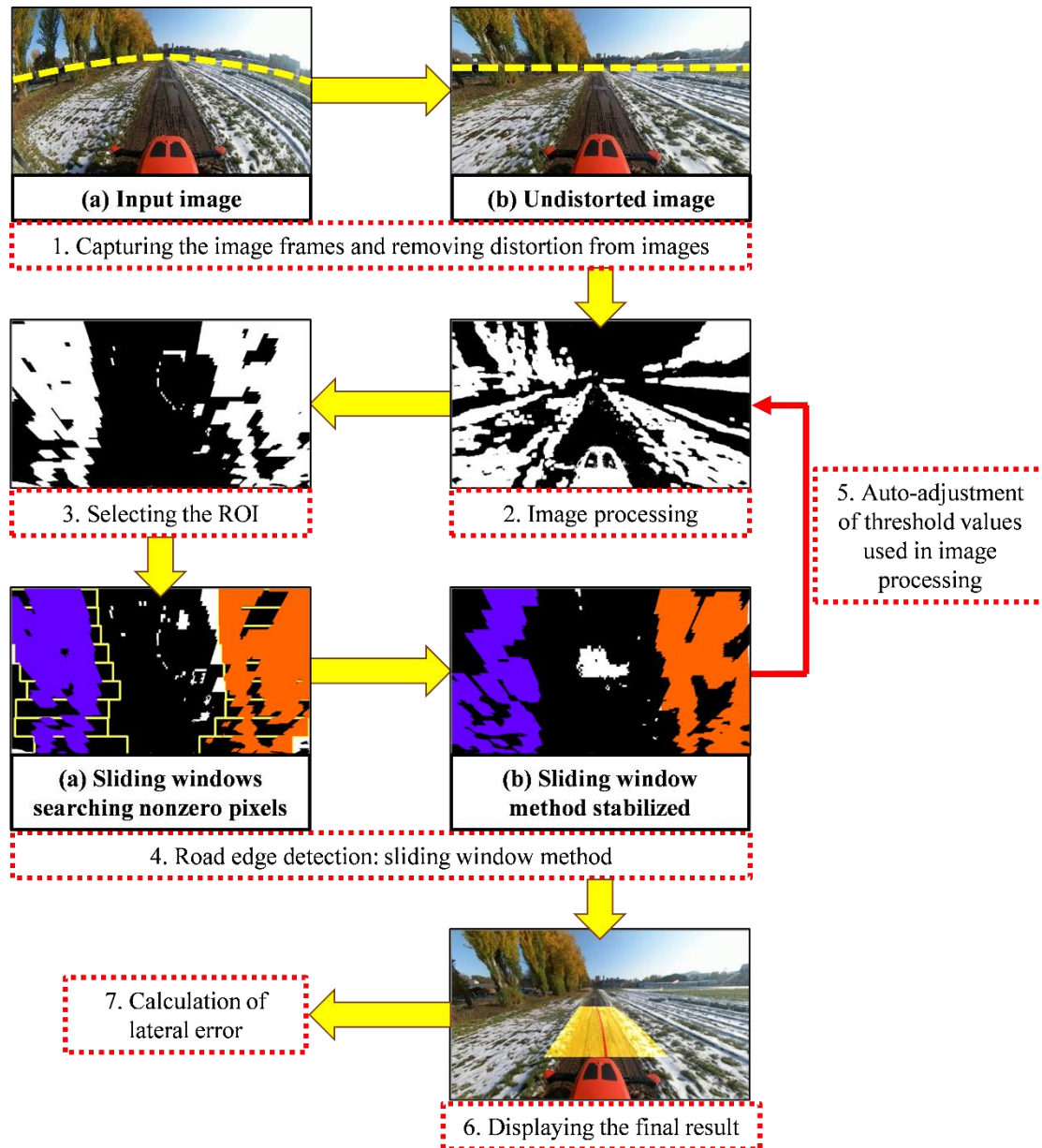


Figure 3.5. Overview of the road detection algorithm step by step

In the development of the road detection algorithm, a meticulous orchestration of seven key processes is undertaken to ensure the robustness and precision of identifying road edges within captured images. An overview of the entire process is depicted in Fig. 3.5. Throughout these sequential processes, particular emphasis is

placed on algorithmic optimization and parameter tuning, aiming to ensure steadfast performance across diverse environmental conditions and road types. The comprehensive amalgamation of these processes ultimately leads to the creation of a sophisticated road detection algorithm, adept at accurately discerning road edges and computing lateral error within agricultural environments.

3.2.3.1. Capturing the image frames and removing distortion from images

The KODAK PIXPRO 4KVR360 camera, selected for image capture in this study, operates within the RGB color space, adhering to the RGB color model. RGB color spaces are defined by the chromaticities of the red, green, and blue additive primaries, facilitating the production of a diverse array of colors. Each color channel typically spans from 0 to 255 on a scale, representing the range of intensity, commonly known as "color depth."

Equipped with a fisheye lens, the camera captures images as 2D matrices containing RGB values for each pixel, thereby enabling comprehensive environmental observation. Leveraging OpenCV's 'VideoCapture' class, frames are captured in real-time and subsequently processed to extract pertinent information. The camera intrinsic matrix $\mathbf{K}$ & $\mathbf{D}$, determined through the procedure outlined in section 3.2.2, is then employed to rectify any distortion present in the images in real time. This distortion correction process ensures the accuracy of subsequent image processing tasks and enhances the reliability of the captured visual data.

3.2.3.2. Image processing

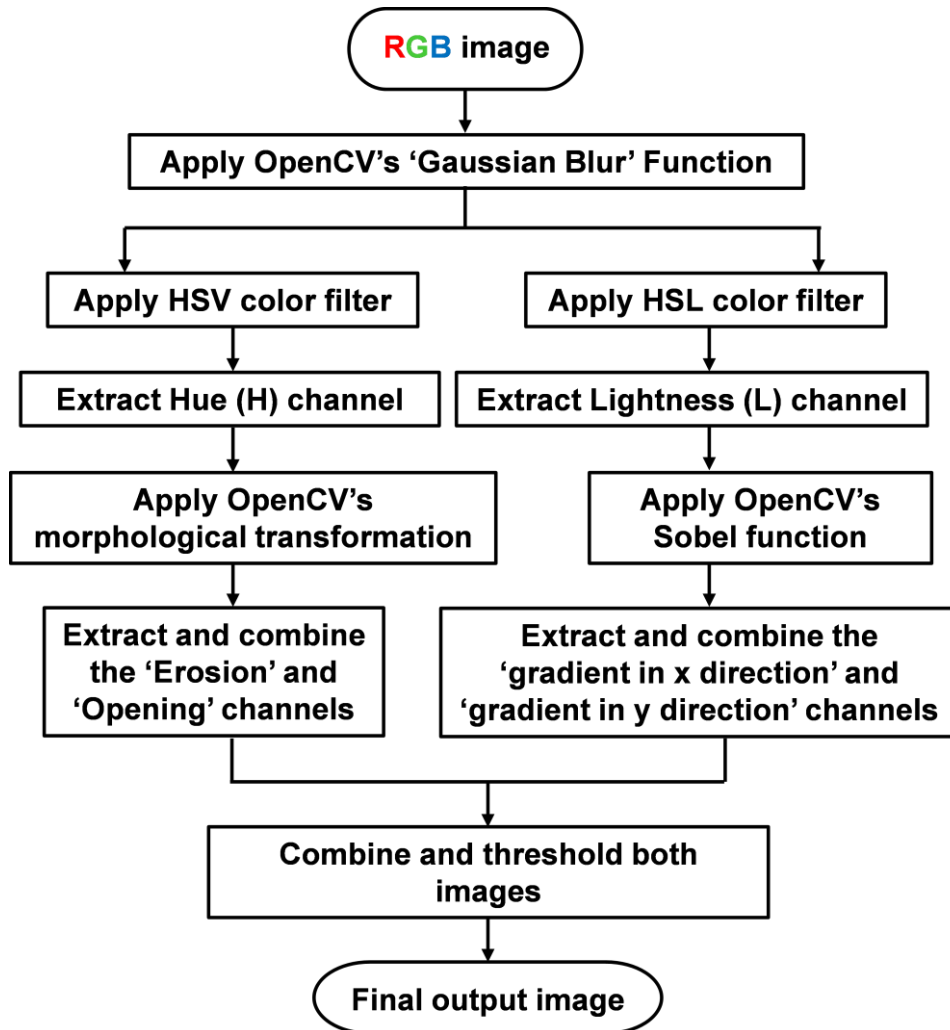


Figure 3.6. Flowchart illustrating the process of image processing under 'general use' condition

The images, now free from camera distortion, undergo a series of filtering processes utilizing a combination of several image filters and pre-defined threshold values. These threshold values were carefully selected after rigorous testing on various video samples recorded under different weather conditions. However, certain extreme conditions, such as very bright or extremely cloudy skies, are not included in these samples. Under such conditions, the threshold values are automatically adjusted, with

the auto-adjustment process described in a later section. An overview of the entire image processing method under 'general use' conditions is shown in Figure 3.6.

Initially, the undistorted RGB image obtained after removing camera distortion contains image noise. Noise in an image is the presence of artifacts that do not originate from the original scene content. Generally, noise is a statistical variation of a measurement created by a random process, often emerging as a grainy structure covering the image. Noise can take different forms and appearances within an image and is, in most cases, an unwanted or disturbing effect that reduces the subjective image quality. Thus, one of the most crucial tasks in imaging is to create a strong signal with a minimum amount of noise.

To address this, OpenCV's 'GaussianBlur' function is applied to smooth the images. Gaussian blur, also known as Gaussian smoothing, is the result of blurring an image by a Gaussian function. It is widely used as a pre-processing stage in computer vision algorithms to reduce image noise and detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen. In OpenCV, a Gaussian kernel is used to perform the blurring operation using the 'GaussianBlur' function. The width and height of the kernel need to be specified, and these values should be positive and odd. Gaussian blur is highly effective in removing noise from an image.

Following this, several filters are successively applied. The first color filter used is the HSV color filter. Originally designed in the 1970s by computer graphics researchers, HSV is a cylindrical color model that remaps the RGB primary colors into dimensions that are easier for humans to understand. These three dimensions are hue, saturation, and value. Hue specifies the angle of the color on the RGB color circle. A 0°

hue results in red, 120° results in green, and 240° results in blue. Saturation controls the amount of color used. A color with 100% saturation will be the purest color possible, while 0% saturation yields grayscale. Value controls the brightness of the color. A color with 0% brightness is pure black while a color with 100% brightness has no black mixed into the color. Because this dimension is often referred to as brightness, the HSV color model is sometimes called HSB.

It is important to note that the three dimensions of the HSV color model are interdependent. If the value of a color is set to 0%, the amount of hue and saturation does not matter as the color will be black. Likewise, if the saturation of a color is set to 0%, the hue does not matter as there is no color used. Because the hue dimension is circular, the HSV color model is best depicted as a cylinder (Fig. 3.7). In this cylinder, colors of each hue are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top.

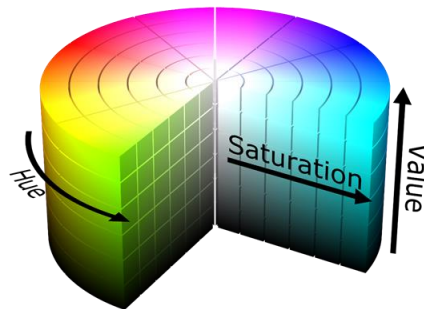


Figure 3.7. HSV color space

HSV color space is more often used in computer vision owing to its superior performance compared to RGB color space in varying illumination levels. The HSV image frames are then thresholded to isolate the desired color range, such as vegetation predominantly found at the edges of unpaved roads.

The second filter converts the RGB image frames to HSL (Hue, Saturation, and Lightness) color space (Fig. 3.8). HSL is another cylindrical color model that shares two dimensions with HSV while replacing the value dimension with a lightness dimension. Hue specifies the angle of the color on the RGB color circle, exactly like HSV. Saturation controls the purity of the color, exactly like HSV. Lightness controls the luminosity of the color, differing from the HSV value dimension in that the purest color is positioned midway between the black and white ends of the scale. A color with 0% lightness is black, 50% is the purest color possible, and 100% is white.

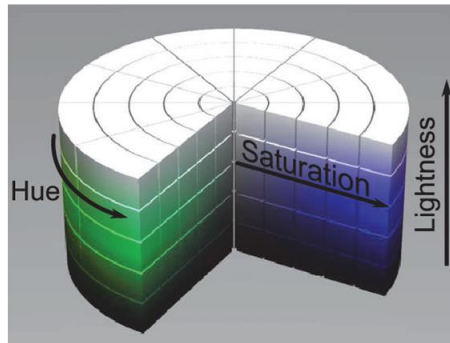


Figure 3.8. HSL color space

Even though the saturation dimension theoretically is similar between the two color models (controlling how much pure color is used), the resulting saturation scales differ between the models due to the brightness to lightness remapping. The HSL color space can detect lines and edges (e.g., white lane markings on paved roads) with proper thresholding.

The resulting HSV image frame is then morphologically transformed to further reduce noise. Morphological transformations are simple operations based on the image shape, typically performed on binary images. These operations require two inputs: the original image and a structuring element or kernel that decides the nature of the

operation. The two basic morphological operators used in this research are Erosion and Opening.

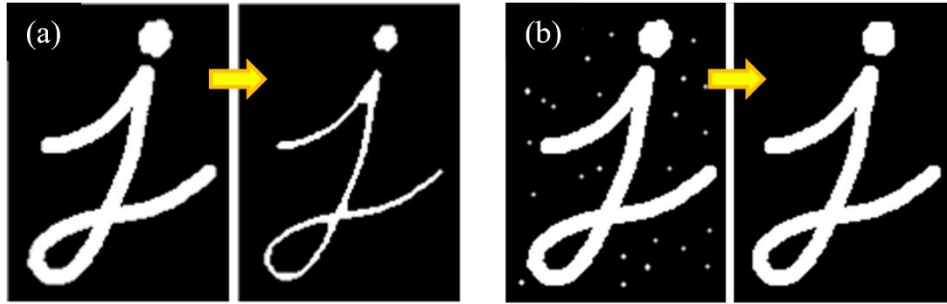


Figure 3.9. Morphological transformation : (a) ‘Erosion’ operation, (b) ‘Opening’ operation

The basic idea of erosion (Fig. 3.9a) is similar to soil erosion; it erodes the boundaries of the foreground object. The kernel slides through the image and a pixel in the original image will be considered 1 only if all the pixels under the kernel are 1; otherwise, it is eroded (made to 0). Therefore, all the pixels near the boundary will be discarded depending on the size of the kernel, reducing the thickness or size of the foreground object. This is useful for removing small white noises and detaching connected noises.

Dilation is the opposite of erosion. Here, a pixel element is 1 if at least one pixel under the kernel is 1. This increases the white region in the image or the size of the foreground object. Normally, in cases like noise removal, erosion is followed by dilation. Erosion removes white noises but also shrinks the object. After dilation, the noise is gone, and the object area increases. Opening, which is erosion followed by dilation, is useful in removing noise (Fig.3.9b).

A Sobel operator is applied to the HSL image frames to emphasize the pixels, highlighting the edges and lines. The Sobel operator, sometimes called the Sobel–

Feldman operator or Sobel filter is used in image processing and computer vision, particularly within edge detection algorithms. It creates an image emphasizing edges. The Sobel operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions. It computes an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. While it produces a relatively crude gradient approximation, particularly for high-frequency variations in the image, it is computationally inexpensive.

Thresholding is a technique in OpenCV involving the assignment of pixel values relative to a specified threshold value. Each pixel value is compared with the threshold value: if the pixel value is smaller than the threshold, it is set to 0; otherwise, it is set to a maximum value (generally 255). Thresholding is a very popular segmentation technique used for separating an object considered as a foreground from its background. In computer vision, thresholding is typically performed on grayscale images. In this research, thresholding is used while combining the two filtered images to avoid any unnecessary pixels in the final filtered image.

The process of combining both filtered images is done using bitwise operators. A bit is a binary digit, the smallest unit of data in computing, representing one of the two binary values, either a 0 or a 1. In Python, bitwise operators are used to perform bitwise calculations on integers. The integers are first converted into binary, and then operations are performed bit by bit, hence the name bitwise operators. Python comes with a few different kinds of operators, such as arithmetic, logical, and comparison operators. In this research, the bitwise AND and bitwise OR operators are used.

The bitwise AND operator (&) performs logical conjunction on the corresponding bits of its operands. For each pair of bits occupying the same position in the two numbers, it returns a 1 only when both bits are 1. Arithmetically, this is equivalent to a product of two bit values. The bitwise AND of numbers n_a and n_b can be calculated by multiplying their bits at every index i :

$$(n_a \& n_b)_i = n_{a_i} \times n_{b_i} \quad (3.16)$$

The bitwise OR operator (|) performs logical disjunction. For each corresponding pair of bits, it returns a 1 if at least one of them is a 1. The arithmetic behind it is a combination of a sum and a product of the bit values. To calculate the bitwise OR of numbers n_a and n_b , the following formula can be applied to their bits at every index i :

$$(n_a | n_b)_i = n_{a_i} + n_{b_i} - (n_{a_i} \times n_{b_i}) \quad (3.17)$$

Finally, the selected filters are combined using Python's logical operators (and, or, not operators). The final output image is thus obtained, displaying the road surface in black and the surroundings in white, effectively distinguishing the road from its environment (Fig. 3.5, step 2).

3.2.3.3. Selecting the region of interest

The selection of a Region of Interest (ROI) is crucial for accurate road detection and depends on various external factors such as the field of view, the width of the road, and the look-ahead distance. An ROI is a portion of an image that is considered important and will be operated on in some way. In this study, the ROI is manually calculated and defined within the system to ensure precise and consistent image processing.

The look-ahead distance is particularly significant in determining the ROI. This distance refers to how far along the path the robot should look from its current location to compute the angular velocity commands. Adjusting this parameter influences how the robot tracks the path, with two major goals: regaining the path and maintaining the path.

To quickly regain the path between waypoints, a small look-ahead distance can be used, causing the robot to move swiftly towards the path. However, this can result in the robot overshooting and oscillating along the desired path. Conversely, a larger look-ahead distance reduces oscillations but may lead to larger curvatures near corners. Thus, the look-ahead distance must be properly tuned for the specific application to achieve a balance between path regaining and maintaining stability (Coulter, 1992).

Using the predefined ROI (based on the look-ahead distance), the filtered image obtained from the previous step is warped to achieve a bird's-eye view. A bird's-eye view is an elevated perspective from above, as though the observer were a bird. OpenCV's 'getPerspectiveTransform' function is used to warp the ROI. This function calculates a perspective transform from four pairs of corresponding points, computing the 3x3 matrix of a perspective transform as follows:

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = map_matrix \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (3.18)$$

$$dst(i) = (x'_i, y'_i) \quad (3.19)$$

$$src(i) = (x_i, y_i) \quad (3.20)$$

Here, i is 0, 1, 2, 3. The parameter src represents the coordinates of the quadrangle vertices in the source image, and the parameter dst represents the coordinates of the corresponding quadrangle vertices in the destination image.

Warping the image transforms it into a bird's-eye view of the ROI, which significantly facilitates the estimation of road edges and the center of the road surface. This transformation allows for a more straightforward analysis and accurate detection of the road's geometry. The resulting image from this step, which demonstrates the selection of the ROI, is depicted in Figure 3.5, step 3, titled 'Selecting the ROI'.

3.2.3.4. Road edge detection: Sliding window method

In this section, the sliding window method is applied to the warped image frames obtained from the previous step to detect road edges. The sliding window method involves moving windows of a specified size over the image frame data, sampling pixel values systematically. These windows, represented by yellow rectangles in Fig. 3.5, step 4a titled 'Sliding windows searching nonzero pixels' collect statistical data over the pixels within each window.

Initially, when a window does not contain sufficient data, a histogram of the image is calculated, and its peak values are used to initialize the windows. The two peaks of the histogram determine the starting points for the sliding windows algorithm. In subsequent steps, the sliding windows continue to search for non-zero pixels using data from the previous step's starting point, maintaining a state that remembers prior data for continuity and accuracy.

Agricultural roads, unlike regular roads, exhibit significant variations in width. To accommodate these changes during navigation, nine sliding windows are utilized along the height (1080 pixels) of the image frames. Each sliding window has a height of 120 pixels, which allows for the detection of even minor changes in road width. Each window searches for non-zero pixels, where black pixels are assigned a value of 0 and white pixels a value of 1. The non-zero pixel values are stored in an array, and a

second-degree polynomial line is fitted to these values. These polynomial lines represent the left and right edges of the road and help in determining the center of the road. The detection result is depicted in Fig. 3.5, step 4a.

As the sliding windows search for non-zero pixels, they create an ROI within the frame. If non-zero pixels are consistently found within this ROI, the sliding window method achieves stability, as shown in Fig. 3.5, step 4b.

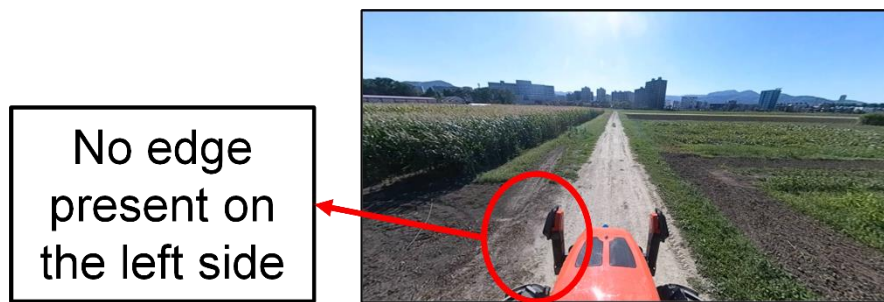


Figure 3.10. Absence of the left edge in an unpaved dirt road inside the Hokkaido University experimental farm

In agricultural settings, the presence of grasses and shrubs at the road edges typically indicates the road boundaries. However, in the absence of such vegetation, the edges are considered missing. Figure 3.10 illustrates an example of a missing edge. In such scenarios, the detected positions of the left and right edges are analyzed, and each edge is assigned a predefined range of position values based on the road width. If the detected position of an edge falls outside this range, the center is calculated using the other edge, with the road width halved and added or subtracted to the considered edge.

It is important to note that the sliding window algorithm cannot rely solely on one edge due to the substantial variation in agricultural road widths, which can range from 2 to 5 m. Consequently, an average width is considered for small portions of the road, although it is unsafe to use an average width for the entire road's navigation.

For unpaved roads that lack defined edges, the edges are calculated by joining the points representing the mean area detected by the sliding windows. The detected area depends on the position of the tractor on the road, i.e., the ROI. This feature can sometimes cause a shift in the calculated centerline to one side of the actual centerline.

This sliding window method, with its robustness and adaptability, ensures that the road edges are accurately detected, facilitating reliable navigation even in the variable and unpredictable conditions characteristic of agricultural environments.

3.2.3.5. Auto-adjustment of threshold values used in image processing

As described in section 3.2.3.2, the 'general use' condition for image processing employs predefined threshold values. However, these values must adapt to changing weather conditions to maintain accurate road detection. Changes in environmental conditions, such as variations in sunlight and cloud cover, affect light intensity, necessitating dynamic adjustment of threshold values for real-time navigation.

The auto-adjustment process begins by calculating the road width from the detected left and right edges using the sliding windows method. This calculated width is then compared to the actual road width. If discrepancies exist between the calculated and actual widths, the auto-adjustment process is initiated. An iterator systematically adjusts the threshold values over a predefined range until the calculated width matches the actual road width.

Unpaved roads, characterized by their varying widths, present additional challenges. In such cases, the average road width is first determined by measuring the width at various points along the road. During the auto-adjustment process, a predefined range for acceptable widths is established. If the calculated width falls

within this range, the auto-adjustment process ceases, having achieved an acceptable match. Figure 3.11 provides a step-by-step illustration of this process.

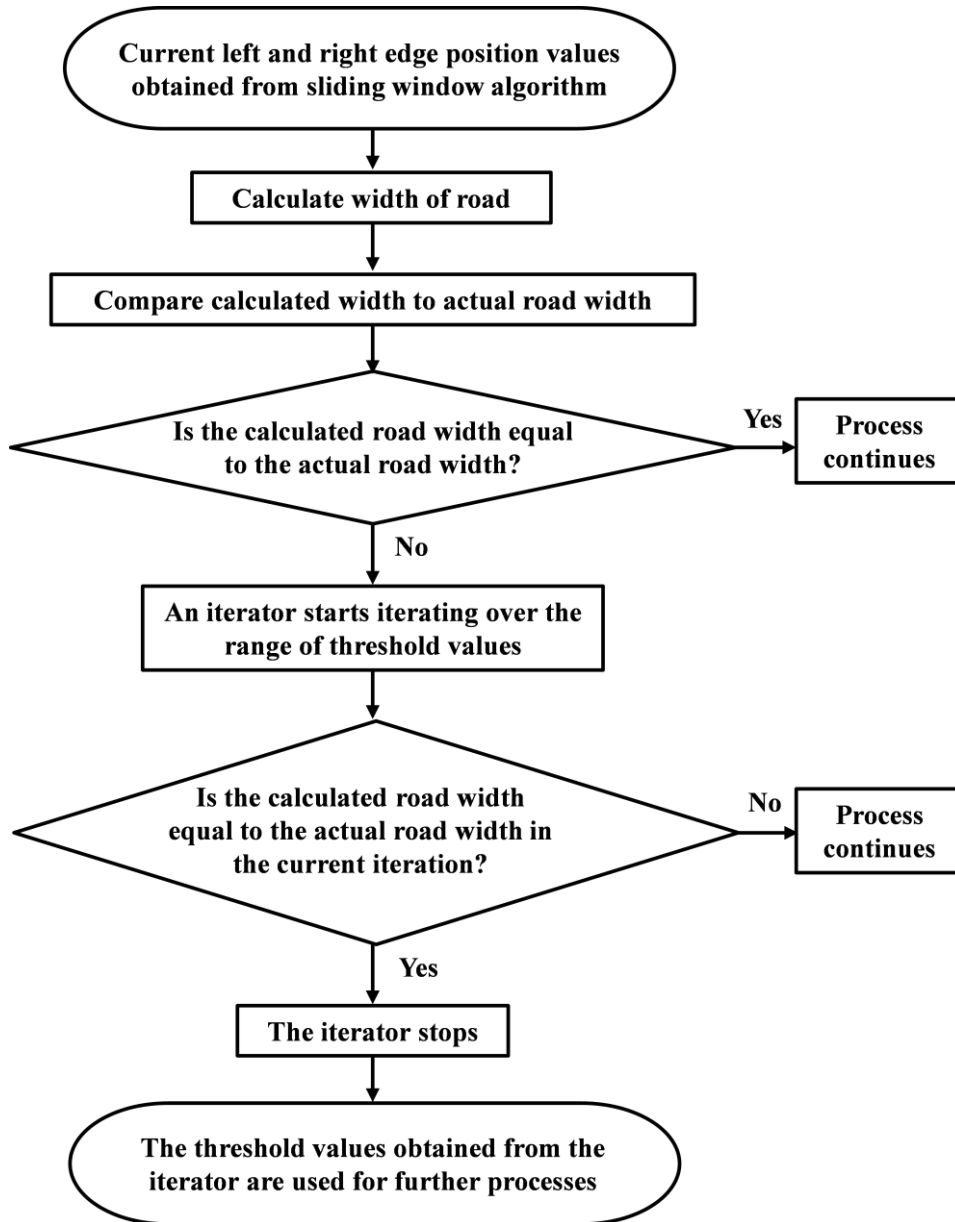


Figure 3.11. Flowchart showing the process of auto-adjusting the threshold values

3.2.3.6. Displaying the final result

Upon obtaining the second-degree polynomials from the warped images, the coordinates within these polynomials are converted to real-world coordinates by inverting the warping transformation. This is achieved using the same OpenCV

function, ‘warpPerspective’, employed in the initial warping operation. The final result is then superimposed onto the original image (as depicted in Fig. 3.5, step 1b), which is free from any camera distortion.

To visualize the detected road features, the OpenCV functions ‘fillPoly’ and ‘polylines’ are used to construct the yellow polygon and the red center line, respectively. The yellow polygon represents the detected road surface within the ROI relative to the identified left and right edges, while the red polyline indicates the calculated center of the road.



Figure 3.12. Display of the final result of road detection : (a) Unpaved (dirt) road, (b) Paved (asphalt) road

These two visual elements are then blended with the undistorted image using the OpenCV function ‘addWeighted’. This function combines two images based on specified weights, following the equation.

$$g(x) = (1 - \alpha')f_0(x) + \alpha'f_1(x) \quad (3.21)$$

where α' is a blending factor that varies between 0 and 1. In this research, a value of 0.7 is used for α' , ensuring a balanced blend that highlights the detected features while maintaining the context of the original image.

After blending the output with the undistorted image, the final result is obtained, as illustrated in Figs. 3.12a and 3.12b. The blended image is then displayed using the

OpenCV function ‘imshow’. This comprehensive visualization step is crucial for validating the road detection process and for practical implementation in autonomous navigation systems. It ensures that the detected road boundaries and centerline are accurately represented in their real-world context, facilitating an intuitive assessment of the road detection algorithm's performance.

3.2.3.7. Calculation of lateral error

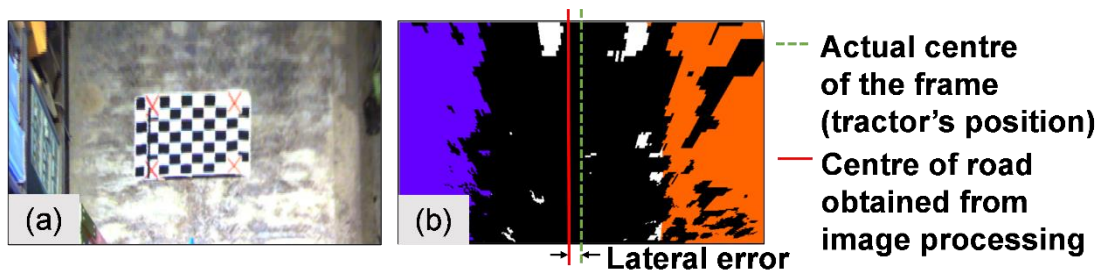


Figure 3.13. Calculation of lateral error from the calculated centerline and tractor's position : (a) Manual calculation of real-world distances from the chessboard, (b)

Lateral error estimation

The lateral error is a critical parameter in autonomous navigation, representing the deviation of the tractor's position from the center of the road. To estimate this error, real-world distances are correlated with pixel distances using a calibration process involving a chessboard pattern. This approach ensures accurate translation of pixel measurements to real-world metrics.

First, the physical dimensions of the chessboard, both horizontal and vertical, are measured in meters. These measurements are then matched with the corresponding pixel dimensions obtained from the image (Fig. 3.13a). By establishing this relationship, a conversion factor is derived that allows the translation of pixel distances in the image frames to real-world distances.

Figure 3.13b illustrates the process of calculating lateral error. The real-world distances, obtained through the aforementioned calibration method, are used alongside pixel values from the image frames. In this figure, the left edge of the road is represented by the blue area, the right edge by the orange area, and the detected center of the road by the red line. The green dashed line marks the actual center of the image frame. The lateral error is the distance between this green dashed line and the red line, measured in pixels. This pixel-based error is then converted to real-world distances using the established conversion factor.

This real-world lateral error can be transmitted to the tractor's automatic navigation system PC (personal computer) via serial port communication, as depicted in Fig. 3.1. The navigation system, developed by Takai et al. (2004), calculates the steering angle (δ) using the Eq. 3.22 (Kise et al., 2001).

$$\delta = \alpha_s \Delta \varepsilon + \beta_s \Delta \phi_I \quad (3.22)$$

In this equation, $\Delta \varepsilon$ represents the lateral error, and $\Delta \phi_I$ denotes the heading error. The parameters α_s and β_s are proportional gains. Typically, the lateral error is determined using RTK-GNSS, and the heading error is provided by an IMU. However, in this research, the lateral error is derived from the machine vision system.

Additionally, the travel speed can be set within the tractor's automatic navigation system, allowing for dynamic adjustment based on real-time navigation requirements. This integration of machine vision with traditional navigation aids enhances the overall accuracy and reliability of the autonomous navigation system.

3.3. Experimental Setup

The experimental setup encompasses a comprehensive framework designed to facilitate a diverse range of real-time field tests. This section elucidates the intricacies

of the test field and the data collection methodology employed throughout the experimental phase. Through detailed explanations, the experimental setup aims to provide a clear understanding of the environment, equipment, and procedures utilized in the research endeavors.

3.3.1. Description of test field

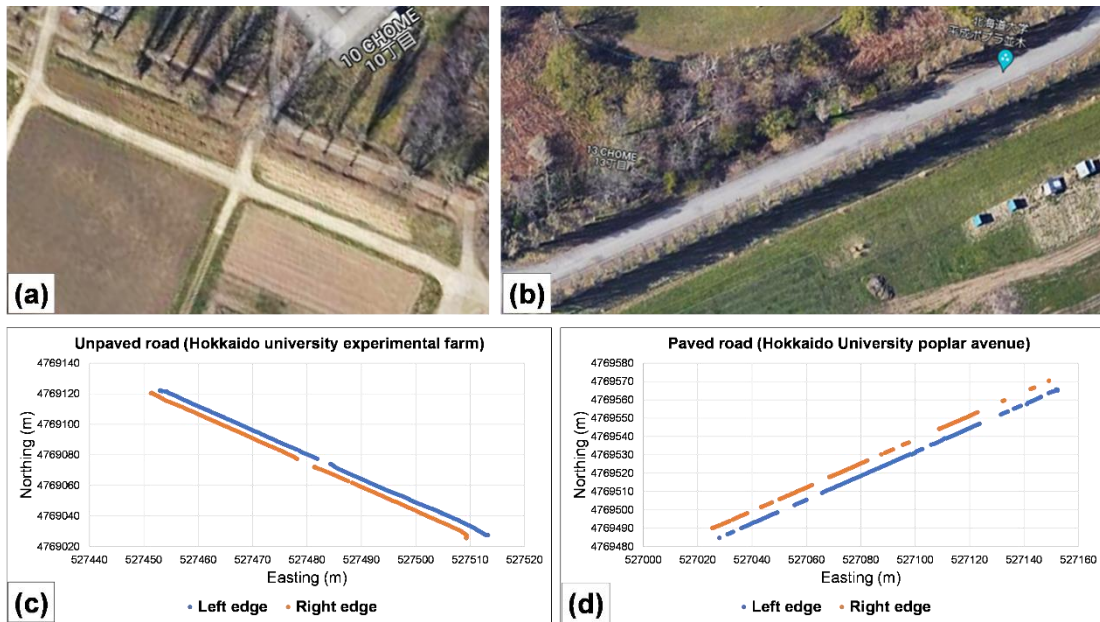


Figure 3.14. Road edge data recorded by using RTK-GNSS : (a) Unpaved road (bird's eye view), (b) Paved road (bird's eye view), (c) Unpaved road edge data, (d) Paved road edge data

The roads depicted in Fig. 3.14 were utilized as the primary test sites for field experimentation. Notably, the unpaved road is characterized by its variable width and the presence of an intersection that effectively divides it nearly in half, as showcased in Fig. 3.14a. Spanning approximately 110 m in length, this straight road presents a clear path for testing purposes, with no immediate tree cover in its vicinity. On the other hand, the paved road showcased in Fig. 3.14b extends about 150 m in length and maintains a straight trajectory. However, it is surrounded by poplar trees on both sides,

presenting a distinct environmental challenge. Both figures provide a bird's eye view of the respective test roads.

3.3.2. Data collection method

Prior to initiating the field experiments, meticulous efforts were undertaken to manually obtain data regarding road width and edge characteristics. Given the variability in width observed in unpaved roads, measurements were taken across different segments to establish an average width, ensuring uniformity during experiments. Furthermore, the left and right edges of the road were precisely recorded using a GNSS receiver (R10, Trimble, USA), with human judgment guiding the determination of these undefined edges.

Conversely, for paved roads where a constant width was assumed, data collection relied on a GNSS receiver to record edge data. In Figs. 3.14c and 3.14d, the left and right edge data are visually represented as blue and orange dotted lines, respectively, within a geographic Cartesian coordinate system (northing and easting).

Fig. 3.14c reveals instances of missing data in the middle of the path, attributable to the presence of an intersection within the road, as detailed in section 3.3.1. Similarly, Fig. 3.14d exhibits discontinuous data, a consequence of the presence of poplar trees on both sides of the road, leading to fluctuations in RTK fix solutions. This scenario exemplifies the challenges encountered in operating an autonomous tractor using RTK-GNSS in environments with extensive tree cover.

The obtained edge data from both road types served as the basis for calculating the true centerlines, which were subsequently utilized as reference data to evaluate the efficiency of the vision-based system. Multiple experimental runs were conducted on both road types, with tractor position data recorded using another GNSS receiver

(Alloy, Trimble, USA) solely for assessing system efficiency. Importantly, the vision-based system operated independently of RTK-GNSS during field experiments, with RTK data utilized solely for performance assessment purposes. The next section delves into a detailed discussion of the results obtained from these field experiments.

3.4. Results and Discussion

This section critically examines the outcomes derived from experiments conducted on both unpaved and paved roads, as outlined in section 3.3.1. These experiments were conducted across varying times of the day and under diverse weather conditions. The subsequent analysis aims to offer comprehensive insights into the performance and adaptability of the proposed system in real-world scenarios, shedding light on its efficacy in navigating roads under different environmental circumstances.

3.4.1. Real-time field testing - Unpaved Road

3.4.1.1. Daytime experiment

Field experiments were conducted on the unpaved road depicted in Fig. 3.14a. Figure 3.15 illustrates two sets of experimental results obtained by operating the tractor at varying speeds. The travel distance ranged from 90–100 m across all experiments. Tractor speeds ranged between 3.6 and 6 km/h and remained constant throughout each trial. All experiments were conducted during daylight hours.

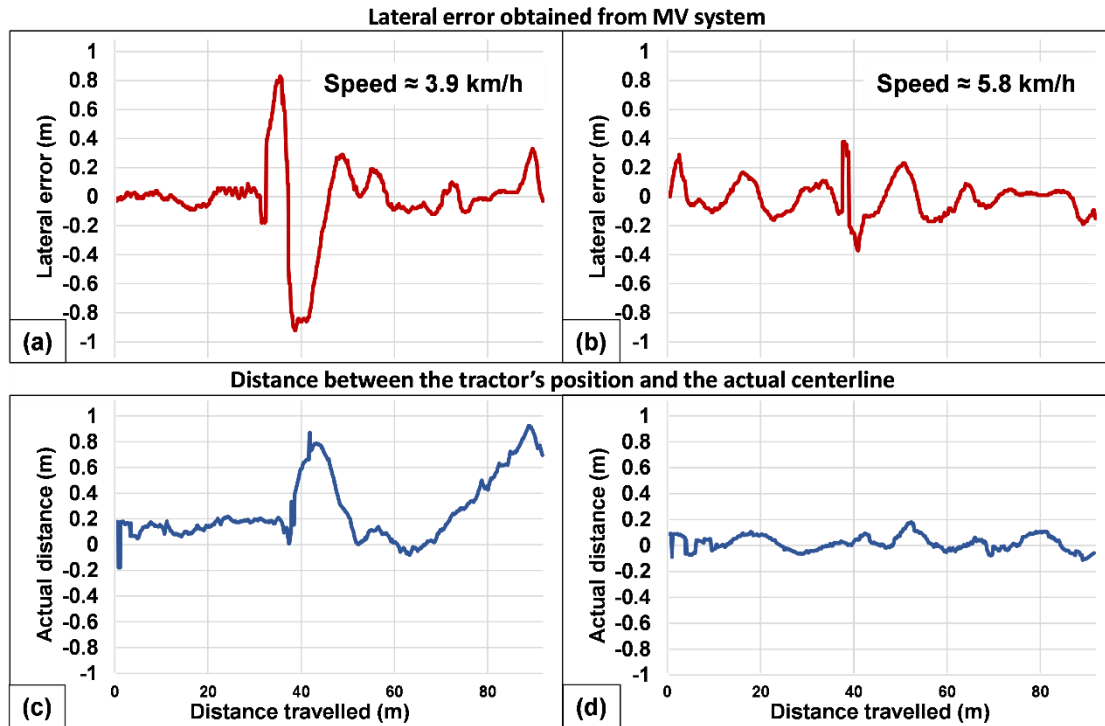


Figure 3.15. Experimental results (daytime) obtained from experiments done on an unpaved road in the Hokkaido University experimental farm : (a), (b) Lateral error obtained from the machine vision system at two different speeds (3.6 km/h and 5.8 km/h respectively), (c), (d) Distance between the tractor's position and actual centerline at two different speeds (3.6 km/h and 5.8 km/h respectively)

The lateral error values obtained from the machine vision system during these experiments are depicted by the red continuous lines in Figs. 3.15a and 3.15b, expressed in meters. Positive and negative values denote the left and right sides of the calculated centerline, respectively. Analysis of these graphs reveals that the lateral error remains within 0.2 m for the majority of the time. The observed oscillations in the lateral error can be attributed to the uneven edges typical of unpaved (dirt) roads.

In Fig. 3.15a (speed ≈ 3.9 km/h), instances of lateral errors reaching up to 0.8 m in the middle portion are evident. This anomaly arises from the presence of an

intersection along the road (Fig. 3.14c). As intersections lack distinct edges, the tractor navigates this section of the road by connecting the endpoints of the edges on both ends of the intersection. However, these endpoints may not always fall within the ROI. Consequently, when the endpoints are not detected, the sliding windows initiate a search for any non-zero pixels within the ROI. The identification of such pixels as edges can result in significant lateral errors. Alternatively, if no non-zero pixels are found within the ROI, previous data are utilized to calculate the edges.

Fig. 3.15b (speed ≈ 5.8 km/h) reveals that the lateral error values at the intersection are comparatively smaller in this experiment. This difference can be attributed to the higher speed of the tractor. With increased speed, the ROI undergoes more rapid changes. Consequently, the endpoints of the intersection enter the ROI at a faster rate, leading to diminished lateral errors.

The blue continuous lines depicted in Figs. 3.15c and 3.15d illustrate the distance calculated between the tractor's position and the actual center of the road. In Fig. 3.15c (speed ≈ 3.9 km/h), the tractor predominantly veers to the left of the actual centerline, with the distance hovering around 0.2 m in the initial segment of the graph (i.e., preceding the intersection). This discrepancy arises from the absence of well-defined edges, as elaborated in section 3.2.3.4. In scenarios involving unpaved roads devoid of clear boundaries, the sliding window method detects an area alongside the road, with the mean of this area considered as the road's edge. Consequently, variations in the extent of edge area within the ROI can influence a deviation in the calculated centerline.

In the section of the road following the intersection, the tractor consistently veered towards the left side of the road, as illustrated in Fig. 3.15c (speed ≈ 3.9 km/h),

despite minimal changes in the lateral error values depicted in Fig. 3.15a. This occurrence can be attributed to specific characteristics of the sliding window algorithm and the consistent width of the road, which serves as a conditional parameter. As elaborated in section 3.2.3.4, for unpaved roads, the sliding window algorithm remains stabilized if the width falls within the predefined range and if non-zero pixels persistently appear within the ROI delineated by the sliding windows. Throughout this experiment, even as the width expanded towards the end of the road, the stipulated condition regarding road width remained valid, thereby ensuring the continued stabilization of the sliding windows. Consequently, the lateral error exhibited minimal variation, resulting in the tractor consistently veering leftwards. However, towards the conclusion of its trajectory, there is an observable correction towards the right. This adjustment occurred due to the non-fulfillment of stabilization conditions at that juncture, prompting the sliding windows to resume searching for non-zero pixels.

In Fig. 3.15d (speed ≈ 5.8 km/h), it is evident that the tractor proceeded straight, adhering closely to the actual centerline for the majority of the duration. This outcome can be attributed to the higher speed of the tractor, resulting in a more rapid alteration of the ROI and consequently leading to improved detection outcomes.

3.4.1.2. Nighttime experiment

Figure 3.16 illustrates experiments conducted during nighttime on the same road depicted in Fig. 3.14a. The distance covered during these nighttime trials ranged from 50 to 55 m, shorter than that of the daytime experiments. Speed was maintained between 2 to 4 km/h. Nighttime detection relies solely on the reach of the tractor's headlights, a significant departure from daytime conditions. In scenarios where the road width exceeds the coverage of the headlights, edge detection becomes unfeasible.

However, the road under consideration, with a width ranging from 2.4 to 4 m, was deemed suitable for nighttime experiments. The graphical representation in Fig. 3.16 mirrors that of Fig. 3.15. Figures 3.16a and 3.16b depict the lateral error values recorded during experimental runs, expressed in meters. Positive and negative values indicate the left and right sides of the calculated centerline, respectively. Figs. 3.16c and 3.16d illustrate the distance measured between the tractor's position and the actual center of the road.

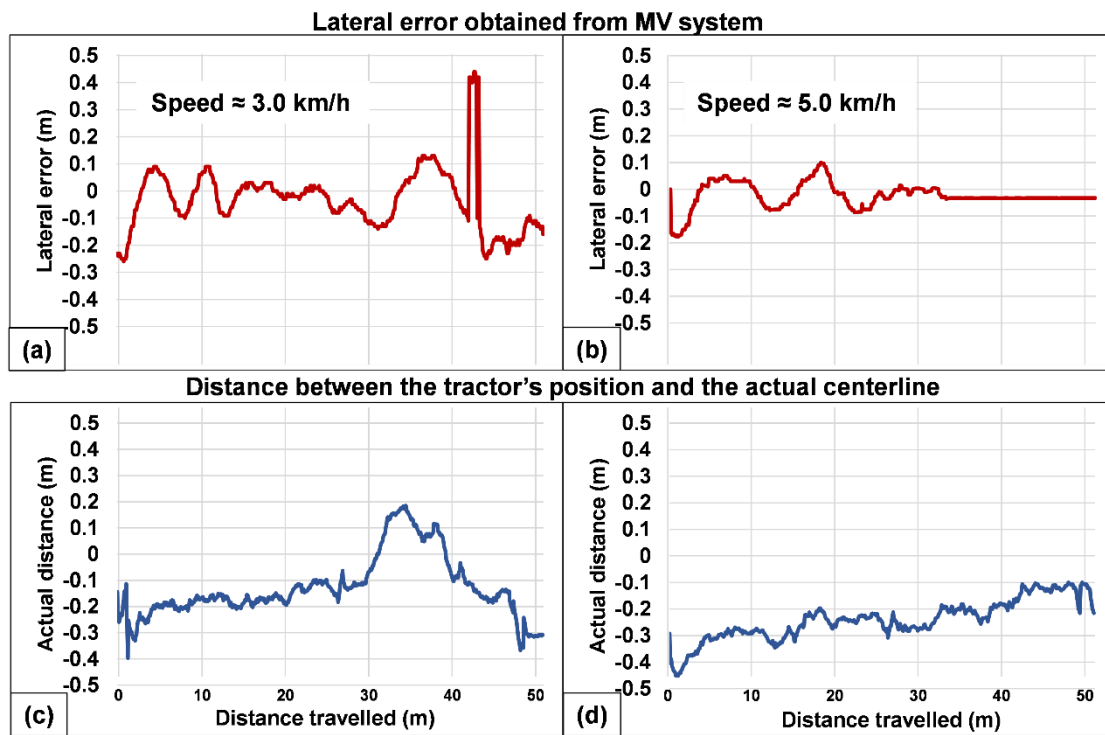


Figure 3.16. Experimental results (nighttime) were obtained from experiments done on an unpaved road in the Hokkaido University experimental farm : (a), (b) Lateral error obtained from the machine vision system at two different speeds (3.0 km/h and 5.0 km/h respectively), (c), (d) Distance between the tractor's position and actual centerline at two different speeds (3.0 km/h and 5.0 km/h respectively)

In Fig. 3.16a and 3.16b, the lateral error values mostly remain within 0.1 m. However, in Fig. 3.16a (speed ≈ 3.0 km/h), there is a momentary spike in the lateral error to approximately 0.5 m towards the end of the path. This anomaly can be elucidated by referencing Fig. 3.16c (speed ≈ 3.0 km/h), where the tractor gradually converges towards the actual centerline, transitioning from the right side to the left side. During the nighttime experiment, visibility primarily relies on the reach of the tractor's headlights, causing the left edge to be more prominently visible within the ROI. Consequently, this visibility bias leads to a deviation of the calculated centerline towards the left side.

In contrast, Fig. 3.16b (speed ≈ 5.0 km/h) demonstrates consistent lateral error values within 0.1 m throughout the experiment, attributed to the tractor's higher speed. This consistency suggests that the tractor followed a straight trajectory without any significant deviations. Conversely, Fig. 3.16d (speed ≈ 5.0 km/h) illustrates a continuous but gradual shift of the tractor from the right side to the left side. Throughout this experiment, the presence of edges within the ROI ensured the stabilization of the sliding windows, resulting in minimal fluctuations in the lateral error values over time.

3.4.2. Real-time field testing - Paved Road

Field experiments akin to those conducted on the unpaved road were also carried out on the paved road. However, the presence of tree canopies on both sides of the road resulted in the RTK fix solution frequently reverting to the float solution. Consequently, the tractor's position data in this experiment comprises a combination of both RTK fixed and float solution data.

In this experiment, the road lacks any white or yellow lane markings. Instead, the edges of the pavements on both sides are identified as the left and right edges (refer to Fig. 3.17 depicting the same road). The tractor traversed an approximate distance of 131 m at a speed of 3.2 km/h. The weather conditions were sunny, characterized by abundant shadows cast by the presence of tree canopies on the road surface. Additionally, being conducted in early winter, the road surface was littered with fallen leaves, concealing numerous potholes (as depicted in Fig. 3.17).



Figure 3.17. Condition of the experimental paved road inside the Hokkaido University campus: Presence of shadows, potholes, and fallen leaves on the road surface

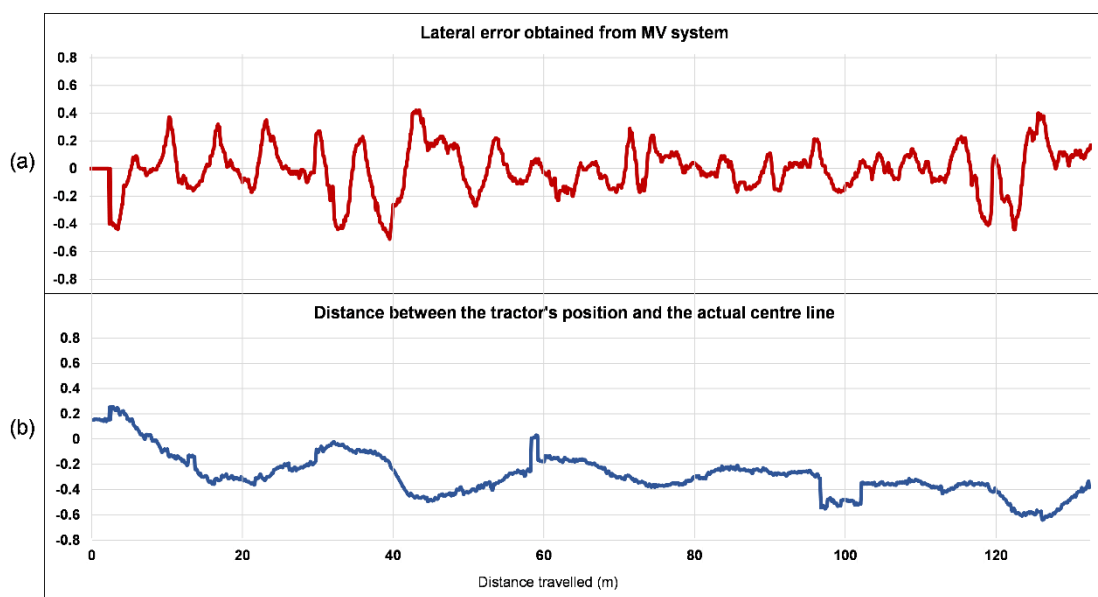


Figure 3.18. Experimental results were obtained from experiments done on a paved road inside the Hokkaido University campus : (a) Lateral error obtained from the machine vision system at a speed of 3.2 km/h, (b) Distance between the tractor's position and the actual centreline

The red continuous line in Fig. 3.18a depicts the lateral error obtained from the machine vision system in meters. Positive and negative values denote the left and right sides of the calculated centerline, respectively. It is evident that the lateral error consistently remained within 0.4 m; however, it exhibited considerable oscillation. Two factors contributed to this oscillation. Firstly, the presence of numerous shadows and fallen leaves on the road surface caused minor shifts in the edges. Secondly, the road's substantial width, approximately 6 m, meant that even slight discrepancies in pixel values translated into significant lateral error values in meters.

In Fig. 3.18b, the blue continuous line illustrates the distance between the tractor's position and the actual centerline. Throughout the duration depicted, the tractor consistently maintains its position to the right of the actual centerline. This phenomenon primarily arises from the accumulation of fallen leaves along the road edges. In certain sections, the right edge becomes almost entirely obscured by fallen leaves, rendering the sliding windows ineffective in detecting it. Consequently, the center of the road is calculated based on the left edge alone. However, the obstruction of fallen leaves also hampers the accurate detection of the left edge, resulting in a gradual shift of the calculated centerline towards the right side of the actual centerline. Nevertheless, given that the values of the lateral error remain relatively small, it can be inferred that the tractor navigated along a predominantly straight trajectory without significant deviations.

3.5. Conclusion

This chapter has introduced a vision-based road detection system designed to facilitate autonomous navigation when the RTK correction signal is unavailable. Leveraging color information captured by an onboard camera, the system employs a sequence of seven distinct processes to delineate road edges. Subsequent iterative computations of the lateral error provide feedback to the automatic navigation system of the tractor, enabling adjustments as necessary.

Field experiments were undertaken to assess the accuracy and efficacy of the developed system. The results confirmed its capability to detect both unpaved and paved roads in real-time scenarios. Moreover, the system demonstrated proficiency in accurately identifying the centerline of roads when adequate edge information was available. During experimental trials, the lateral error typically remained within 0.4 m, a value deemed satisfactory for autonomous navigation, as evidenced by the tractor maintaining a straight trajectory without significant deviation.

It is evident from the experiments that navigating on paved roads is comparatively easier due to the clearer definition of road edges. Conversely, unpaved roads, characterized by varying edge conditions, present greater navigational challenges. Despite these nuances, the vision-based system exhibited robust performance across different road types and environmental conditions, underscoring its potential for enhancing autonomous navigation capabilities in challenging real-world settings.

CHAPTER 4. AUTONOMOUS NAVIGATION OF ELECTRIC VEHICLES IN VINEYARDS USING MACHINE VISION AND YOLOV8

4.1. Introduction

In this chapter, an innovative framework that utilizes machine vision technology is presented, specifically tailored for facilitating the autonomous navigation of electric vehicles (EVs) within vineyard environments. This framework addresses the challenges posed by the unreliable GNSS signals commonly encountered in such settings. Emphasizing the convergence of machine vision and deep learning techniques, this chapter explores the pivotal role they play in developing an algorithm proficient in generating crucial metrics such as lateral and heading error information. These metrics are indispensable for ensuring the accuracy and safety of autonomous navigation systems in vineyards. Additionally, this chapter introduces an inventive technique for detecting vineyard row ends using machine vision techniques.

4.2. System Design and Architecture

This section outlines the design and architecture of the vineyard row navigation system, aiming to facilitate the autonomous movement of EVs within vineyards. Fig. 4.1 provides an overview of the system's architecture, showcasing its key components and their interactions. The following subsections detail the various aspects of the system design.

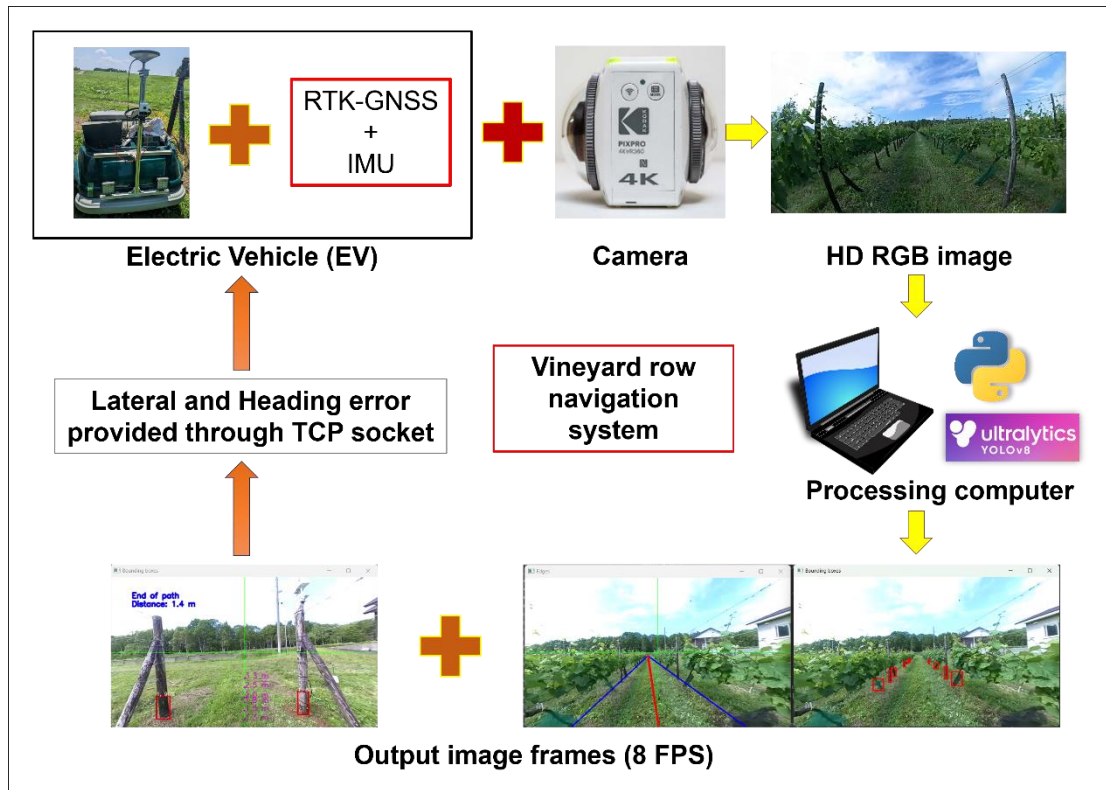


Figure 4.1. Overview of the whole vineyard row navigation system architecture

4.2.1. Camera setup

The KODAK PIXPRO camera, as described in section 2.1.2, is utilized for this research. This camera is strategically positioned at the front section of the vehicle, precisely at a height of 116 cm, and horizontally aligned with the ground plane, as depicted in Fig. 4.2a. The camera's resolution of 1920x1080 pixels has been deemed suitable for vineyard row navigation applications. This meticulous setup guarantees that the vine ends fall within the camera's field of view, facilitating comprehensive coverage of the surroundings. This also ensures that fine details necessary for the effective operation of the navigation algorithm in diverse vineyard environments are captured. Moreover, it ensures a clear and adequate look-ahead distance of 4.5 m, essential for effective navigation and obstacle detection.

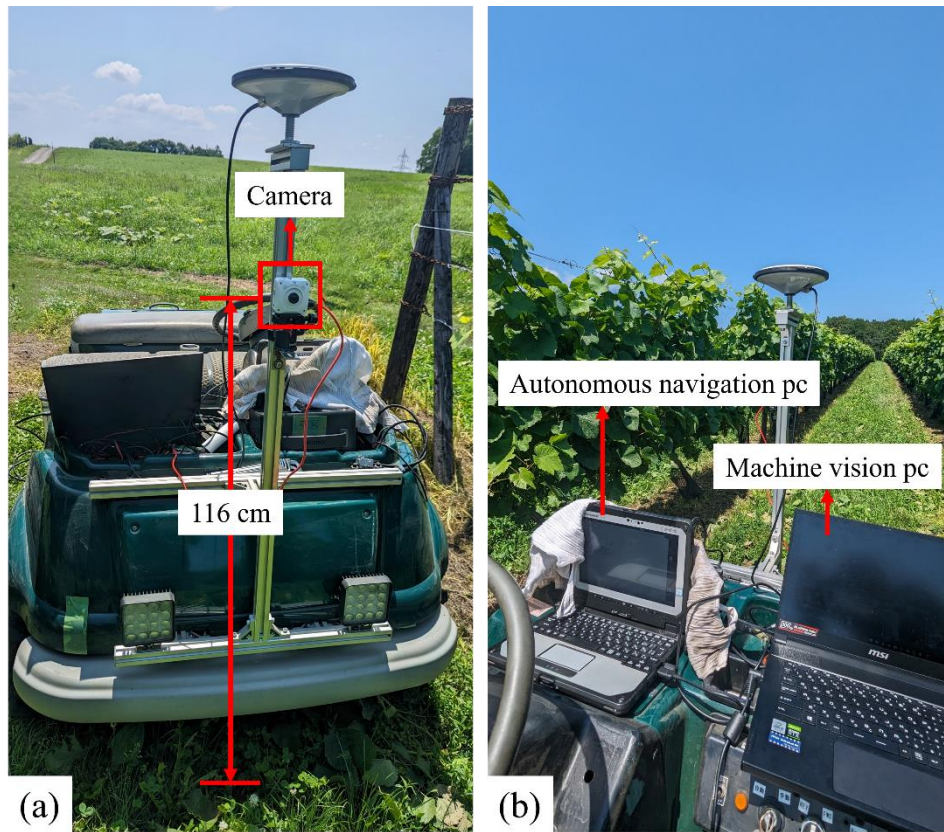


Figure 4.2. Robot Electric Vehicle : (a) Camera positioning, (b) Two types of computers integrated into the EV

4.2.2. Data acquisition

To initiate the development of the tree row navigation algorithm, the first step involved gathering MV data. The annual growth cycle of grapevines, beginning with bud break in the spring and culminating in leaf fall in autumn followed by winter dormancy, necessitated capturing data throughout the entire production season. Each step in this process plays a vital role in the development of grapes with ideal characteristics for winemaking. Consequently, MV data was captured from spring to early winter.



Figure 4.3. Variability in vineyard conditions across grape production seasons under different weather conditions

Videos were recorded at Tsurunuma Winery Co. in Hokkaido prefecture, Japan, throughout the 2022 grape production season. Vineyard environments present certain challenges, primarily due to the variability in environmental conditions and the structure of the vineyard itself. Some of the key challenges include:

Varying lighting and weather conditions. The presence of direct sunlight, shadows, and varying light intensities can significantly affect image quality. Moreover, changes in weather conditions can alter the appearance of the vineyard environment. These variations can complicate the detection of vine ends and line posts, making it necessary to include diverse lighting conditions in the training datasets.

Seasonal changes. The appearance of grapevines changes dramatically throughout the growing season, from sparse early growth to dense foliage in the summer and then leaf fall in autumn. These seasonal changes affect the visibility and shape of vine ends, requiring the model to be trained on images from different times of the year.

Occlusions and clutter. Leaves, branches, and other vineyard elements can occlude vine ends and line posts, making it challenging for the model to detect these

features accurately. Including images with various levels of occlusion helps the model learn to recognize vine ends even when they are partially obscured.

Terrain and Vineyard Layout. The vineyard in this research is situated in hilly terrain and is irregularly shaped. This introduces additional complexity to the navigation system.

Given these challenges, it was necessary to capture data under different weather conditions along with seasonal data. This comprehensive data collection approach ensured the dataset included diverse scenarios, which is crucial for developing a robust navigation algorithm.

In Fig. 4.3, MV data collected during different weather conditions and seasonal changes is illustrated. These recorded images served as the basis for subsequent analysis and algorithm development. A total of 2,800 images were utilized to develop the datasets necessary for this research. This comprehensive data acquisition process ensured that the machine vision system could operate effectively across a wide range of environmental conditions, enhancing its robustness and reliability in real-world vineyard settings.

4.2.3. Preprocessing: Camera calibration

After data acquisition, the initial phase of the entire process involved camera calibration, a critical step essential for ensuring the accuracy of subsequent procedures. The camera underwent calibration in two distinct steps, as outlined below.

4.2.3.1. Fisheye lens calibration

The camera calibration process follows the same methodology described in Section 3.2.2. The KODAK PIXPRO camera, equipped with a fisheye lens, inherently introduces distortion in the original image frames, as depicted in Fig. 4.4a. This

distortion presents challenges in accurately determining real-world distances during subsequent stages. However, through the calibration process, these distortions are corrected. The resulting undistorted image, shown in Fig. 4.4b, clearly illustrates the elimination of curvature, with straight supporting line posts and a level horizon providing a stark contrast to the curved appearance in the distorted input image. This correction is crucial for ensuring accurate data acquisition and analysis, as it aligns the visual input with the real-world geometry, facilitating the reliable detection of vine ends and subsequent navigation tasks.

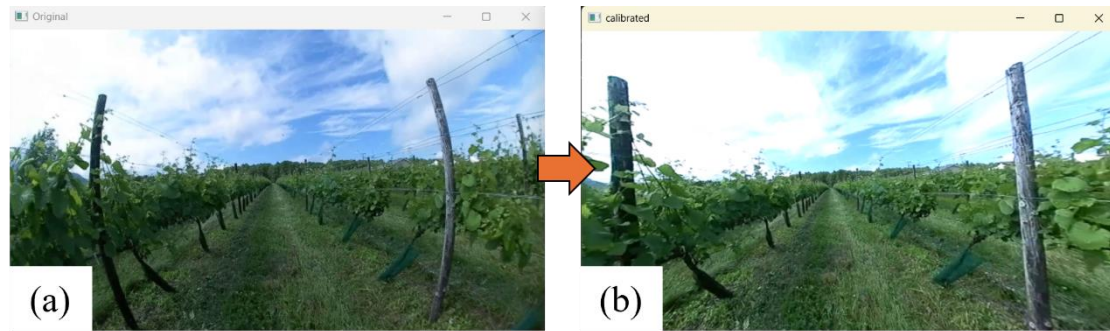


Figure 4.4. Fisheye lens calibration : (a) The original image displaying curvature, (b) The image post-lens calibration, devoid of curvature

4.2.3.2. Determining the co-relation between pixel dimensions and real-world distances

To elucidate the relationship between pixel dimensions and real-world distances for navigation accuracy, a second calibration procedure was conducted. This calibration is essential for ensuring the EV's precise navigation within the vineyard by accurately interpreting pixel data in terms of real-world measurements.

The primary role of the steering control for autonomous navigation is to keep the vehicle within the boundaries of the desirable path. This is achieved by adjusting the steering actuator to minimize the lateral error between the vehicle's position and a

target point in the generated reference path. The choice of the target point is critical; if it is the closest point to the vehicle, the actuator may struggle to minimize the lateral error at high speeds or sharp curves due to sensor and actuator delays. Hence, a look-ahead approach is employed to predict future lateral error at a chosen look-ahead distance (Atoui, H. et al., 2021). In this research, to generate look-ahead distances at varying speeds, the specific calibration mentioned in this section was performed. The target points are hereby referred to as look-ahead points.

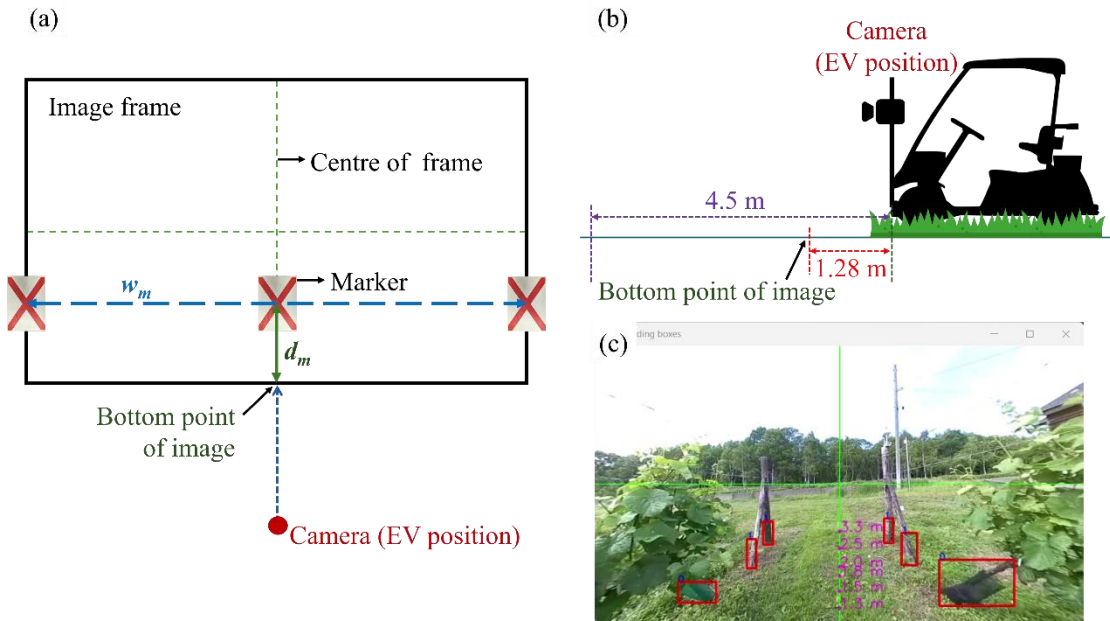


Figure 4.5. Mapping pixel dimensions to real-world distances : (a) Illustration of the process for measuring pixel and real-world dimensions using markers, (b) Schematic representation of the ground coordinates with respect to the EV, (c) Results obtained after fitting models to the manually measured data

The calibration process involved selecting six distinct look-ahead points, ranging from 1.28 m (positioned at the bottom center of the image frame) to 4.5 m. Both pixel and meter measurements of width were manually recorded at these six distances using markers, as shown in Fig. 4.5a. The look-ahead points were marked

along the centerline of the image frame, with a schematic representation provided in Fig. 4.5b. It is important to note that undistorted images obtained from the previous camera calibration step were utilized in this procedure.

Using the collected data, two linear regression models were trained to determine the correlation between (a) pixel height and look-ahead distance (in meters) and vice versa, essential for calculating lateral and heading errors, as well as for measuring distance, and (b) width (in meters) and look-ahead distance, aiding in width determination at specific distances and, consequently, the calculation of lateral error. Python's 'joblib' and 'scikit-learn' libraries were employed to train these models. These relationships will be elaborated upon in Section 4.2.6.4 (Calculation of Lateral Error) and Section 4.2.7 (Tree Row End Detection). The outcome of this calibration process is presented in Fig. 4.5c.

4.2.4. Dataset preparation

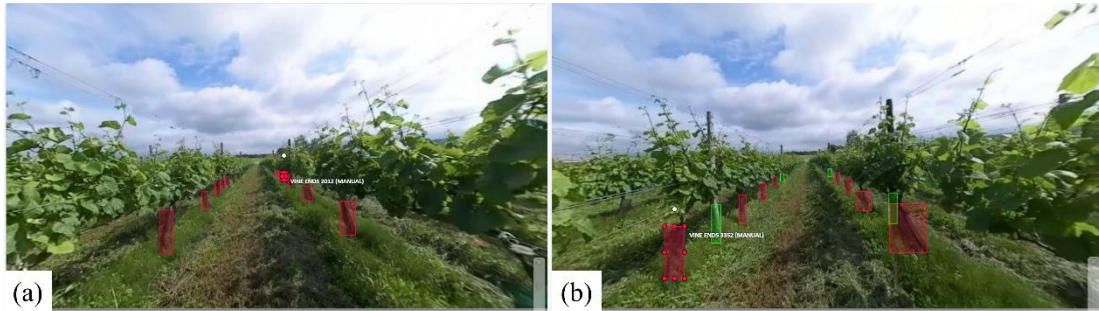


Figure 4.6. Dataset preparation : (a) Vineyard Dataset I with a single annotated class, (b) Vineyard Dataset II with two annotated classes

For annotation purposes, the Computer Vision Annotation Tool (CVAT), described in section 2.2.2, was employed to annotate the recorded images. This facilitated the identification and labeling of relevant features within the vineyard

environment. The precise annotation of images is crucial for training accurate machine vision models, particularly in diverse and complex environments such as vineyards.

Table 4.1

Features of Vineyard Dataset I and II

Dataset name	Number of images	Type of data	No. of data	Type of instances	Number of instances
Vineyard Dataset I	2500	Train data	2125	0: vine ends	24336
		Test data	375	0: vine ends	4254
Vineyard Dataset II	1323	Train data	1126	0: vine ends	9087
				1: line posts	2736
		Test data	197	0: vine ends	1539
				1: line posts	474

Two distinct datasets were then created to facilitate model training and evaluation using the images captured at Tsurunuma Winery, as depicted in section 4.2.2.

The details of the datasets are as follows:

Vineyard Dataset I. This dataset contained a total of 2,500 images, of which 2,125 were designated for training purposes. The dataset focused on a single class, 'vine ends' (class '0'), and included 24,336 instances of this class. The images in this dataset were annotated to highlight vine ends under various weather conditions and different stages of the growing season. The annotation process ensured that each vine end was accurately labeled, providing a robust dataset for training a machine vision model.

Vineyard Dataset II. This dataset comprised 1,323 images, capturing both vine ends and line posts. Out of these, 1,126 images were used for training. Vineyard Dataset II included two classes: vine ends (class '0') and line posts (class '1'). The inclusion of line posts aimed to assist in detecting the end of the vine rows. The dataset

encompassed a variety of weather conditions and canopy densities to ensure the model's robustness across different scenarios.

As depicted in Fig. 4.6 and summarized in Table 4.1, both datasets were curated to include a diverse range of conditions. The careful preparation and annotation of these datasets were integral to developing and evaluating the YOLOv8 model's performance in detecting vineyard features. This ultimately contributed to the autonomous navigation system's reliability and accuracy.

4.2.5. YOLOv8 model training

Several YOLOv8 models were trained using the datasets mentioned in the previous section. During the training process, key hyperparameters such as learning rate, batch size, and the number of epochs were fine-tuned to optimize the model's performance and prevent overfitting or underfitting. Additionally, advanced augmentation techniques such as mosaic and translate were employed. The rationale behind tuning these hyperparameters includes the following examples.

Mosaic augmentation combines multiple images into a single composite image, enhancing the dataset's diversity and aiding the model in generalizing better under varying lighting and environmental conditions.

Translate augmentation randomly shifts the position of images, adding variability to the training samples and helping the model adapt to object positioning variations in real-world vineyard environments.

The details of the model training are outlined in the following subsections.

4.2.5.1. YOLOv8 model training on Vineyard Dataset I

As mentioned, the initial dataset, denoted as Vineyard Dataset I, comprised 2500 images and focused on a single class (class '0') referred to as "vine ends" (Table

4.1). These end sections of vine trunks were selected as high-level feature objects due to their consistent appearance and position across different seasons and weather conditions. Annotation of this dataset involved the utilization of bounding boxes, as depicted in Fig. 4.6a, ensuring that the entirety of the vine trunk ends was encapsulated within each bounding box.

Initially, two YOLOv8 models, YOLOv8n and YOLOv8m, were trained on this Vineyard Dataset I. Among the available YOLOv8 detection models, YOLOv8n is recognized as the smallest and fastest variant. In contrast, YOLOv8m, while more accurate and larger, requires more computational resources and time. The dataset was split into train and test datasets. Specifically, the test dataset comprised 15% of randomly selected samples from Vineyard Dataset I. The trained YOLOv8m model will henceforth be referred to as YOLOv8m-vine for clarity, as it will be utilized in subsequent stages of this research.

4.2.5.2. YOLOv8 model training on Vineyard Dataset II

Vineyard Dataset II, comprising 1323 images, encompassed two distinct classes (Fig. 4.6b). Class '0' corresponds to the same category as the 'vine ends' class in Vineyard Dataset I. The second class (class '1') encompasses the line posts installed within the vineyard (Table 4.1). These line posts serve as essential fixtures in vineyards, providing structural support for the growth of grape vines and helping protect the plants against high winds.

The Vineyard Dataset II was utilized to train two additional YOLOv8 models. The first model was trained by using the weights of the pre-trained YOLOv8m-vine model trained with Vineyard Dataset I. The second model trained was the YOLOv8x model, known as the most accurate YOLOv8 model. Similar to the previous training

procedure, the dataset was split into train and test datasets, with 15% allocated to the test set for evaluation. Hereafter, the YOLOv8m-vine model trained with Vineyard Dataset II is referred to as YOLOv8m-vine-classes for clarity and consistency. The creation of Vineyard Dataset II serves to identify the termination point of a pathway; a concept elaborated upon in section 4.2.7 (Tree row end detection).

4.2.6. Path calculation algorithm

The path calculation algorithm consists of six steps. These steps are designed to generate lateral error and heading error, which are then sent to the autonomous navigation PC via a TCP (Transmission Control Protocol) socket, as depicted in Fig. 4.1. Each step plays a vital role in ensuring the vehicle's trajectory adheres to the desired navigation path. The detailed steps of the algorithm are outlined below in six subsections.

4.2.6.1. Capturing the image frames and removing distortion from images

This initial step involves capturing real-time images of the vineyard environment using the camera setup mentioned in section 4.2.1. The two calibration procedures described in section 4.2.3 are then applied to the captured image frames. These calibration techniques account for the distortion caused by the fisheye lens. By removing these distortions, the resulting images are undistorted, providing accurate pixel-to-real-world dimension data.

This process lays the foundation for effective feature detection and further path planning, ultimately contributing to the efficient autonomous navigation of electric vehicles in vineyard environments.

4.2.6.2. Application of trained YOLOv8 model for detection and localization

The trained YOLOv8 model from the previous section was applied to video streams to predict the classes and obtain bounding boxes. YOLOv8 provides bounding box location information in various formats. In this research, the format (center x, center y, width, height) is utilized to extract the locations of these bounding boxes. These location values represent the center point of the bounding box and its dimensions in terms of width and height. The bounding box coordinates denote the bounding box's position relative to the image frame.

These location values are then used to determine whether they belong to the left or right vineyard row relative to the EV. The middle of the image is assumed to represent the position of the EV, as the RGB camera is positioned in the middle section at the front of the EV (Fig. 4.2a). Consequently, any bounding box situated in the left half of the image is regarded as part of the left edge, and vice versa.

This assumption is based on the typical row spacing in vineyards, which ranges from 2 m to 4.5 m, with the vineyard in this study having a row spacing of 2.5 m. The width of the robot EV is approximately 1.7 m, closely aligning with the row spacing. To prevent accidental damage to vines, it is essential for the EV to operate consistently in the center of the row. Thus, every bounding box on the left side of the EV is considered to belong to the left row, and conversely for boxes on the right side.

4.2.6.3. Calculation of the reference path line

Following the determination of bounding box positions within the frame, the left edge points are identified by calculating the right bottom corner point of the bounding boxes, while the right edge points are determined by identifying the left bottom corner point. These points are then stored in two separate arrays. The process

continues by computing a regression line, which serves as the best-fit line for both the left and right edge points. To compute the regression lines, the slope (m) and intercept (b) are calculated using the following equations.

$$m = \frac{n(\sum x_e y_e) - (\sum x_e)(\sum y_e)}{n(\sum x_e^2) - (\sum x_e)^2} \quad (4.1)$$

$$b = \frac{\sum y_e - m \sum x_e}{n} \quad (4.2)$$

Here, x_e and y_e are the coordinates of the edge points. n represents the number of edge points in the array. Σ represents the sum of all the values in a certain array.

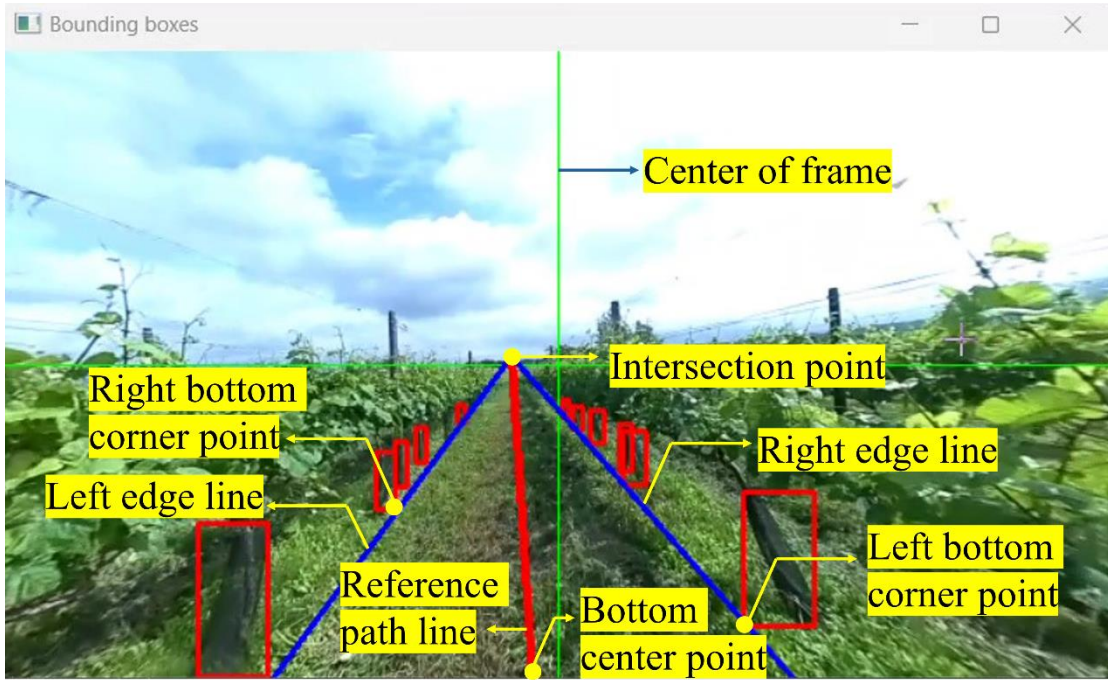


Figure 4.7. Schematic representation of key elements required for determining the reference path line

The intersection point (ix, iy) of these two straight lines is then computed using their respective slope and intercept values with the following equations:

$$ix = \frac{(b_l - b_r)}{(m_r - m_l)} \quad (4.3)$$

$$iy = (m_l * ix) + b_l \quad (4.4)$$

Here, m_l and m_r represent the slopes of the left and right edge lines, respectively. Similarly, b_l and b_r represent the intercepts of the left and right edge lines, respectively.

After this calculation, the bottom center point of the image is determined by averaging the bottommost points of the left and right edge lines using the following equation.

$$p_b = ((h - b/m), h) \quad (4.5)$$

Here, p_b represents the bottommost point of left or right edge lines, and h represents the height of the image frame.

Subsequently, the intersection point and the bottom center point are connected to establish the path of the robot EV, henceforth referred to as the reference path line. The points and lines calculated thus far are depicted in Fig. 4.7.

The left and right edge arrays, used to calculate the left and right edge lines, can store up to 30 points each. While a maximum of 10 points (on each side) are detected within one image frame during each instance, points from previous frames are stored alongside the newly detected points. When the arrays become full, older points are removed to append newly detected points. This strategy aims to minimize detection noise during the calculation of the left and right edges. Additionally, at the end of each row, when only one point (i.e., one line post) is detected on each side, the presence of old points in the array helps with the edge line calculation.

4.2.6.4. Calculation of lateral error

As depicted in Fig. 4.8, the center of the frame is assumed to represent the EV's location, and lateral error is calculated with reference to this center line. Initially, the

position of the look-ahead point is determined, after which the horizontal pixel difference between this point and the frame's center is calculated. This horizontal pixel distance represents the lateral error. If the look-ahead point resides on the left side of the frame's center, the lateral error is deemed positive, and conversely, if it resides on the right side, the lateral error is negative.

The determination of the look-ahead point depends on the look-ahead distance l (Fig. 4.8). As outlined in section 4.2.3.2, the correlation between the look-ahead distance (in meters) and the vertical coordinates of the image (pixel height) has already been established. Using this relationship, the pixel height is first calculated based on the look-ahead distance. This height is then used to precisely locate the look-ahead point on the reference path line (Fig. 4.7). This point serves as the base for calculating the lateral error as well as the heading error, as detailed in the following section. Therefore, lateral error is calculated by the following equation.

$$d_{pix} = (w/2) - x_{la} \quad (4.6)$$

Here, d_{pix} represents the lateral error in pixel value, w is the width of the image frame, and x_{la} is the horizontal coordinate of the look-ahead point.

Ultimately, the lateral error value in pixels is converted to meters. To accomplish this, the second model described in section 4.2.3.2 is employed. This model allows the calculation of the width (in meters) at the desired look-ahead distance. The width, expressed in meters, corresponds to the pixel width of the image frame.

$$d = d_{pix} * X_s \quad (4.7)$$

$$X_s = X_a/w \quad (4.8)$$

In Eq. 4.7, d is the lateral error in meters. X_s is the ratio of the actual real-world width X_a (in meters) to the pixel width w . X_a is dependent on the look-ahead distance

and is calculated by using the second model described in section 4.2.3.2. Therefore, by utilizing these equations, the lateral error in pixels is converted to a lateral error in meters.

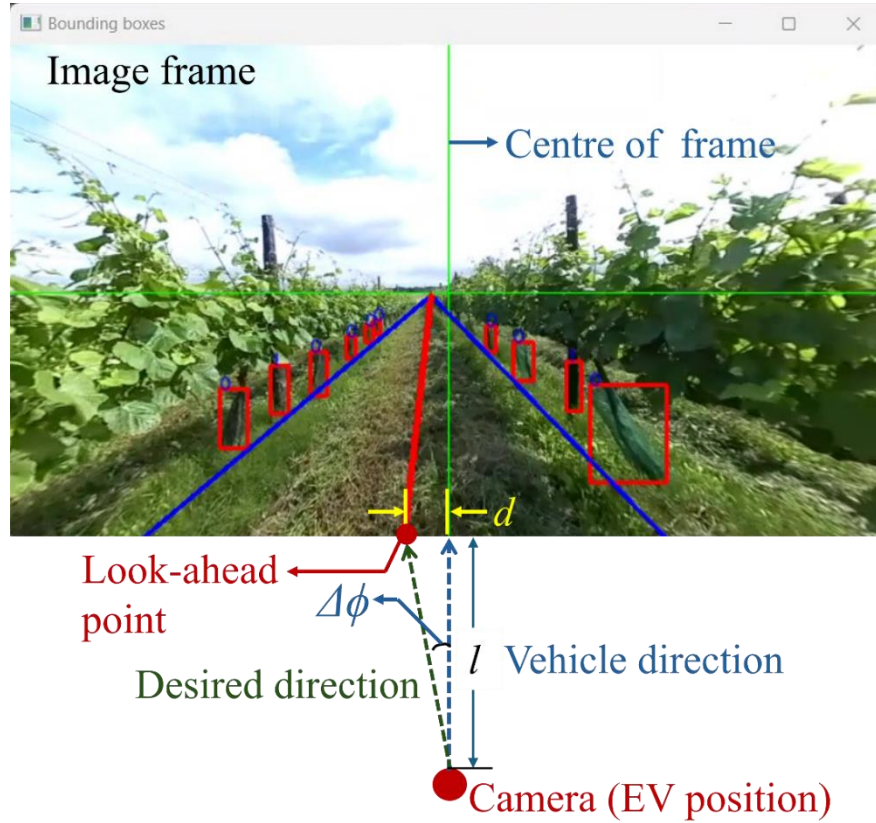


Figure 4.8. Schematic representation of essential factors involved in the calculation of lateral error and heading error

4.2.6.5. Calculation of heading error

The calculation of heading error focuses on determining the orientation or directional alignment of the vehicle relative to the target or desired trajectory. Heading error signifies the deviation or angle difference between the actual heading of the vehicle and the optimal path or trajectory it aims to follow.

In this context, the vehicle's orientation or heading is perceived to be in line with the centerline of the image frame, given the positioning of the RGB camera at the

front and center of the EV. Thus, the desired heading direction is defined by the alignment of the look-ahead point relative to the EV's position. The heading error, represented as $\Delta\phi$ (Figure 4.8), is calculated by measuring the angular deviation between these two points using the following equation.

$$\Delta\phi = \frac{\text{atan2}(l, d) * 180}{\pi} \quad (4.9)$$

Here, as mentioned before, l is the look-ahead distance (in meters), and d is the lateral error (in meters). The function 'atan2' is used to return the arctangent of the quotient of its arguments. It returns a numeric value between $-\pi$ and π representing the angle $\Delta\phi$ (heading error) in degrees. The number π is a mathematical constant that is the ratio of a circle's circumference to its diameter, approximately equal to 3.14159.

If the look-ahead point is positioned to the left of the EV's position relative to the centerline, the heading error is computed as a positive deviation. Conversely, if the look-ahead point is situated to the right of the centerline, the heading error will be a negative deviation. Accurate calculation of this heading error ensures that the EV maintains a proper course along the intended path, enhancing overall navigation precision and safety.

4.2.6.6. Calculation of steering angle

The calculation of lateral error (d) and heading error ($\Delta\phi$) is conducted in the machine vision PC onboard the robot EV (Fig. 4.2b). These values are then transmitted to the autonomous navigation PC (Fig. 4.2b) via a TCP socket. In the autonomous navigation PC, the steering angle (δ) is computed using Eq. 4.10, developed by Kise et al. (2001). The entire process, commencing from video capture to the calculation of lateral and heading errors, is executed in real time. These real-time calculations are then

forwarded to the autonomous navigation PC to facilitate the real-time navigation of the robot EV.

$$\delta = \alpha_s d + \beta_s \Delta \phi \quad (4.10)$$

Here, α_s and β_s represent the proportional gains used in the steering controller. The optimum values of α_s and β_s were obtained through field testing. The accurate calculation of the steering angle is crucial for ensuring that the robot EV follows the desired path with precision. By adjusting the steering angle in response to the calculated errors, the system can maintain alignment with the vineyard rows, thereby optimizing navigation efficiency and reducing the risk of collision with the vines.

4.2.7. Tree row end detection algorithm

The preceding section delineated the path calculation algorithm developed for navigating within vineyard rows. However, as the EV nears the end of a row, it becomes imperative to implement a stopping algorithm to prevent the EV from straying off the path. Therefore, a row-end detection algorithm was developed.

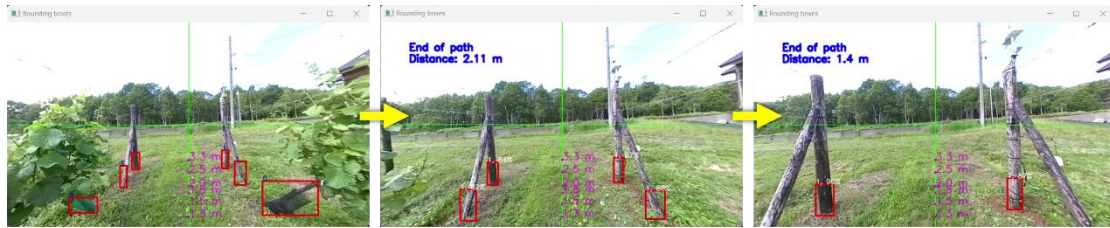


Figure 4.9. Tree row end detection and estimating the distance to the endpoint of the row

This algorithm relies on the presence of end posts positioned at the termination of each row. End posts, typically line posts fixed at the start and end of every vine row in conventional vineyards, serve as key markers. Notably, at the end of a vine row, no vine ends are visible within the image frame. Therefore, the algorithm was devised to

identify a minimum of two end posts (one on the left row and one on the right row) and no vine ends. To detect vine ends and end posts distinctly, the YOLOv8m-vine-classes model was employed.

Upon detecting the specified condition of two end posts and no vine ends, the algorithm generates a flag called 'End of row'. This flag initiates a continuous calculation of the distance from the EV's position to the end of the path. The distance is determined by calculating the vertical pixel position (pixel height) of the final end post and applying the pixel height-to-distance relationship described in Section 4.2.3.2.

Once the 'End of row' flag is set, the distance to the path's end is continuously calculated. This row-end detection algorithm is integral to the overall navigation system, ensuring the safe and efficient operation of the EV within the vineyard environment.

4.3. Experimental Setup

The experimental setup is designed to rigorously evaluate the performance of the proposed path calculation and tree row end detection algorithms within a real-world vineyard environment. This section details the specific characteristics of the vineyard test environment and the data collection methods employed during the experimental phase.

4.3.1. Description of the vineyard test environment

The vineyard selected for the experimental trials is the Tsurunuma Winery Co. vineyard located in the Hokkaido prefecture of Japan. This vineyard, one of the largest in Japan, spans a vast 447 hectares and cultivates approximately 40 grape varieties annually. Situated at an elevation ranging from 80 to 130 meters above sea level, the vineyard provides a diverse and representative environment for evaluating the proposed algorithms.

Figure 4.10a illustrates the specific section of the vineyard chosen as the experimental area, depicted from a bird's-eye view. This particular section was selected due to the absence of tall obstacles, such as trees, ensuring the unobstructed operation of the RTK-GNSS. It's important to note that RTK-GNSS was not utilized for navigation; its role was confined solely to evaluation tasks. The experimental area encompasses 15 rows of grapevines, offering a sufficient number of rows to thoroughly test the path calculation and row end detection algorithms under varying conditions.

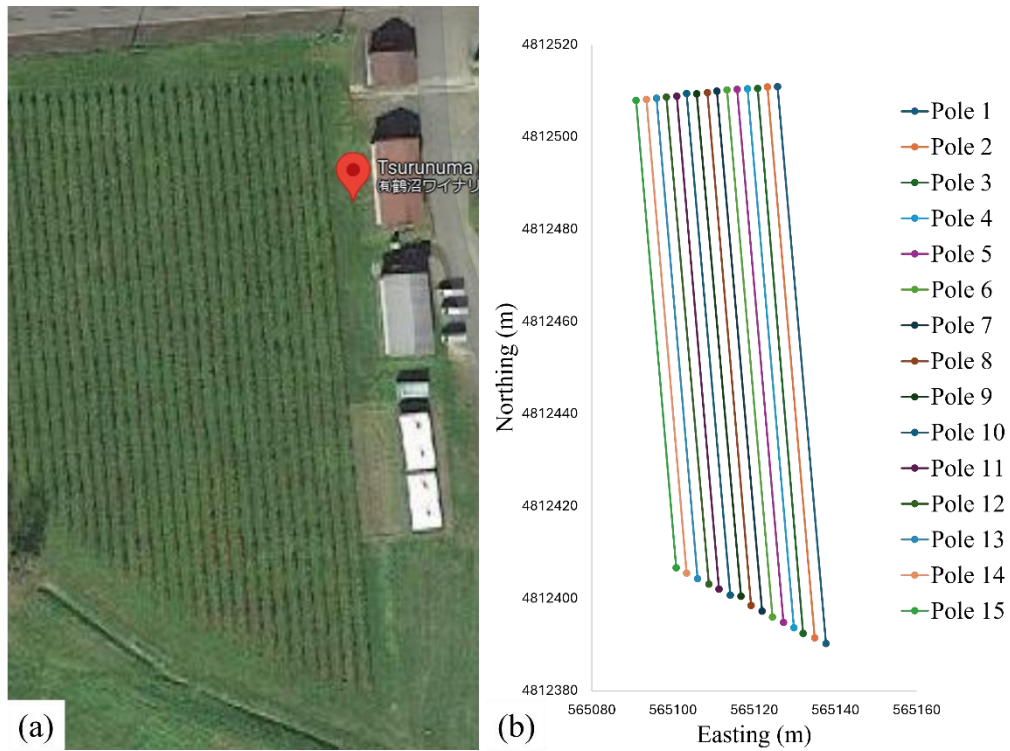


Figure 4.10. Vineyard test area : (a) in bird's eye view, (b) in UTM Coordinates

The UTM (Universal Transverse Mercator) coordinates, specifying the northing and easting positions of the grapevine rows, are presented in Figure 4.10b. This georeferencing is crucial for correlating the visual data captured by the RGB camera with the physical layout of the vineyard.

The vineyard exhibits an irregular structure, as seen in Figure 4.10a. Unlike conventional vineyards where rows are uniform in length, the rows in this vineyard vary in length due to the natural contours and shape of the land area. This irregularity presents additional challenges for the navigation system, requiring it to adapt to varying distances and orientations of the vine rows. Overall, the Tsurunuma Winery Co. vineyard in Hokkaido provides an ideal test environment for the experimental trials.

4.3.2. Data collection methods

Initially, row edge data were recorded manually, focusing on the positions of the end posts, with the assistance of an RTK-GNSS receiver (Alloy, Trimble Inc., USA). These data points were carefully recorded to ensure precise localization within the vineyard, facilitating the accurate mapping of the vineyard rows and end posts. Subsequently, these recorded data points were connected to form straight edge lines, as depicted in Figure 4.10b. Each line was identified, with "Pole 1" corresponding to the end post of vine row 1, and so on.

After determining the edge lines, the centerline of each row was calculated using Python programming. These centerlines are essential for evaluating the accuracy of the MV-based lateral and heading error algorithms. The calculation of the centerlines involved analyzing the positions of the end posts, identifying their respective coordinates, and determining the midpoint between them.

The data acquisition method for the MV system was previously described in Section 4.2.2. The video footage and image data were captured under real-world vineyard conditions to ensure that the experimental trials accurately reflect the operational challenges encountered during autonomous navigation.

4.4. Results and Discussion

This section presents the results and a discussion of the experiments conducted to assess the performance of the MV system in autonomous navigation of the vineyard environment. The evaluation encompasses training results of the YOLOv8 model and field testing under diverse conditions to validate the system's effectiveness.

4.4.1. YOLOv8 model training results

As described in section 4.2.5, four YOLOv8 models were trained, and the outcomes are presented in Table 4.2. The evaluation of these models is based on several metrics. Precision (P') assesses the accuracy of predictions (Eq. 4.11), while recall (R') quantifies the total number of predictions in relation to the ground truth (Eq. 4.12). Mean Average Precision (mAP) (Eq. 4.14) serves as a crucial metric widely adopted in computer vision research to assess the effectiveness of object detection models. mAP effectively balances precision and recall, optimizing both metrics simultaneously. The evaluation of true and false positives in the object detection task employs the Intersection over Union (IoU) threshold. IoU calculates the degree of overlap between two boundaries, thereby measuring the extent to which the predicted boundary aligns with the ground truth (i.e., the actual object boundary). In certain datasets, a predefined IoU threshold (e.g., 0.5) is utilized to classify predictions as true positives or false positives. In this study, mAP over an IoU of 0.5, denoted as mAP50, is employed for assessment.

$$P' = \frac{TP}{TP + FP} \quad (4.11)$$

$$R' = \frac{TP}{TP + FN} \quad (4.12)$$

$$AP = \int_0^1 P'(R')dR' \quad (4.13)$$

$$mAP = \frac{AP_1 + AP_2 + \dots + AP_{n_c}}{n_c} \quad (4.14)$$

Here, TP denotes the number of true positives (i.e., correctly classified positive samples), FN represents false negatives (i.e., positive samples incorrectly classified as negative), and FP represents false positives (i.e., negative samples incorrectly classified as positive). AP is computed as the area under the Precision-Recall (P - R) curve, integrating Precision and Recall. Here, n_c represents the number of classes. In the case of the YOLOv8n and YOLOv8m-vine models, where only one class (vine ends) exists, n_c equals 1. Consequently, in such instances, AP and mAP values are equivalent.

Table 4.2

Performance Metrics of YOLOv8 Training

Model	Class	Precision	Recall	mAP50
YOLOv8n	0: vine ends	0.917	0.776	0.877
YOLOv8m-vine	0: vine ends	0.907	0.812	0.897
YOLOv8m-vine-classes	all	0.95	0.872	0.937
	0: vine ends	0.921	0.858	0.93
	1: line posts	0.979	0.885	0.944
YOLOv8x	all	0.941	0.88	0.94
	0: vine ends	0.908	0.876	0.932
	1: line posts	0.974	0.884	0.949

Table 4.2 provides an overview of the training results. It illustrates that both YOLOv8n and YOLOv8m-vine models, each with one class (class 0: ‘vine ends’), exhibit comparable Precision values of 91.7% and 90.7%, respectively. However, the YOLOv8m-vine model demonstrates higher Recall (81.2%) and consequently a higher mAP50 value (89.7%). In this study, Recall holds significant importance as it ensures the correct identification of all positively detected instances of a specific class. Therefore, the YOLOv8m-vine model is preferred over the YOLOv8n model to train the YOLOv8m-vine-classes model and to conduct real-time field testing.

For the two-class models, YOLOv8m-vine-classes and YOLOv8x, exhibit remarkably similar performance metrics (Table 4.2). The yolov8m-vine-classes model demonstrates an overall precision of 95%, an overall recall of 87.2%, and an overall mAP50 value of 93.7%. Given the comparable results between the two models, the YOLOv8m-vine-classes model is favored due to its smaller size and lower computational demands compared to the YOLOv8x model.

4.4.2. Real-time field testing

Real-time field experiments were conducted under various weather conditions and canopy states to assess the performance of the MV system utilizing the YOLOv8m-vine model as the primary detection model. Only the tree row end detection tests used the YOLOv8m-vine-classes model. The system’s efficiency and robustness across diverse vineyard environments were evaluated, focusing on the ability to navigate autonomously and detect vine end posts effectively.

During these field trials, the operational speed was maintained at 3 km/h, which is optimal for autonomous navigation tasks such as spraying and weeding. This pace

allows for accurate detection and response to environmental and spatial cues while minimizing disruption to vineyard activities.

4.4.2.1. Cloudy weather conditions

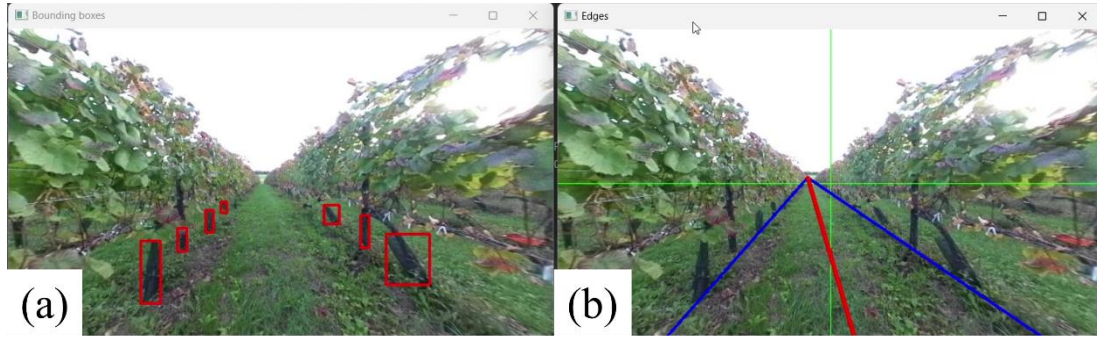


Figure 4.11. Experimental results visualization for Experiment 1: (a) Detected bounding boxes and (b) Calculated edge lines and reference path line

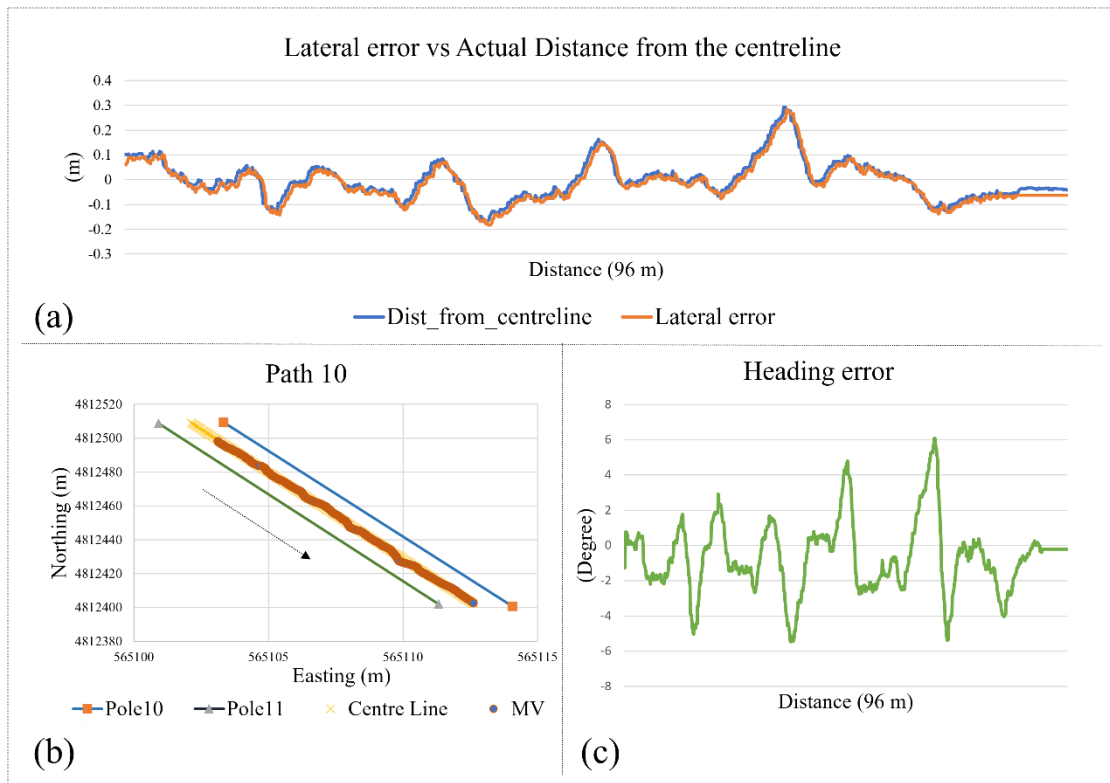


Figure 4.12. Experimental Results for Experiment 1: (a) Comparison between the calculated lateral error and the actual distance of the EV from the centerline, (b)

Visualization of the path traversed by the EV within path 10, and (c) Analysis of the heading error determined by the MV system during the experimental procedure

The experiment depicted in Fig. 4.12 was conducted under cloudy weather conditions. Figure 4.11 provides an overview of the canopy and weather conditions alongside real-time experimental outcomes, including bounding boxes and calculated edge lines. In Fig. 4.12b, the experiment took place along path 10, situated between vine rows 10 and 11, denoted by the blue and green lines, respectively. The yellow line illustrates the calculated center of the path, while the orange line represents the actual path traversed by the EV equipped with the MV-based navigation system. The EV's position was recorded using RTK-GNSS. The black arrow indicates the direction of navigation as the EV starts its journey from within the row and covers a total distance of 96 m until reaching the end of the path. The path trajectory exhibits predominantly straight movement, with minor oscillations observed at certain intervals.

In Fig. 4.12a, the blue line represents the actual distance of the EV from the centerline at various points, determined by measuring the distances between the orange and yellow lines in Fig. 4.12b. Meanwhile, the orange line in Fig. 4.12a illustrates the lateral error calculated by the machine MV algorithm. Comparison of these two lines provides insights into the accuracy of the lateral error calculation. Notably, in Fig. 4.12a, the alignment between the blue and orange lines indicates a high level of accuracy in the lateral error calculation. Furthermore, the root mean square error (RMSE) value of the lateral error is calculated to be 5.8 cm, closely approximating the 5 cm accuracy value of the RTK-GNSS-based navigation.

In Fig. 4.12c, the green line represents the heading error calculated by the MV system. Like the lateral error, the heading error is also influenced by the look-ahead

point, resulting in similar fluctuations depending on position, as observed in the lateral error values. The RMSE of the heading error is calculated to be 1.5 degrees, indicating a satisfactory level of accuracy for straight path navigation.

4.4.2.2. Variable weather conditions

This experiment was conducted in path 7, located between vine rows 7 and 8, covering a distance of 110 m. A significant aspect of this trial was the fluctuating weather conditions, alternating rapidly between sunny and cloudy, as depicted in Fig. 4.13. The results demonstrated the MV system's ability to operate effectively under variable weather conditions.

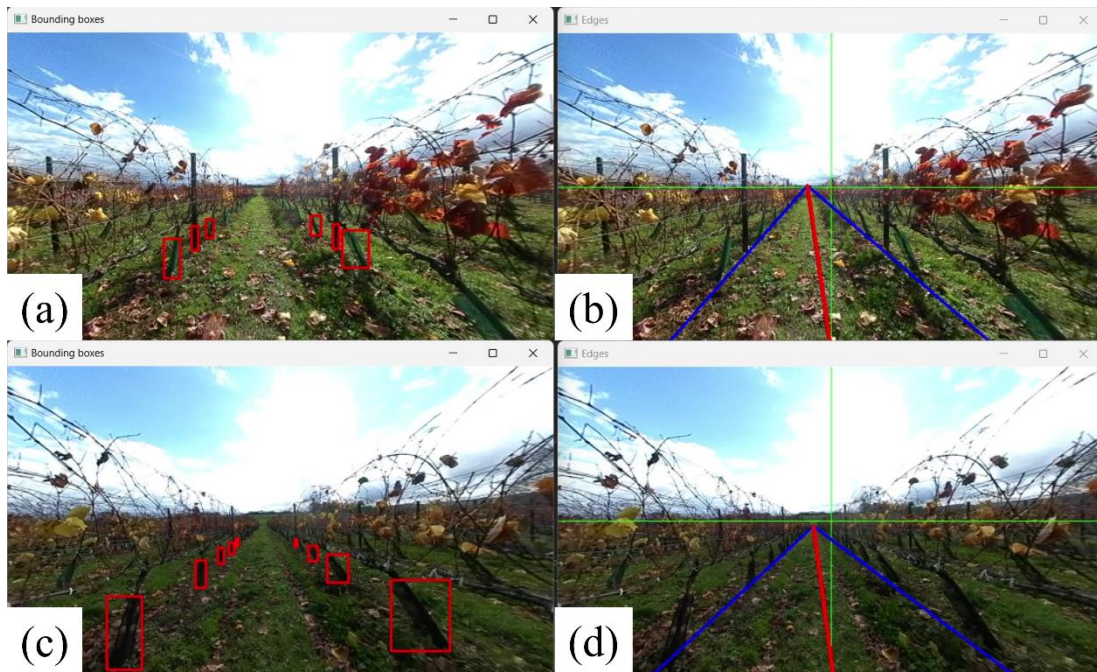


Figure 4.13. Experimental results visualization for Experiment 2: (a) and (c) Detected bounding boxes under sunny and cloudy weather conditions respectively, (b) Calculated edge lines and reference path line during sunny and cloudy weather conditions respectively

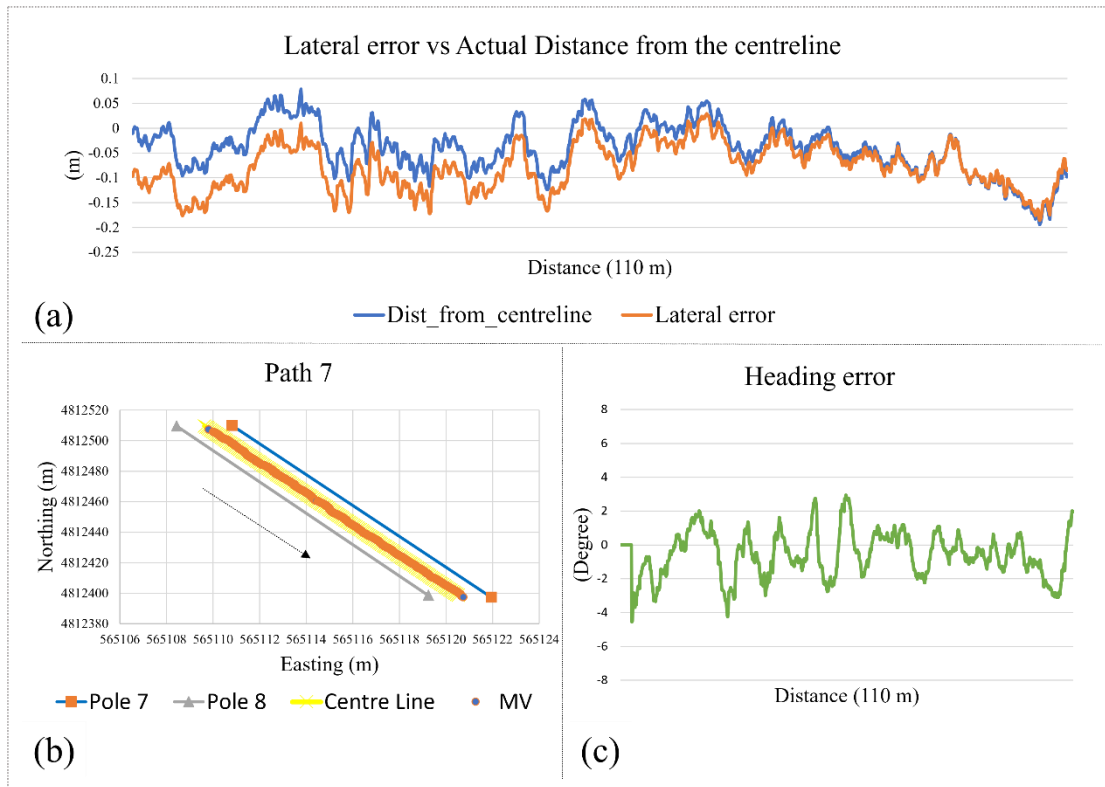


Figure 4.14. Experimental Results for Experiment 2: (a) Comparison between the calculated lateral error and the actual distance of the EV from the centerline, (b) Visualization of the path traversed by the EV within path 7, and (c) Analysis of the heading error determined by the MV system during the experimental procedure

The layout and interpretation of Fig. 4.14 follow the same format as Fig. 4.12. An analysis of Fig. 4.14a and 4.14b reveals that the EV maintained a straight trajectory throughout the course. The RMSE for lateral error was calculated to be 7.8 cm, aligning closely with the 5 cm accuracy threshold of RTK-GNSS. However, the initial segment of the path, as shown in Fig. 4.14a, exhibited a deviation in the accuracy of lateral error calculation compared to the previous experiment. An average discrepancy of 5 cm was noted between the actual position (blue line) and the calculated lateral error (orange line). Despite the absence of misdetections during this navigational test, the positioning

of bounding boxes and the computation of the optimal fit line may have influenced the overall lateral error calculations.

The calculated heading error values are presented in Fig. 4.14c. The RMSE for the heading error stood at 1.15 degrees, affirming its adequacy for navigation along straight paths.

4.4.2.3. Starting one meter ahead of the vine rows

The first two experiments validated the accuracy of the lateral and heading error calculations, demonstrating the capability of the MV system to guide the EV under varying weather and canopy conditions. The objective of the third experiment was to ascertain whether the EV could autonomously navigate into a vineyard row from an external starting point.

For this trial, the EV was positioned approximately 1 m from the first end post of path 7, with adjacent vine rows within its field of view. The experiment confirmed that the model accurately ignored the adjacent vine rows, allowing the EV to seamlessly enter and navigate the intended path without any detection errors (as illustrated in Fig. 4.15).



Figure 4.15. Experimental results visualization for Experiment 3: (a) Detected bounding boxes and (b) Calculated edge lines and reference path line

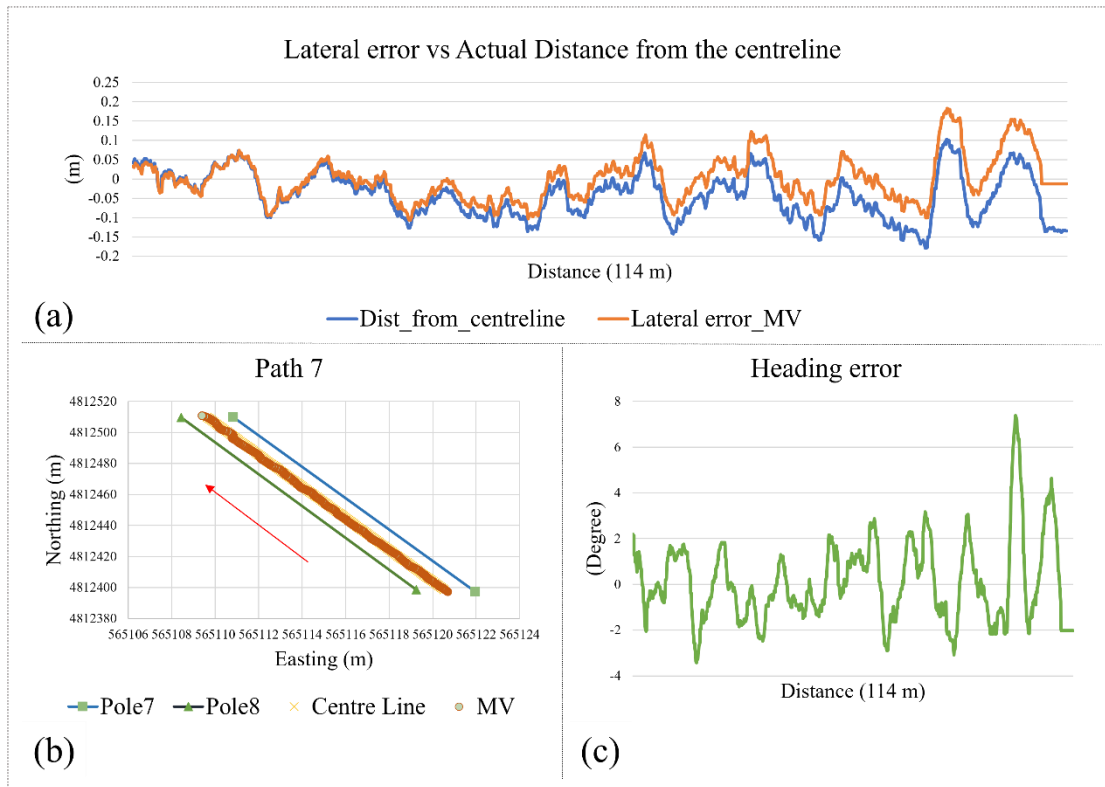


Figure 4.16. Experimental Results for Experiment 3: (a) Comparison between the calculated lateral error and the actual distance of the EV from the centerline, (b) Visualization of the path traversed by the EV within path 7, and (c) Analysis of the heading error determined by the MV system during the experimental procedure

Throughout this experiment, conducted along path 7 under predominantly cloudy weather conditions, the EV covered a distance of 114 m. The data representation in Fig. 4.16 is similar to that in Fig. 4.14. Analysis of Fig. 4.16a and 4.16b reveals that the EV maintained a straight trajectory. The RMSE for lateral error was measured at 4.3 cm, falling within the desired accuracy threshold of 5 cm. Nonetheless, similar to the previous experiment, there was an average discrepancy of approximately 6 cm between the actual position (blue line) and the calculated lateral error (orange line) along the latter half of the path (Fig. 4.16a). The absence of misdetections suggests that the variance may stem from factors such as bounding box

alignment and line fitting, which in turn impacted the precision of lateral error estimations.

The computed heading error, as shown in Fig. 4.16c, yielded an RMSE of 1.35 degrees. This level of accuracy supported the successful straight-line navigation of the EV, further confirming the efficacy of the MV system under the conditions tested.

4.4.2.4. Tree row end detection tests

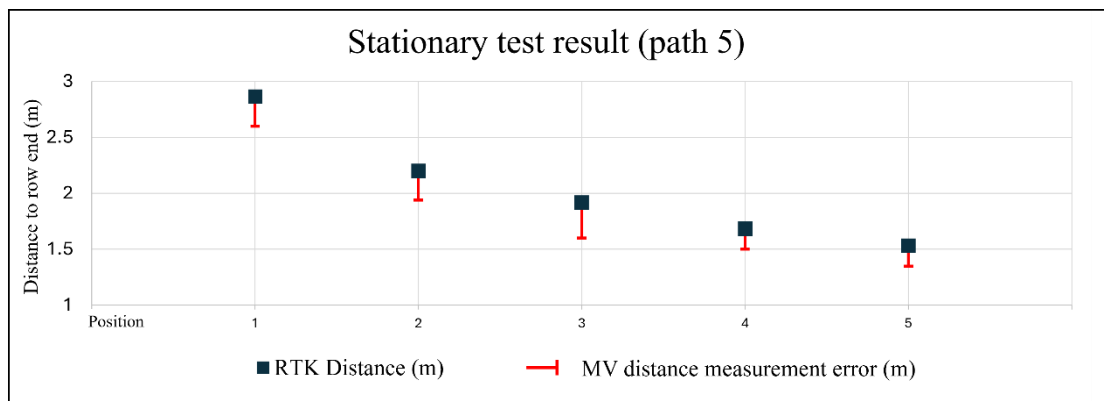


Figure 4.17. Experimental results obtained from stationary tests

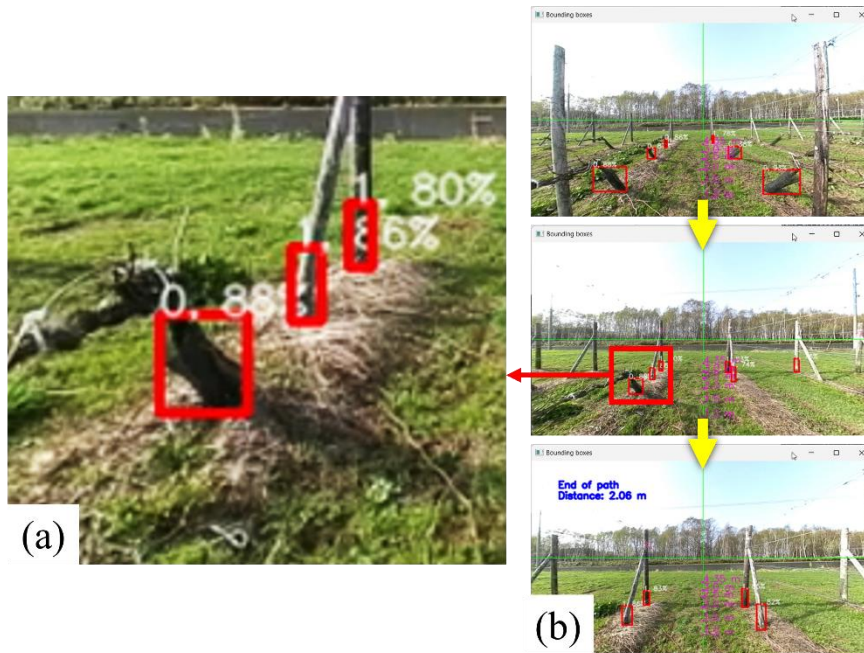


Figure 4.18. Experimental results: (a) Real-time classification of vine ends (class '0') and line posts (class '1') by the 'YOLOv8m-vine-classes' model, (b) Real-time distance measurement within path 7

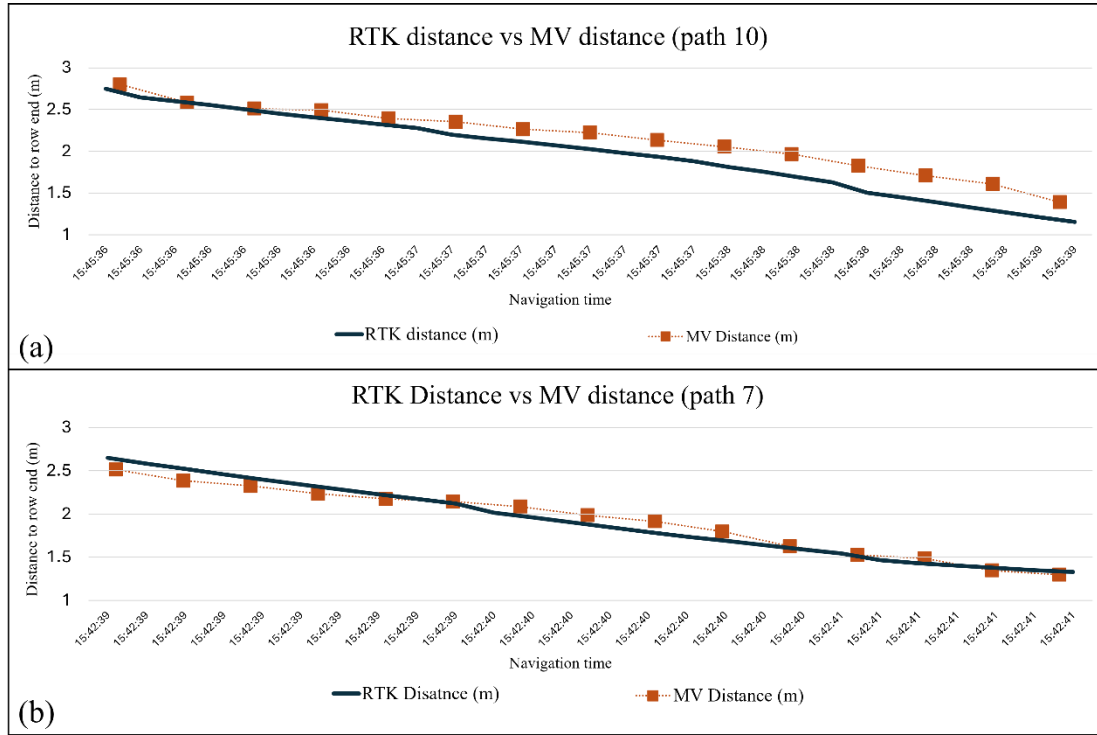


Figure 4.19. Comparison of experimental results obtained from driving tests within (a) path 10, and (b) path 7

To evaluate the accuracy of tree row end detection, two types of tests were carried out using the 'YOLOv8m-vine-classes' model. First, stationary tests were conducted where the EV was positioned near the end of the tree row, and the MV system calculated the distance to the end of the row. Simultaneously, the EV position was recorded using RTK GNSS and utilized to calculate the distance to the end of the tree row. Fig. 4.17 presents the results, showing a close alignment between the MV distances and the RTK distances.

Following the stationary tests, a real-time driving test was performed to examine two aspects: firstly, to determine whether the MV system could generate the 'End of row' flag and calculate distances in real time as the EV moved; and secondly, to compare these calculated distances with the RTK distances obtained by tracking the EV's position during the driving test.

The tests demonstrated that the 'End of row' flag is generated immediately once the vine ends exit the image frame. The initial distance measured ranged from 2.5 m to 3.0 m, which is suitable for stopping the EV within the next 2–3 seconds approximately 1 m from the end of the row. This 1 m distance is significant, as it ensures that the end posts remain in the image frame and that the EV can safely stop before the end of the row. Fig. 4.18 provides an example of real-time flag generation and distance measurement carried out along path 7. Both the vine-end and line post classes were accurately detected during these experiments, leading to precise end-of-row distance calculations.

Figs. 4.19a and 4.19b display the results of tests conducted along paths 10 and 7, respectively. As depicted in Fig. 4.19a, the calculated distances closely match the RTK distances, with a maximum error of 0.27 m. Similarly, in Fig. 4.19b, the accuracy of the calculated distances was notable, with a maximum error of 0.18 m. These errors were primarily due to the unstructured vineyard environment and the MV system's reliance on the last end post's position to compute the distance. Conversely, the RTK distance is calculated using the EV's position and the midpoint of the two end posts (as manually determining the end-of-row point can be challenging).

Overall, the results suggest that this method effectively halts the EV at the end of the row and facilitates the transition to the subsequent row.

4.5. Conclusion

The research presented in this chapter successfully demonstrates the training and application of YOLOv8 models for detecting vine ends in vineyard environments, a critical component for autonomous EV navigation. Among the models evaluated, the YOLOv8m-vine-classes model proved to be the optimal choice, offering a favorable balance of accuracy and computational efficiency. Real-time field testing further confirmed the system's robust performance across a variety of environmental conditions, effectively calculating lateral and heading errors for EV navigation.

Experiments conducted in different weather and canopy states affirmed the MV system's reliability and precision, with the EV achieving navigation accuracy comparable to RTK-GNSS-based navigation. The ability to autonomously navigate vineyard rows from external start points also highlights the system's practical application potential in precision agriculture.

Additionally, the ability to detect the end of a row and calculate the distance to the final end post allows the EV to execute a controlled deceleration and stop safely. This capability ensures the system can transition smoothly to the next row and supports the system's safe operation during autonomous navigation.

These outcomes validate the effectiveness of integrating advanced computer vision techniques with agricultural machinery, promoting enhanced autonomy in agricultural tasks and advancing sustainable farming practices. The research marks the initial steps towards full machine vision-based autonomous navigation in vineyards, with future work aiming to enable EVs to navigate between rows and explore broader applications in precision agriculture.

CHAPTER 5. CONCLUSION

5.1. Summary of the Findings

This dissertation presents a comprehensive study on the development and application of MV systems for autonomous navigation of agricultural vehicles. The research spans several critical aspects, from background and hardware setup to experimental validation and results.

Chapter 1 outlines the pressing issue of agricultural labor shortages in Japan and the potential of automation in addressing this challenge. It discusses the limitations of existing navigation technologies, such as RTK-GNSS and IMU, and highlights advancements in machine vision, particularly deep learning models like YOLOv8, which offer promising solutions for autonomous navigation.

Chapter 2 details the hardware and software components of the research platform. It describes the autonomous agricultural vehicle, including the robot tractor and EV, and the features of the vision sensor (camera) and processing computer. The software setup includes the Python programming language, the Computer Vision Annotation Tool (CVAT), and the YOLOv8 model, which were essential for developing the MV system.

Chapter 3 focuses on the design and implementation of a vision-based road detection system for an autonomous tractor. It covers the camera setup, image processing, and the road detection algorithm. When this algorithm was applied in real-time experimental runs, results showed that the lateral error was less than 0.2 m for unpaved roads and 0.4 m for paved roads. This demonstrates the system's capability to detect agricultural roads, suggesting it can be used in place of RTK-GNSS.

Chapter 4 details the system design and architecture for autonomous navigation in vineyards. It describes the camera setup, data acquisition, and preprocessing steps, including camera calibration and the determination of pixel-to-real-world distance correlations. The YOLOv8 model training process and the path calculation algorithm are explained, highlighting the steps for calculating lateral and heading errors and determining the steering angle. The chapter also introduces a tree row end detection algorithm, which is crucial for safe navigation.

The results from YOLOv8 model training and real-time field testing are presented, revealing the system's robust performance under various environmental conditions. Real-time field experiments underscore the system's effectiveness, with the RMSE for lateral error consistently near the 5 cm mark and that for heading error within 2 degrees. These findings indicate the system's adeptness at maintaining both proximity to the desired path and correct orientation throughout the navigation process.

Additionally, the system demonstrated robust performance in detecting the ends of tree rows and calculating the distance to the final end post. Tests showed that the 'End of row' flag is generated as soon as the vine ends exit the image frame, with initial distance measurements falling within the 2.5–3 m range. This ensures that the EV can safely stop within the next 2–3 seconds at a distance of approximately 1 m from the end of the row. The RMSE for distance measurement during these tests was within acceptable limits, validating the system's accuracy and reliability.

In Chapter 3, multiple image processing techniques were explored to ultimately calculate lateral error. Steps such as camera calibration and the determination of the relationship between pixel distance and real-world distance paved the way for the research presented in Chapter 4. The experimental results in Chapter 3 highlighted the

limitations of traditional MV-based navigation, which informed the integration of DL with MV for autonomous navigation in Chapter 4.

In conclusion, this research demonstrates the successful training and application of YOLOv8 models for vine end detection, an essential task for autonomous EV navigation in vineyard settings. The integration of DL with MV significantly enhances navigation accuracy and reliability, marking a substantial advancement in precision agriculture.

5.2. Contributions to the Field

One of the major goals of advancements in agricultural automation is to address the persistent labor shortages. However, fully autonomous agricultural vehicles face significant obstacles, particularly the dependence on RTK-GNSS signals, which can be obstructed by tall objects like trees or compromised by unstable internet connections. Since RTK-GNSS provides precise data essential for safe navigation, any alternative method must achieve similar levels of precision.

This research makes a significant contribution by developing a machine MV system designed to replace RTK-GNSS when its signal is unavailable. The experiments conducted, as detailed in the previous chapters, demonstrated the MV system's ability to achieve navigation precision comparable to RTK-GNSS, thus presenting a viable solution to one of the critical challenges in autonomous agricultural navigation.

While the system still has certain limitations, the research lays a solid foundation for further advancements in agricultural automation, especially in the field of machine vision. The MV techniques demonstrated in this study represent a benchmark development in the field of agricultural vehicle robotics. This work not only addresses current challenges but also opens avenues for future research aimed at

achieving full automation in agricultural settings, ultimately contributing to more efficient and sustainable farming practices.

5.3. Future Research

The research presented in Chapter 3 is grounded in traditional MV techniques and serves as a backup method when the RTK correction signal is unavailable. This method must be robust enough to achieve reliable navigation under such conditions. However, a system based solely on RGB images and traditional image processing faces several limitations. Factors such as variable light conditions, sudden growth of vegetation on the road, and large potholes can affect its performance. Additionally, the diversity of farm roads—ranging from gravel and mud to dirt and asphalt—necessitates a more universal detection procedure.

The research in Chapter 4 addresses some of these limitations by integrating various datasets recorded under different lighting, weather, and seasonal conditions. This chapter demonstrates substantial improvements in MV-based navigation by incorporating DL with MV, compared to traditional MV techniques alone. Despite these advancements, the differences between on-road and off-road navigation, particularly in terms of environmental features, suggest that integrating DL with MV might not entirely resolve the current limitations of traditional MV-based navigation. Therefore, future research will focus on addressing these limitations, with a significant direction being the integration of DL into the existing system.

Regarding the research in Chapter 4, it has shown promising results in tree row navigation and precise stopping at the end of each row. However, fully autonomous navigation can only be achieved when the EV can seamlessly navigate between rows and from the farmer's warehouse to the field. Current row transfer methods rely on path

maps to determine the next row (Yamasaki et al., 2022). Future work will explore integrating path maps with MV techniques to enhance this process. Additionally, further development of the road navigation algorithm presented in Chapter 3 could facilitate the EV's navigation from the farmer's home to the field.

In summary, future research will focus on overcoming the current limitations of traditional MV and DL integration, improving the robustness of the navigation system under various conditions, and achieving full autonomy in agricultural vehicle navigation. This includes refining the road navigation algorithm, integrating path maps with MV techniques, and expanding the system's adaptability to diverse farm road conditions.

References

- Ministry of Agriculture, Forestry and Fisheries. (2023). *Nōgyō rōdō-ryoku ni kansuru tōkei: Kikanteki nōgyō jūjisha (kōjin keiei-tai)* [Statistics on Agricultural Labor Force: Core agricultural workers (individuals)]. *Ministry of Agriculture, Forestry and Fisheries, Japan*. Retrieved from <https://www.maff.go.jp/j/tokei/sihyo/data/08.html>
- Anaconda, Inc. (2020). *Anaconda Documentation* [Computer software]. Available at <https://docs.anaconda.com/>
- Atoui, H., Milanés, V., Sename, O., & Martinez, J. J. (2021). Real-Time Look-Ahead Distance Optimization for Smooth and Robust Steering Control of Autonomous Vehicles. In *2021 29th Mediterranean Conference on Control and Automation (MED)*, (pp. 924-929). <https://doi.org/10.1109/MED51440.2021.9480235>
- Azimi, S., Abidin, M.S.Z., Emmanuel, A., & Hasan, H. (2020). Robotics and Automation in Agriculture: Present and Future Applications. *Applications of Modelling and Simulation*, 4, 130-140. Retrieved from https://arqiipubl.com/ojs/index.php/AMS_Journal/article/view/130
- Barawid, O.C., Mizushima, A., Ishii, K., & Noguchi, N. (2007). Development of an Autonomous Navigation System using a Two-dimensional Laser Scanner in an Orchard Application. *Biosystems Engineering*, 96(2), 139-149. <https://doi.org/10.1016/j.biosystemseng.2006.10.012>
- Chiu, K., & Lin, S. (2005). Lane detection using color-based segmentation. In *IEEE Proceedings Intelligent Vehicles Symposium*, (pp. 706-711). <https://doi.org/10.1109/IVS.2005.1505186>

- Coulter, R. C. (1992). *Implementation of the Pure Pursuit Path Tracking Algorithm*. (Publication No. CMU-RI-TR-92-01).
https://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf
- CVAT.ai Corporation. (2022). Computer Vision Annotation Tool (CVAT) (Version 2.2.0) [Computer software]. Available at <https://www.cvat.ai/>
- Deere & Company. (2024). The Future of Farming Technology. Retrieved from <https://www.deere.co.uk/en/agriculture/future-of-farming/>
- Gan, H., & Lee, W.S. (2018). Development of a Navigation System for a Smart Farm. *IFAC-PapersOnLine*, 51(17), 1-4. <https://doi.org/10.1016/j.ifacol.2018.08.051>
- Hu, M., Yang, W., Ren, M., Yang, J. (2004). A Vision Based Road Detection Algorithm. *IEEE Conference on Robotics, Automation and Mechatronics*, 2, 846-850.
<https://doi.org/10.1109/RAMECH.2004.1438028>
- Joubert, N., Reid, T. G. R., & Noble, F. (2020). Developments in Modern GNSS and Its Impact on Autonomous Vehicle Architectures. In *2020 IEEE Intelligent Vehicles Symposium (IV)* (pp. 2029-2036).
<https://doi.org/10.1109/IV47402.2020.9304840>
- Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLO (Version 8.0.0) [Computer software]. Available at <https://github.com/ultralytics/ultralytics>
- Kise, M., Noguchi, N., Ishii, K., & Terao, H. (2001). Development of the agricultural autonomous tractor with an RTK-GPS and a FOG. *Proceedings of the 4th IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2001)*, 34(19): 99–104.
[https://doi.org/10.1016/S1474-6670\(17\)33120-8](https://doi.org/10.1016/S1474-6670(17)33120-8)

- King, R. (2023, January 10). *Model structure of YOLOv8 detection models(P5) - yolov8n/s/m/l/x* [Photograph]. GitHub. Retrieved from <https://github.com/ultralytics/ultralytics/issues/189>
- KUBOTA Corporation. (2024). Kubota Concept Tractor. Retrieved from <https://www.kubota.co.jp/innovation/concept-tractor/index.html>
- Mccorkle, D. A., Dudensing, R. M., Hanselka, D. D., & Hellman, E. W. (2019). The long-term viability of US wine grape vineyards: assessing vineyard labour costs for future technology development. *International Journal of Entrepreneurship and Small Business*, 36(3), 308–334.
<https://doi.org/10.1504/IJESB.2019.097734>
- Mendez, E., Piña Camacho, J., Escobedo Cabello, J. A., & Gómez-Espinosa, A. (2023). Autonomous Navigation and Crop Row Detection in Vineyards Using Machine Vision with 2D Camera. *Automation*, 4(4), 309–326.
<https://doi.org/10.3390/automation4040018>
- Morales, Y., & Tsubouchi, T. (2007). DGPS, RTK-GPS and StarFire DGPS performance under tree shading environments. In *2007 IEEE International Conference on Integration Technology* (pp. 519 – 524).
<https://doi.org/10.1109/ICITECHNOLOGY.2007.4290370>
- Naïo Technologies. (2024). TED, the vineyard weeding robot. Retrieved from <https://www.naio-technologies.com/en/ted/>
- Nguyen, V., Seo, C., Kim, H., & Boo, K. (2017). A study on detection method of vehicle based on lane detection for a driver assistance system using a camera on highway. In *11th Asian Control Conference (ASCC)*, (pp. 424-429).
<https://doi.org/10.1109/ASCC.2017.8287207>

- Noguchi, N., Reid, J.F., Manager, Zhang, Q., Will, J.D., & Ishii, K. (2001). Development of Robot Tractor Based on RTK-GPS and Gyroscope. In *2001 ASAE Annual Meeting*. <https://doi.org/10.13031/2013.7297>
- Oliveira, L.F.P., Moreira, A.P., & Silva, M.F. (2021). Advances in Agricultural Robotics: A State-of-the-Art Review and Challenges Ahead. *Robotics*, 10(2), 52. <https://doi.org/10.3390/robotics10020052>
- O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., & Walsh, J. (2020). Deep Learning vs. Traditional Computer Vision. In: Arai, K., Kapoor, S. (Eds), *Advances in Computer Vision*, 943, 128–144. https://doi.org/10.1007/978-3-030-17795-9_10
- Python Software Foundation (2019). *Python Language Reference* (Version 3.7) [Computer software]. Available at <https://www.python.org/>
- Python Software Foundation (2021). *Python Language Reference* (Version 3.10) [Computer software]. Available at <https://www.python.org/>
- Roshanianfard, A., Noguchi, N., Okamoto, H., & Ishii, K. (2020). A review of autonomous agricultural vehicles (The experience of Hokkaido University). *Journal of Terramechanics*, 91, 155–183. <https://doi.org/10.1016/j.jterra.2020.06.006>
- Schneiderman, H. & Nashman, M. (1992). Visual processing for autonomous driving. In *Proceedings IEEE Workshop on Applications of Computer Vision*, (pp. 164-171). <https://doi.org/10.1109/ACV.1992.240315>
- Sharifi, M., & Chen, X. (2015). A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards. In *2015 6th International*

- Conference on Automation, Robotics and Applications (ICARA)*, (pp. 251–255).
<https://doi.org/10.1109/ICARA.2015.7081155>
- Sivaraman, S. & Trivedi, M.M. (2013). Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4), 1773-1795.
<https://doi.org/10.1109/TITS.2013.2266661>
- Stefas, N., Bayram, H., & Isler, V. (2019). Vision-based monitoring of orchards with UAVs. *Computers and Electronics in Agriculture*, 163.
<https://doi.org/10.1016/j.compag.2019.05.023>
- Takai, R., Yang, L. & Noguchi, N. (2014). Development of a crawler-type robot tractor using RTK-GPS and IMU. *Engineering in Agriculture, Environment and Food*, 7(4), 143–147. <https://doi.org/10.1016/j.eaef.2014.08.004>
- Wang, C., Zhang, G., Cheng, Z., & Zhou, W. (2023). Rethinking Low-Level Features for Interest Point Detection and Description. In: Wang, L., Gall, J., Chin, T.J., Sato, I., Chellappa, R. (Eds), *Computer Vision – ACCV 2022*, (pp. 108–123).
https://doi.org/10.1007/978-3-031-26284-5_7
- Wang, R., Xu, Y., Libin, & Zhao, Y. (2002). A vision-based road edge detection algorithm. In *Intelligent Vehicle Symposium*, 1, 141-147.
<https://doi.org/10.1109/IVS.2002.1187942>
- Yamasaki, Y., Hizatate, T., & Noguchi, N. (2021). Development of Electrical Vehicle Robot for Orchard Application. In *Proceedings of the European Conference on Agricultural Engineering AgEng2021*, (pp. 527-534). [AgEng2021-
Proceedings.pdf \(researchgate.net\)](https://www.researchgate.net/publication/354111111/figure/fig/1/figure-fig1/161111111/AgEng2021-Proceedings.pdf)

Yamasaki, Y., & Noguchi, N. (2022). Optimal route planning and a turning method for an electric vehicle robot in an irregularly shaped vineyard. *IFAC-PapersOnLine*, 55(32), 1-5. <https://doi.org/10.1016/j.ifacol.2022.11.105>