DEVELOPMENT OF A MODULAR AGRICULTURAL ROBOTIC SPRAYER

Paolo Rommel P. Sanchez
Rowan University

DEVELOPMENT OF A MODULAR AGRICULTURAL ROBOTIC SPRAYER

by

Paolo Rommel P. Sanchez

A Dissertation

Submitted to the
Department of Mechanical Engineering
College of Henry M. Rowan College of Engineering
In partial fulfillment of the requirement
For the degree of
Doctor of Philosophy in Mechanical Engineering
at
Rowan University
April 4, 2023

Dedications

For my family, Eden and Pi.

Acknowledgements

I express my deepest gratitude to Dr. Hong Zhang for his guidance and help throughout this research. The experiences, skills, and knowledge that I have gained are things that I will take with me into my next professional endeavor.

I also express my sincerest appreciation to Professor Shen-Shyang Ho for his guidance in developing and implementing this research's machine-learning components. I would also like to thank Dr. Shreekanth Mandayam, Dr. Mitja Trkov, and Dr. Wei Xue for sharing their knowledge and expertise and serving on my dissertation committee.

I also express my sincerest gratitude to the Engineering Research and Development for Technology under the Department of Science and Technology of the Republic of the Philippines for providing full financial assistance to pursue my degree. My sincerest thanks are also expressed to the University of the Philippines Los Baños, especially to the Agribiosystems Machinery and Power Engineering Division of the Institute of Agricultural and Biosystems Engineering, for their support in my study.

I thank the Filipino community in New Jersey, especially my dear friend Father Philip, for their assistance in my stay in the United States.

I also express my forever gratitude to my mother. Thank you for all your prayers and motivation. You are always a pillar that keeps me standing in the face of adversity.

Finally, I want to thank my wife, Eden, for her unwavering love and support through this endeavor. And to my son, Pi, I thank you for always being a source of joy in my life.

Abstract

Paolo Rommel P. Sanchez
DEVELOPMENT OF A MODULAR AGRICULTURAL ROBOTIC SPRAYER
2022-2023
Hong Zhang, Ph.D.
Doctor of Philosophy in Mechanical Engineering

Precision Agriculture (PA) increases farm productivity, reduces pollution, and minimizes input costs. However, the wide adoption of existing PA technologies for complex field operations, such as spraying, is slow due to high acquisition costs, low adaptability, and slow operating speed. In this study, we designed, built, optimized, and tested a Modular Agrochemical Precision Sprayer (MAPS), a robotic sprayer with an intelligent machine vision system (MVS). Our work focused on identifying and spraying on the targeted plants with low cost, high speed, and high accuracy in a remote, dynamic, and rugged environment. We first researched and benchmarked combinations of one-stage convolutional neural network (CNN) architectures with embedded or mobile hardware systems. Our analysis revealed that TensorRT-optimized SSD-MobilenetV1 on an Nvidia Jetson Nano provided sufficient plant detection performance with low cost and power consumption. We also developed an algorithm to determine the maximum operating velocity of a chosen CNN and hardware configuration through modeling and simulation. Based on these results, we developed a CNN-based MVS for real-time plant detection and velocity estimation. We implemented Robot Operating System (ROS) to integrate each module for easy expansion. We also developed a robust dynamic targeting algorithm to synchronize the spray operation with the robot motion, which will increase productivity significantly. The research proved to be successful. We built a MAPS with three independent vision and spray modules. In the lab test, the sprayer recognized and hit all targets with only 2% wrong sprays. In the field test with an unstructured crop layout, such as a broadcast-seeded soybean field, the MAPS also successfully sprayed all targets with only a 7% incorrect spray rate.

Table of Contents

Abstract v

List of Figures xi

List of Tables xvii

Chapter 1: Introduction 1

1.1 Background of the Study 1

1.2 Motivation of the Study 4

1.2.1 Plant Detection System 5

1.2.2 Valve Control System 8

1.2.3 Physical Configurations 8

1.2.4 Problem Statements 9

1.3 Significance of the Study 10

1.3.1 Hypothesis 1: Distributed and Parallel Processing with Cluster of Low-Power Devices 10

1.3.2 Hypothesis 2: Vision-Based Velocity Estimation and VTD Valve Control for In-Motion Spray 11

1.4 Objectives of the Study 12

1.5 Research Organization 15

1.6 Research Highlights 16

1.7 Research Contribution 17

1.8 Published Works 18

Chapter 2: Literature Review 20

2.1 Introduction 20

2.2 Field Agricultural Machinery 20

2.3 Agricultural Field Robots 21

Table of Contents (Continued)

2.4 Precision Sprayers 23

2.4.1 Non-Real-Time Precision Sprayers 25

2.4.2 Real-Time Precision Sprayers 26

2.5 MVS-CNN 39

2.5.1 One- and Two-Stage CNN Architectures 41

2.5.2 SSD MobileNet 42

2.5.3 YOLO 45

2.5.4 MVS-CNN in Weed Detection 47

2.6 Modularity 50

2.6.1 Module-Based Hardware 51

2.6.2 Robotics Software Framework 52

2.6.3 Modular Agricultural Robots 54

2.7 Review Summary 59

Chapter 3: Materials and Methods 61

3.1 Overview 61

3.2 Development of CNN-Based Vision Module 62

3.3 Development of the MAPS 63

3.3.1 Modular Hardware and Software Architecture 65

3.3.2 Vision-Based Velocity Estimation 70

3.3.3 Valve Control by Queuing of VTDs 71

3.3.4 Specification and Cost Summary 71

3.4 Field Testing 74

3.5 CNN Model Development 75

3.5.1 CNN Hardware 76

Table of Contents (Continued)

3.5.2 Dataset Preparation 76

3.5.3 Training and Validation 79

3.5.4 TensorRT Optimzation 80

3.5.5 Detection Performance 80

3.6 Statiscal Analysis 85

Chapter 4: Benchmarking of One-Stage CNN Object Detection Models for Weed Detection 86

4.1 Introduction 86

4.2 Methodology 88

4.2.1 Dataset Preparation 88

4.2.2 Training and Validation 91

4.2.3 Performance Testing 92

4.3 Results and Discussion 92

4.3.1 Training of Weed Detection Models 92

4.3.2 Detection Performance 95

4.3.3 Processing Time in Different Hardware Setups 102

4.3.4 Cost Analysis 107

4.3.5 Performance Summary 109

Chapter 5: Simulation-Aided Development of a Modular MVS-CNN for Plant Detection: Effect of Travel Velocity, Inference Speed, and Camera Configurations 113

5.1 Introduction 113

5.2 Materials and Methods 115

5.2.1 Concept 115

5.2.2 Field Map Modeling 120

Table of Contents (Continued)

5.2.3 Motion Modeling 121

5.2.4 Detection Algorithm 122

5.2.5 Experimental Design 124

5.2.6 Vision Module Development 125

5.3 Results and Discussion 130

5.3.1 Sensitivity Analysis 131

5.3.2 Effects of Travel Velocity and Inference Speed 132

5.3.3 Effect of Increasing *S* or Multiple Cameras 137

5.3.4 Vision Module Simulation and Testing Performance 139

5.3.5 Proposed Reference Chart 141

Chapter 6: Development a CNN-Based Precision Sprayer with Vision-Based Velocity Estimation and Valve Control Using Variable Time Delay 144

6.1 Introduction 144

6.2 Materials and Methods 145

6.2.1 Scalable Unit 145

6.2.2 CNN Model Development 157

6.2.3 Testing of Vision-Based Velocity Estimation 158

6.2.4 Sprayer Performance Testing 159

6.3 Results and Discussion 163

6.3.1 Velocity Measurement Calibration 163

6.3.2 Effect of Plant Height Factor 165

6.3.3 Effect of Tracking Distance Threshold 167

6.3.4 Spraying Performance 168

6.3.5 Summary of Observations 175

Table of Contents (Continued)

Chapter 7: Field Evaluation of a CNN-Based Modular Precision Sprayer 176

7.1 Introduction 176

7.2 Materials and Methods 177

7.2.1 Overview of the MAPS 177

7.2.2 Experimental Field 179

7.2.3 SSD-MB1 Training and Validation 180

7.2.4 Field Testing 181

7.3 Results and Discussion 184

7.3.1 CNN Model Performance 184

7.3.2 Targeting Performance 186

7.3.3 Spray Volume Reduction 190

7.3.4 Performance Summary 192

Chapter 8: Conclusion and Future Works 193

8.1 Conclusion 193

8.1.1 MVS-CNN Benchmarking 193

8.1.2 MVS-CNN Modeling 194

8.1.3 MAPS Development 195

8.1.4 Field Testing 196

8.2 Future Works 197

Appendix A: List of Acronyms, Abbreviations, Units, and Symbols 199

Appendix B: Vision Module Velocity Estimation Raw Calibration Data 206

Appendix C: Precision Sprayer Components 212

List of Figures

Figure Page

Figure 1. The Objectives of the Research with Respect to Identified Problems and Formulated Hypotheses 14

Figure 2. Types of Controls in Crop Protection 24

Figure 3. Common Weed Control Mechanisms (Adapted from [118]): (a) Torsion, (b) Finger, and (c) Pneumatic Blower 24

Figure 4. Examples of Agricultural Sprayers: (a) Knapsack Sprayer [81] and (b) Tractor-Drawn Precision Boom Sprayer [120] 25

Figure 5. Some Weed Detection Techniques Using UAV for Patch Spraying: (a) Segmentation [127], and (b) Bounding-Box [128] 26

Figure 6. Common Precision Sprayer Framework 27

Figure 7. Some Existing Vision-Based Precision Sprayers: (a) [120], (b) [133], (c) [59], (d) [79], (e) [75], (f) [73], (g) [135], (h) [137], (i) [102], and (j) [86], and (k) [76] 31

Figure 8. Distribution of Technology Types for Weed Detection from 16 Reviewed Studies 32

Figure 9. Types of Nozzle Layout with Respect to the Sensors 33

Figure 10. Distribution of Sensors Technologies for Location and Motion Sensing of Precision Sprayers from 16 Reviewed Studies 36

Figure 11. Distribution of Targeting and Spraying Algorithms for Weed Control from 16 Reviewed Studies 38

Figure 12. Convolutional Neural Network for Object Classification 40

Figure 13. General Architecture of a Modern CNN for Object Detection 42

Figure 14. Summary of mAP and Inference Speed of Selected Object Detection Architectures on COCO 2017 Dataset [151] 44

Figure 15. Comparison of YOLO Architectures with Other Object Detection Architectures 46

List of Figures (Continued)

Figure Page

Figure 16. A Generic Three-Level Modular Hardware Architecture 52

Figure 17. The Synchronous Communication of ROS Nodes Using Port Mechanism 53

Figure 18. The Asynchronous Communication of ROS Nodes Through the Topic Mechanism 54

Figure 19. Flowchart of the Development Process of the Modular Agricultural Robotic Sprayer 61

Figure 20. The Steps in the Benchmarking of Selected CNN Models on Different Hardware 62

Figure 21. Steps in Modeling the Detection Rate of an MVS-CNN for Plant Detection 63

Figure 22. The Precision Sprayer Prototype with the Push-Type (a) Base Unit and (b) Extended Configuration 64

Figure 23. The Specific Steps in the Development of the MAPS 65

Figure 24. Hardware Schematic of the Vision-Based Precision Sprayer with Modular Components 66

Figure 25. The Bottom View of the MAPS 66

Figure 26. General Software Architecture of the MAPS 67

Figure 27. Graphical User Interface of the MAPS 69

Figure 28. The Relative Positions of the Detection, Velocity Estimation, and Spray Regions of the MAPS 71

Figure 29. Cost Distribution of Each Component of the MAPS 73

Figure 30. The Broadcast-Seeded Soybean Field 74

Figure 31. Field Testing Flowchart 75

Figure 32. CNN Development Flowchart 76

List of Figures (Continued)

Figure Page

Figure 33. Sample Dataset Annotation Using LabelImg 77

Figure 34. Sample Raw Images from the Three Datasets Used in the Development of the MAPS 78

Figure 35. The General Procedure on CNN Model Optimization Using TensorRT in Nvidia Jetson Platform 80

Figure 36. Image Level Processing Times 83

Figure 37. Application Level Processing Times 84

Figure 38. Onion Plant Beds with Black Polyethylene Mulch 88

Figure 39. Cropped Image of a Mulched Onion Bed with Class Annotations 89

Figure 40. Training and Validation Losses of Tested CNN Models with the Vertical Broken Lines Representing the Epoch at Overfitting 93

Figure 41. A Sample Detections Using the Trained CNN Object Detection Models on a Mulched Onion Plot 96

Figure 42. Average (a) $mAP^{0.5}$ and (b) $mAP^{0.5:0.95}$ of Tested CNN Models on the Test Dataset 97

Figure 43. The Loading Time, in Seconds, of Trained CNN Models in Test CUDA Devices 103

Figure 44. The Inference Time, in Seconds, of Trained CNN Models in Test CUDA Devices 105

Figure 45. The Average Inference Speeds, in fps, of Trained CNN Models in Test CUDA Devices 107

Figure 46. Comparison of Cost Efficiency, in fps Per USD, of Trained CNN Models in Test CUDA Devices 109

Figure 47. Camera Mounting Location and Orientation in a Boom (Not Drawn in Scale) 116

List of Figures (Continued)

Figure Page

Figure 48. Cases of Gaps and Overlaps in Vision-Based Plant Detection at Different Values of r_o 118

Figure 49. Illustration of Multiple Adjacent Cameras for Plant Detection at $n_{vis} = 2$ or $S_{eff} = 2S$ 120

Figure 50. Virtual Field with Map and Motion Modeling Parameters 121

Figure 51. Software Framework of the Vision Module 128

Figure 52. Laboratory Test Setup of the CNN-Based Vision Module 130

Figure 53. The Theoretical (Solid) and Simulated (Broken-Line) Detection Rates of the Virtual Vision Module 131

Figure 54. Simulated Plant Hill Detection Rates of the Vision Module Different Velocities 133

Figure 55. Simulated Plant Hill Detection Rates of the Vision Module Different Inference Speeds 133

Figure 56. Detected and Undetected (Broken Red Lines) Plant Hills on Each Frame and Detection Pattern (Blue Broken Lines) 136

Figure 57. Theoretical Maximum Travel Velocity to Prevent Missed Detections at Different Number of Cameras and Inference Speeds 137

Figure 58. Detected and Undetected (Broken Red Lines) Plant Hills on Each Frame Along the First 10 m at $fps = 2.4$ 138

Figure 59. Sample Real-Time Inferencing Using Trained SSD MobileNetV1 Model and Optimized in TensorRT 139

Figure 60. Proposed Reference Chart in Designing an MVS-CNN for Plant Detection 142

Figure 61. Workflow of an SU of the MAPS 145

Figure 62. The Bottom View of the Sprayer Showing the SU Components 146

Figure 63. Sample Weed and Crop Detections Using SSD-MB1 147

List of Figures (Continued)

Figure Page

Figure 64. Distances Used in Tracking Plants 148

Figure 65. Sample Image of Crop and Weed with Detections, Velocity Measurements, and Extents of the Location and Velocity Estimation Region (Red Line) 150

Figure 66. Distance Nomenclature Illustrating for LCR Calculation 152

Figure 67. The Workflow of the Targeting Algorithm of the MAPS Showing the Three Sub-Processes 154

Figure 68. Distance Nomenclature of Targeting Algorithm Showing the i^{th} Detected Plant at the k^{th} Captured Frame 155

Figure 69. Sample Annotated Images of Artificial Weeds and Crops in (a) Laboratory and (b) Outdoor Settings 157

Figure 70. Laboratory Setup for Velocity Calibration and Preliminary Spraying Performance Testing of the SU 159

Figure 71. The Two Field Experimental Setups 162

Figure 72. Sample Recorded Spray Instances of the Precision Sprayer 163

Figure 73. The Measured Frame Velocity of the MVS-CNN Against Different Ranges of Test Velocities 164

Figure 74. Comparison of the MAEs of Velocity Estimates Using Regression and Analytical Models 165

Figure 75. The MAEs of Velocity Estimates at Different PHF Using the Analytical Approach 166

Figure 76. The MAEs Using Analytical Approach at Different Tracking Distance Thresholds 167

Figure 77. Comparison of the MAEs from 0.04 to 0.53 m/s at Different Tracking Distance Thresholds 168

Figure 78. Sample Detections of the (a) Left, (b) Middle, and (c) Right SUs of the MAPS 169

List of Figures (Continued)

Figure Page

Figure 79. Workflow and Communication of the SU of the MAPS 178

Figure 80. The Developed MAPS with Labeled Components in the Experimental Field 179

Figure 81. Sample Images of Target Weeds: (a) Horseweed, (b) Purslane, (c) Carpet Weed, (d) Cut-Leaved Evening Primrose, (e) Hairy Fleabane, (f) Goosegrass, (g) Ragweed, (h) Lambquarter, and (i) Thistle 180

Figure 82. (Weed and Soybean Distributions in the Three Test Rows with Values Enclosed in Parenthesis Represent Soybean Plants Without Adjacent Weeds 182

Figure 83. The Precision-Recall Curve of the SSD-MB1 Model on Detecting Soybean and Weeds at 0.5 IOU Threshold 185

Figure 84. Soybean and Weed Detections of SSD-MB1 at 0.5 Confidence Threshold 185

Figure 85. Spray Rate of Weeds and Soybeans 188

Figure 86. Sample Detection Scenarios During the Experiment with Labeled Targets Weeds (Yellow) and Non-Target Soybeans (Green)........... 189

Figure 87. The Weed Population, Spray Instances Ratio, and Spray Volume Reductions 191

List of Tables

Table Page

Table 1. Reviewed Articles on Agricultural Robots 23

Table 2. Summary of Previous Research on Real-Time Precision Spraying of Weeds 28

Table 3. Past MVS-CNN Applications in Agricultural Field Operations 47

Table 4. Examples of Past MVS-CNN Applications in Real-Time Precision Spraying 49

Table 5. List of Agricultural Robots with Modular Sub-Components 55

Table 6. List of Fully-Modularized Agricultural Robots 56

Table 7. The Rated Voltages and Currents of Individual Components of the MAPS 70

Table 8. The Technical Specification of the MAPS with Three SUs 72

Table 9. Detailed Cost of Each Component of the MAPS 73

Table 10. Reference URLs of the CNN Algorithms 79

Table 11. Class Distribution of Samples in the Unbalanced Dataset 90

Table 12. Class Distribution of Samples in the Balanced Dataset 91

Table 13. Average Resource Consumption of the Tested Object Detection Models During Training 95

Table 14. One-Way ANOVA Tests of the $mAP^{0.5}$ and $mAP^{0.5:0.95}$ of Tested Object Detection Models on Unbalanced and Balanced Datasets 99

Table 15. Tukey's HSD Tests of the $mAP^{0.5}$ and $mAP^{0.5:0.95}$ of the Tested CNN Object Models Trained on an Unbalanced Dataset 100

Table 16. Tukey's HSD Tests of the $mAP^{0.5}$ and $mAP^{0.5:0.95}$ of the Tested CNN Object Models Trained on an Balanced Dataset 101

List of Tables (Continued)

Table Page

Table 17. The Paired T-Test of the $mAP^{0.5}$ and $mAP^{0.5:0.95}$ of Tested CNN Models on Unbalanced and Balanced Datasets 102

Table 18. Summary of the Component Price of the Jetson Nano Test System 108

Table 19. Summary Performance Comparison of Each Test Configuration 111

Table 20. The Complete Factorial Design for Vision Module Simulation and Theoretical Analyses 125

Table 21. Summary of Vision Module Hardware 126

Table 22. Summary of Vision Module Software 127

Table 23. Output Parameters of the Vision Module with Their Description 128

Table 24. Theoretical and Simulation Performance at $S = 0.5m$ at Different Travel Velocities and Inference Speeds 132

Table 25. Theoretical and Simulation Performance of the Developed Vision Module Different Test Velocities 140

Table 26. The Detection Performance of the CNN-Based Vision Module 141

Table 27. Parameters in the Performance Testing of the Precision Sprayer 160

Table 28. Calculated Performance Metrics of the Precision Sprayer 161

Table 29. The Detection Performance of the MAPS at Different FSF (low and high), spraying duration (200, 350, and 500 ms), and 1.03 ± 0.16 $\frac{m}{s}$ (3.71 ± 0.57 $\frac{km}{h}$) 170

Table 30. The Targeting Performance of the MAPS at Different FSF (low and high), spraying duration (200, 350, and 500 ms), and 1.03 ± 0.16 $\frac{m}{s}$ (3.71 ± 0.57 $\frac{km}{h}$) 171

Table 31. The Detection Performance of the MAPS with Three SUs at Low-Level Spray Schedule Filtering, 200 Spraying Duration, and 0.87 $\pm$ 0.14 $\frac{m}{s}$ (3.14 ± 0.49 $\frac{km}{h}$) 173

List of Tables (Continued)

Table Page

Table 32. The Targeting Performance of the MAPS with Three SUs at Low-Level Spray Schedule Filtering, 200 Spraying Duration, and 0.87 $\pm$ 0.14 $\frac{m}{s}$ (3.14 $\pm$0.49 $\frac{km}{h}$) 174

Table 33. Soybean and Weed Detection Performance at 0.5 Threshold IOU of SSD-MB1 186

Table 34. Targeting Performance of the MAPS at 0.69 $\pm$ 0.13 $\frac{m}{s}$ 187

Table 35. Spray Volume Reduction of the MAPS at 0.69 $\pm$ 0.13 $\frac{m}{s}$ and 15-Second Average Traverse Time 191

Chapter 1

Introduction

1.1 Background of the Study

Providing food for the growing global population in the presence of increasing urbanization, decreasing amount and quality of natural resources, and changing climate is the primary challenge of modern agriculture. In the next 30 years, the United Nations estimates that the global population shall rise to 9.7 billion from its current 7.9 billion [1], while studies predict a 70 to 100% increase in global food demand [2]. Concerns regarding the adverse environmental effects of conventional farming practices have increased over the past years. For instance, the over-consumption of groundwater due to extensive farming lowers water tables, affecting the water supply surrounding communities [3]. Excessive pesticide use also caused the accumulation of heavy metals beyond safe levels in soils [4]. Likewise, the over-application of fertilizer resulted in large quantities of synthetic chemical residues in groundwater [5].

Further, many farms worldwide face workforce unavailability due to declining rural populations and aging rural demographics [6]. The percentage of the total economically active population engaged in agriculture has been decreasing from 51% in 1980 to 39.08% in 2012 [7]. Traditional farming or any agricultural-related employment is unappealing to the modern youth. Most young people perceive farming as a low-skilled, low-technology, arduous, hazardous, and unsustainable employment [8, 9].

These modern challenges inspired researchers to re-evaluate current agricultural practices, develop new technologies, and recommend policies that promote sustainable farming and food security [10, 11, 12]. Meeting the growing food demand while facing environmental, economic, and social issues requires innovating from traditional agricultural expansion and intensification [13]. Studies suggest that data-driven technologies and policies are needed to make food production systems more efficient and reduce food waste during distribution [13, 14]. Likewise, new technology-based approaches to farming should be highly adaptable and transferable in under-yielding nations where most of the growth in food production is expected to come [15].

Nonetheless, developing and adopting sustainable farming technologies and practices remain slow. Farmers adopt sustainable practices when the enabling technologies are accessible, affordable, and effective. This situation was evident in the case of conservation agriculture (CA). CA implements the principles of minimum physical soil disturbance, permanent soil cover, and crop diversification to promote sustainability and increase crop yield [16, 17]. Compared to traditional land preparation, conservation tillage can significantly decrease overall labor cost [18] and fuel consumption without significantly reducing yield [19]. Further, zero-tillage, residue retention, crop rotation, and inter-cropping can improve soil organic matter and consequently increase yield [20]. However, studies showed that CA practices were only adopted when an affordable and effective technology was accessible. In Africa and South Asia, conservation tillage only became popular with the development and successful commercialization of a low-cost two-wheel tractor implement that integrated conservation tillage, seeding, and fertilizer application in a single pass [21]. Likewise, minimum or zero-tillage became widely practiced with the introduction of glyphosate-based herbicide and glyphosate-resistant crops [20, 18, 19]. However, with the decreased effectiveness of glyphosate in recent years, significant reductions in conservation tillage fields have also been observed [22].

Another modern agricultural principle that aims to increase production through modern technologies sustainably is precision agriculture (PA). PA uses robots, intelligent automation systems, and remote sensing technologies for the site- and time-specific application rates of farm inputs [23, 24]. These modern farming technologies resulted in efficient use of farm resources, increased yield, and improved environmental quality [25, 26]. For example, equipping tractors with robotic navigation systems reduced fuel consumption by 41%, 13%, and 5% during weed spraying, plowing and weed control, and fumigation, respectively [27]. Collaborative agricultural robots increased the detection rate of stressed plants during plant surveys by 71.88% compared to manual plant surveys [28]. Variable rate application of fertilizer using computer vision also increased application accuracy by up to 3.19% versus manually measured application [29]. Additionally, PA technologies can reduce labor costs, enable farm workers to safely operate in hazardous field environments, and reduce the impact of physically demanding, mundane, and arduous jobs [30, 31]. Further, robots, autonomous systems, and smart farming tools can potentially solve rural aging and population outflow challenges by increasing worker productivity and offsetting farm labor requirements [32].

Combining PA and CA can also achieve higher levels of sustainability and production than individual applications of each farming principle. For example, using wheat yield maps to identify areas with low yields enabled the execution of site-specific conservation practices, thereby increasing yield by 71% [33]. Compared to traditional land leveling, precision laser leveling in a zero-tilled rice-wheat production reduced water usage by 15.9% and increased yield by 3.6 to 10.3% [34]. Using auto-steering, CO_2 emissions in conservation tillage were reduced by approximately 10% [35]. Finally, optoelectronic sensors for detecting weeds enabled mechanical-based tillage at 90% weed control efficiency in conservation cropping systems, [36].

Nonetheless, despite the potential benefits of using PA technologies compared to traditional agricultural machines, the high acquisition cost, complex design requirements, and low durability during field operations remain the major barriers to adoption [31, 37, 38]. As a result, most commercially available PA technologies are for crop monitoring and farm information management, which require easier hardware implementation than robots for real-time field operations [39].

In real-time field operations, the varying nature of farming environments and cultural practices in growing various crops require specialized machines with complex functional requirements [40]. Integrating mechatronic systems into traditional agricultural machines commonly satisfies these functional requirements. However, with the integration of complex mechatronic systems that require a multidisciplinary approach to develop, modern agricultural machines are more expensive to produce and maintain than their traditional counterparts [41]. Thus, bigger farms are more likely to invest in PA technologies than smaller farms by leveraging the economy of scale [42, 43]. For instance, autonomous harvest machinery is fewer in specialty crops than in field crops due to the economic and technical challenges [44]. Another research also showed that the area needs to be sufficiently large to offset the high cost of robotic sprayers through chemical cost savings [45].

Nonetheless, a significant amount of farms in the world are small. About 84% of the estimated 570 million farms globally are less than 2 ha and occupy approximately 40% of the total global agricultural area [46]. Due to the lack of access to new technologies, small farms also produce less yield despite using more fertilizer [47] and pesticides [48] compared to large farms. This situation necessitates research that will lower the cost and overcome the technical barriers of PA technologies to make them a feasible alternative to traditional farm equipment for small and large farms.

In summary, sustainable farm practices, such as CA and PA, are among the potential components of the complex set of solutions to the intricate problem of food security in the changing global conditions. Nevertheless, the limited availability and accessibility to affordable and technically feasible CA and PA technologies negatively influence the wide adoption and acceptability of sustainable farming practices. **In the case of PA technologies, more work must be done to reduce cost, increase performance, and improve machine reliability.**

1.2 Motivation of the Study

Precision spraying is among the PA technologies that interest many researchers due to its potential to increase farm net revenue and reduce environmental risks through efficient liquid chemical application. These sprayers target specific plants and deliver specific rates of pesticides and fertilizers through plant detection and sprayer valve control systems. Through the years, various approaches were developed for robust plant detection and valve control systems. The following sub-sections briefly discussed these approaches to identify existing technical challenges. An exhaustive discussion of the current approaches is provided in Chapter 2.

1.2.1 Plant Detection System

Differentiating and locating target plant species in field conditions proved to be a challenging task due to the high variation in plant morphology at different growth stages [49], coupled with varying field lighting [50] and object occlusion [51]. Various methods have been explored in the past to locate target plants accurately, as summarized in the review papers of [49], [52], [53], and [54]. These proposed approaches include spectrometric, optical, distance, and image-based sensors. However, the high variation in spectral characteristics of weeds and crops at different growth stages and weather conditions makes differentiating between plant species using spectrometric and optoelectronic sensors difficult [55]. Additionally, spectral-based sensing with current approaches cannot reliably identify weeds at sufficiently low weed densities due to inadequate differences between the field spectral characteristics of weed and crop [56].

On the other hand, image-based sensing delivered promising results. However, increasing the robustness of plant detection algorithms using complex approaches, such as machine-learning-based segmentation and classification, for image-based sensing generally increases computational cost [57, 49]. For example, a precision sprayer for targeting weeds between carrot rows by color-based thresholding of normalized difference vegetation index images could operate up to 4.17 $\frac{m}{s}$, but without differentiating between weeds and crops [58]. A recent precision sprayer utilized a support vector machine (SVM) to classify shape, texture, and color feature vectors from a 4-megapixel RGB-IR camera for in-row weed targeting in carrots. Still, processing RGB-IR images using SVM limited the operation to

0.8 $\frac{m}{s}$. [59].

Recently, machine vision systems with convolutional neural networks (MVS-CNN) for plant detection have been gaining popularity due to their high detection reliability in dense agricultural environments [60]. CNN is a deep-learning approach that utilizes intricate feature extraction and inferencing layers to classify and locate objects in digital images [61]. Many studies have demonstrated the effectiveness of CNN for plant classification and localization. This situation has likely been catalyzed by the wide availability of state-of-the-art CNN architectures in public repositories in recent years [62, 52], including single-shot multibox detector (SSD) [63], MobileNets [64], You Only Look Once (YOLO) [65], and Region-based CNN (RCNN) [66].

Nonetheless, despite high accuracy, the large computational power requirement of CNN limits its application in real-time operations [61]. As a result, most MVS-CNN applications in agriculture were primarily employed in non-real-time scenarios. Presently, various studies demonstrated the robustness of CNN for non-real-time object detection in dense agricultural scenes [67, 52, 68, 69, 62, 60, 54, 70]. For example, a study that surveyed CNN-based weed detection and plant species classification reported 86-97% and 48-99% precisions, respectively [68]. Similarly, research on fruit classification and recognition using CNN showed 77-99% precision [68, 71, 72].

On the other hand, field-tested MVS-CNN precision sprayers were very few. Like other agricultural robots, developing complete systems requires a fusion of knowledge in machine learning, control systems, vision systems, mechanical design, and machine fabrication [40]. Among the occasional complete systems, much of the recent work focused on targeted weed spraying for different crops. These studies include precision weed sprayers for strawberries [73], potatoes [74], and sugar beets [75]. Other MVS-CNN demonstrated targeted application of pesticides to crops, including cabbages [76] and citrus trees [77].

Still, the slow inference speed of MVS-CNN causes weeds to be missed at high travel velocity ($\overrightarrow{v}_{travel}$). In the study of Liu et al. (2021) [73], the variable rate agrochemical sprayer with MVS-CNN for targeted weeds control in strawberry crops equipped with a 1080TI showed increasing missed detections as $\overrightarrow{v}_{travel}$ increases regardless of CNN architecture (VGG-16, GoogleNet or AlexNet). At 1, 3, and 5 $\frac{km}{h}$, their system missed 5-9%, 6-10%, and 13-17% of the targeted weeds, respectively. Thus, few studies focused on increasing the *fps* of CNN. However, reducing the size of CNN for faster *fps* decreased detection performance. For instance, the study of Chechliński et al. (2019) [78] that used a CNN based on combinations U-Net, MobileNets, DenseNet, and ResNet on a Raspberry Pi 3B+ resulted in only 60.2% precision at 10.0 *fps*. Thus, most systems rely on a fast Graphics Processing Unit (GPU) to achieve fast inference speed and prevent missed sprays. For example, the robotic sprayer of Partel et al. (2019) [79] with YOLOv3 missed 43% of the plants using an Nvidia Jetson TX2 as compared to 8% missed plants when using Nvidia 1070TI.

Nonetheless, despite using fast GPUs, the developed system had to operate below the average speed of hand-held spraying (3.4 $\frac{km}{h}$) [80, 81, 82] and boom spraying (5.76 $\frac{km}{h}$) [83, 84, 85] to be accurate. While other systems reported a high targeting performance of 70 to 96%, operating velocities were not quantified [79, 75]. In contrast, some systems only reported high velocities at 5 $\frac{km}{h}$ but did not quantify spraying accuracy [86].

A high-power GPU will have a short operation time for mobile applications and can easily fail due to limited power supply and vibrations [87]. In addition, desktop- or laptop-grade GPUs are expensive [88]. Thus, low-power and inexpensive devices such as Nvidia Jetson Nano, Nvidia Jetson TX2, and Raspberry Pi are continuously explored as substitutes for discrete GPUs. However, as previously described, due to limited computational power, low-power devices were used for non-real-time applications [89], had high missed plants for real-time spraying [79], had to sacrifice accuracy for *fps* gain [78], or operate at slow $\overrightarrow{v}_{travel}$ [76].

Recently, TensorRT-based optimization that eliminates unused paths in the network without affecting model accuracy became available [90]. For instance, in non-real-time weed detection, Olsen et al. (2019) [91] applied TensorRT to optimize and increase the inference speed of ResNet-50 from 5.5 to 18.7 *fps* in an Nvidia Jetson TX2 without affecting the 95.1% precision. Nonetheless, precision sprayers with low-power devices and TensorRT-optimized CNN models are yet to be demonstrated.

1.2.2 Valve Control System

Most MVS-CNN sprayers open the valves when the targeted plants are within a specific region of interest (ROI) in a captured frame [92] after a fixed-time delay (FTD) [73]. However, FTD-based valve triggering may result in missing the target plant or incorrectly spraying non-targets at high velocity. Thus, other precision sprayers integrate motion sensors, such as real-time kinematics and a global positioning system (RTK-GPS), to trigger the valve using a variable time delay (VTD). However, RTK-GPS accuracy is limited by the availability of nearby base stations [93, 94]. Further, high-resolution RTK-GPS receivers are expensive [95]. Given that the high cost of entry is among the main barriers to PA technology adoption [42], alternative low-cost velocity estimation methods are needed. Wheel encoders can be used for odometry-based valve triggering [59, 76] but were demonstrated be prone to wheel skidding [76].

1.2.3 Physical Configurations

In addition to being reliant on RTK-GPS, the fixed configurations of agricultural robots limit their adaptability to various farm conditions. Further, most systems can only perform specific field operations, providing farmers with an unattractive value proposition. An approach in machine design to minimize production costs and shorten the lead time of customized products is module-based design [96]. A module can be defined as a repeatable and reusable machine component that performs partial or full functions and interact with other machine components, resulting in a new machine with new overall functionalities [97].

Modular systems are also a long-time established and cost-effective strategy in agricultural mechanization to address the need for a highly adaptable and reusable machine for different farm operations. For example, tractors provide power and navigation, while the different implements offer various functionalities. However, the modular design approach is rarely implemented on agricultural robots. Almost all robotic systems in agriculture employ fixed configurations and are non-scalable [98, 99, 100, 101]. Among recently developed precision sprayers, none were reconfigurable or scalable.

1.2.4 Problem Statements

Past studies showed that MVS-CNN is a robust method for weed detection compared to non-optical-based sensing, traditional image processing techniques, and more simple machine learning algorithms. Past studies also showed that RTK-GPS was commonly used for sensing the motion of an MVS-CNN precision sprayer. Further, existing precision sprayers lack adaptability to varying crop layouts. In summary, similar to identified limitations of PA technologies, MVS-CNN precision sprayers suffer from high cost, slow velocities, not interchangeable components, and low adaptability in unstructured field environments. These specific problems of current MVS-CNN precision sprayers are outlined as follows:

1. **Problem 1:** MVS-CNN precision sprayers utilize expensive GPUs with high power requirements and can easily fail in agricultural field environments.

2. **Problem 2:** Existing MVS-CNN precision sprayers have fixed configurations, which limits their reconfigurability and scalability for varying farm layouts and cropping systems.

3. **Problem 3:** Despite utilizing fast GPUs, existing MVS-CNN precision sprayers operate below average hand-held or boom spraying velocities, resulting in low field capacities.

4. **Problem 4:** Using RTK-GPS to estimate the motion of precision sprayers is expensive and requires developed infrastructure for RTK reference base stations.

1.3 Significance of the Study

This study proposes that **a modular design approach can increase the feasibility of MVS-CNN for real-time precision spraying**. First, we demonstrated that a modular design approach could potentially reduce the component cost and power requirement of MVS-CNN precision sprayers by distributing the high computation load of CNN among parallel low-cost and energy-efficient devices. Further, in case of failure, each device can be easily replaced. Second, each vision module was utilized for localized vision-based velocity estimation. Compared to RTK-GPS, localized vision-based velocity estimation can provide a faster update of velocity readings and reduce cost by eliminating the need for auxiliary motion sensors. Third, modular architecture represented the hardware and software components as virtual nodes, simplifying the development process. A dedicated node used detection and velocity estimates from the MVS-CNN to queue multiple VTDs and control solenoid valves. The valves were positioned at an offset distance from the RGB cameras, which enabled fast field operating velocities compared to previous MVS-CNN precision sprayers. Finally, a modular agricultural robot is highly adaptable to different farm conditions and operations through reconfigurability, scalability, and reusability. Despite the advantages of modular designs, there is a lack of research on modular precision sprayers with MVS-CNN.

1.3.1 Hypothesis 1: Distributed and Parallel Processing with Cluster of Low-Power Devices

Using multiple low-power and low-cost computers for MVS-CNN precision sprayers can provide sufficient computation performance as a single powerful computer while en-

abling reconfigurability, decreasing the cost, and lowering energy consumption

The high computational requirement of MVS-CNN, despite having high detection performance and robustness in a dense agricultural scene, hinders its wide adoption for real-time precision spraying [55]. However, this study claims that low-power and low-cost devices can be used for MVS-CNN precision spraying despite their low computation capability. Past studies that utilized Jet with GPU-accelerated machine learning either used the low-power device to process high-resolution stitched images from multiple cameras [79] or did not implement TensorRT optimization on the CNN model [88, 102, 76, 103]. This research is the first to implement a dedicated CUDA-capable low-power device in each camera and utilize a TensorRT-optimized CNN model for inferencing. Further, this research is also the first to utilize modular hardware and software architecture to integrate multiple low-power devices into a single CNN-based precision sprayer.

Past studies also did not model the process of vision-based plant detection as affected by fps, $\overrightarrow{v}_{travel}$, and camera field of view S. If the precision sprayer travels too fast relative to the inference speed of the CNN, gaps can occur between consecutive frames causing certain plants to be missed. This research aims to provide a model that describes the minimum fps for a given $\overrightarrow{v}_{travel}$ and S for real-time plant detection during field operations. Similarly, the model can also be used to determine the maximum $\overrightarrow{v}_{travel}$ for a given fps and S. This way, designs can be optimized toward the applicability of low-cost and energy-efficient devices as computational hardware for MVS-CNN.

1.3.2 Hypothesis 2: Vision-Based Velocity Estimation and VTD Valve Control for In-Motion Spray

Accurate spot spraying at fast travel velocities can be achieved using individual vision and sprayer modules for vision-based velocity estimation and for queuing and

dynamic filtering of VTDs.

The long processing time between image capture and valve actuation may cause late valve opening at fast travel velocities. Aside from using fast GPUs, existing MVS-CNN offset the nozzle location away from the camera and utilized RTK-GPS or wheel encoders. MVS-CNN determines the target plant location, and motion sensors (RTK-GPS or wheel encoders) estimate the travel velocity or the relative position of the target plant to the nozzle [59, 102, 76, 103]. The target plant locations and velocity estimates were then utilized to schedule valve actuation using VTDs or odometry (nozzle-to-plant relative distance). This study suggests that MVS-CNN can estimate accurate velocity using plant detection bounding boxes as a reference. Adding velocity estimation capability to the MVS-CNN allows multiple VTDs to be calculated without relying on auxiliary systems for motion estimation. Consequently, the proposed method can simplify the overall design and reduce the total system cost of vision-based precision sprayers.

This study is also the first to utilize vision-based velocity estimation for precision spraying. Traditionally, vision-based velocity estimation was mostly applied in transportation, as summarized by [104] in their comprehensive review of the different vision-based velocity estimation methods. [105] estimated the distance traveled using a tractor-drawbar-mounted camera, and the travel distance was calculated from consecutive images using a k-nearest neighbor. Their vision-based system achieved lower errors ($\approx$ 3 mm) than wheel-encoder-based measurements ($\approx$ 7 mm) on soil tests. Nonetheless, vision-based velocity estimation for precision spraying remains unexplored.

1.4 Objectives of the Study

To demonstrate that the proposed approaches can reduce the cost and improve the performance of MVS-CNN precision sprayers, **this dissertation aims to develop a modular agrochemical precision sprayer (MAPS) with MVS-CNN.** Specifically, this main objective is divided as follows:

1. **MVS-CNN Benchmarking**. Compare the performance in different hardware of one-stage CNN object detection models for weed detection;

2. **MVS-CNN Modeling**. Develop theoretical and simulation methods in determining the effect of fps, $\vec{v}_{travel}$, and S on missed plant detections of an MVS-CNN;

3. **Modular Architecture Design**. Design the modular software and hardware architectures for a precision sprayer with MVS-CNN;

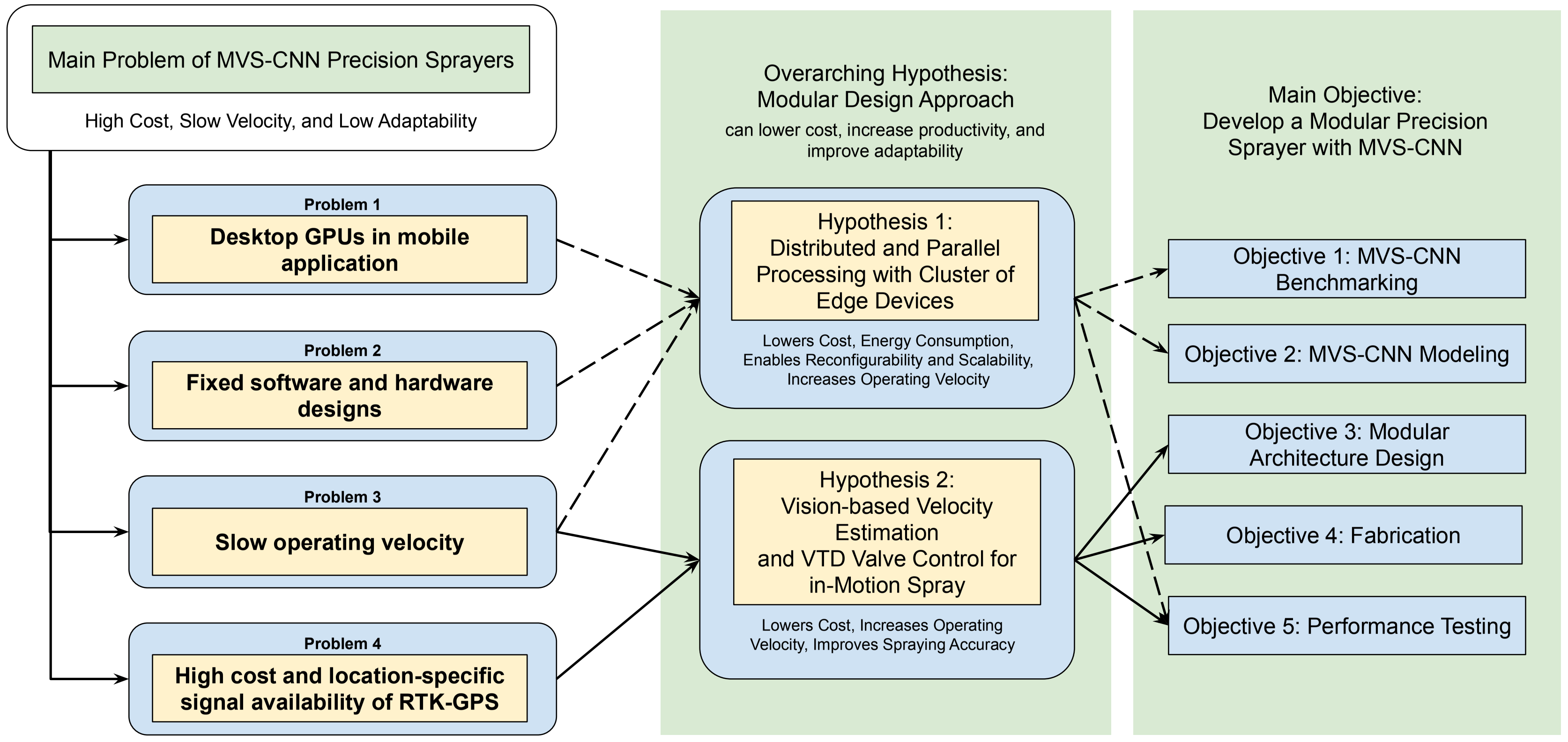
4. **Fabrication**. Fabricate scaled model and prototype of the MAPS;

5. **Performance Testing**. Evaluate the laboratory and field performance of a modular precision sprayer.

Figure 1 illustrates the relationships of the recognized problems, formulated hypotheses, and set objectives of the research. The first two and last objectives aim to prove the first hypothesis that low-cost and energy-efficient devices can substitute computers with discrete GPUs for CNN-based weed detection without compromising performance. The last three objectives demonstrate the second hypothesis that the multiple low-power devices integrated by a modular software framework can perform accurate spot spraying using the proposed velocity estimation and valve control approaches in simulated and actual field environments.

Figure 1

The Objectives of the Research with Respect to Identified Problems and Formulated Hypotheses


Main Problem of MVS-CNN Precision Sprayers
High Cost, Slow Velocity, and Low Adaptability
Problem 1
Desktop GPUs in mobile application
Problem 2
Fixed software and hardware designs
Problem 3
Slow operating velocity
Problem 4
High cost and location-specific signal availability of RTK-GPS
Overarching Hypothesis: Modular Design Approach
can lower cost, increase productivity, and improve adaptability
Hypothesis 1: Distributed and Parallel Processing with Cluster of Edge Devices
Lowers Cost, Energy Consumption, Enables Reconfigurability and Scalability, Increases Operating Velocity
Hypothesis 2: Vision-based Velocity Estimation and VTD Valve Control for in-Motion Spray
Lowers Cost, Increases Operating Velocity, Improves Spraying Accuracy
Main Objective: Develop a Modular Precision Sprayer with MVS-CNN
Objective 1: MVS-CNN Benchmarking
Objective 2: MVS-CNN Modeling
Objective 3: Modular Architecture Design
Objective 4: Fabrication
Objective 5: Performance Testing

1.5 Research Organization

This dissertation was divided into eight chapters. Chapter 2 summarizes the important concepts and past studies related to the development of the MAPS. Chapter 3 outlines the general research methods and techniques followed in the study. Specific details of the methodologies and a discussion of the results were presented in the succeeding chapters:

1. Chapter 4 presents the benchmarking of popular one-stage CNN object detection models, such as YOLO and SSD with MobileNet (SSD-MB), on different mobile platforms. Scaled-YOLOv4-CSP, YOLOv5s, SSD-MB1, and SSD-MB2 were tested for weed detection in mulched onions plots running on fast discrete computational hardware (Nvidia RTX 2080 Super) and low-power device (Nvidia Jetson Nano 2GB). Chapter 4 aimed to determine which among the tested CNN architectures had the best balance of inference speed and detection performance.

2. Chapter 5 presents the analytical modeling of the plant detection of a vision module as affected by travel velocity, inference speed, and camera configuration. The model was then verified using simulation and actual performance testing of a developed vision module.

3. Chapter 6 presents the prototyping of the MAPS utilizing the developed vision module in Chapter 5. The chapter also explains the conceptualization, implementation, performance testing, and parameter optimization of the novel vision-based travel velocity estimation method and out-of-frame targeting under controlled conditions.

4. Chapter 7 presents the performance testing of the MAPS in broadcast-seeded soybean plots to determine the system performance in actual agricultural field conditions.

Finally, Chapter 8 summarizes the conclusions, generalizations, and future directions of the research based on the results from Chapters 4 to 7.

1.6 Research Highlights

1. **SSD-MB1-TRT on Jetson Nano is a compelling MVS-CNN configuration for energy-efficient and low-cost weed detection.** Benchmarking of selected one-stage CNN object detection architectures showed a TensorRT-optimized SSD-MB1 (SSD-MB1-TRT) on a Jetson Nano was the most compelling configuration for the MAPS due to high detection accuracy and fast inference speed in low-power hardware. Further, SSD-MB1-TRT and Jetson Nano combinations provided better cost-effectiveness than an expensive laptop with RTX 2080 Super. This observed performance of SSD-MB1-TRT on a Jetson Nano directly addresses the current concerns of MVS-CNN precision sprayers, potentially solving the high cost of PA technologies and low productivity of MVS-CNN precision sprayers.

2. **Based on the verified model, the developed CNN-based vision module has sufficient inference speed and field of view for real-time precision spraying at standard operating velocity.** The results of the simulation aided in understanding the effects of combinations of $\vec{v}_{travel}$, fps, and S on missed plant detections of an MVS-CNN. A dimensionless parameter, called overlapping rate (r_o), was derived as a theoretical predictor of gaps between processed frames as a function of the mentioned parameters. r_o is the ratio of the product of S and fps to $\vec{v}_{travel}$. The results of simulation and using existing values of $\vec{v}_{travel}$, fps, and S from existing systems showed that missed detection due to gaps occurs when $r_o < 1$. Using this concept, a reference chart (Figure 60) in determining the minimum fps for specific $\vec{v}_{travel}$ and S was developed. Plotting the actual fps and S of the developed MVS-CNN showed that SSD-MB1-TRT and Jetson Nano combinations have sufficient performance in preventing gaps between processed frames. For spraying operations at 1.6 $\frac{m}{s}$ (5.76 $\frac{km}{h}$), the developed MVS-CNN had sufficient fps and field of view to evaluate a scene approximately 7.9 times.

3. **The designed and fabricated modular CNN-based precision sprayer sprayed simulated plant rows at high accuracy.** A precision sprayer (MAPS) prototype with the previously developed MVS-CNN for detection and velocity estimation was successfully designed, fabricated, and tested. The application of Euclidean-based tracking, buffer regions in the captured frame, and derived analytical method to calculate LCR resulted in a reliable way of velocity estimation at mean absolute errors of 0.036 $\frac{m}{s}$ (0.13 $\frac{km}{h}$). Performance testing showed that the proposed velocity estimation and targeting algorithms, based on queuing and dynamic filtering of VTDs, had high accuracy within the identified and designed operating conditions. Missed detections and sprays were absent at the optimum performance, which only resulted in 2% wrong sprays.

4. **Field testing on broadcast-seeded soybean showed accurate spraying of weeds at three times faster velocity than similar CNN-based precision sprayers.** Finally, the developed modular precision sprayer was tested in broadcast-seeded soybean to obtain actual spraying performance. The SSD-MB1-TRT model achieved 76% $mAP^{0.5}$ at 19 fps for weed and soybean detection in a broadcast-seeded field. Further, the sprayer targeted all weed samples at up to 48.89% spray volume reduction with a typical walking speed up to 3.0 $\frac{km}{h}$, three times faster than similar systems with known targeting performance. With these results, the study demonstrated that CNN-based precision spraying in a complex broadcast-seeded field could achieve increased velocity at high accuracy without needing powerful and expensive computational hardware using modular designs.

1.7 Research Contribution

1. A summary of the current weed detection and valve control techniques for precision spraying

2. Annotated image datasets of mulched onion and broadcast-seeded soybeans for train-

ing CNN models for weed detection

3. A comparison of the different performance and cost-effectiveness of several one-stage CNN object detection models (SSD-MB1, SSD-MB2, Scaled-YOLOv4-CSP, and YOLOv5s) on a laptop with high-power GPU (RTX 2080 Super) and a low-power single embedded board (NVIDIA Jetson Nano 2GB) in detecting weeds in mulched onion fields

4. The introduction of the dimensionless parameter r_o and development of analytical and numerical models to predict the presence of gaps between two consecutive frames of an MVS-CNN vision module for plant detection as a function of $\overrightarrow{v}_{travel}$, fps, and S

5. The development of a low-cost, low-power, reusable, and scalable CNN-based vision module for plant detection and velocity estimation based on ROS and Jetson Nano platform

6. The development of a targeting algorithm based on queuing and dynamic filtering of VTDs for accurate spot spraying at up to 1.19 $\frac{m}{s}$ (4.28 $\frac{km}{h}$)

7. The development of a modular hardware and software architecture for distributed and parallel processing of weed detection and valve control of a precision sprayer

8. Field evaluation of the spraying accuracy and spray volume reduction of a precision sprayer under a broadcast-seeded condition

1.8 Published Works

1. P. R. Sanchez, H. Zhang, S.-S. Ho, and E. D. Padua, "Comparison of one-stage object detection models for weed detection in mulched onions," 2021 IEEE International Conference on Imaging Systems and Techniques (IST), pp. 1–6, Aug. 2021.

2. P. R. Sanchez and H. Zhang, “Simulation-aided development of a cnn-based vision module for plant detection: Effect of travel velocity, inferencing speed, and camera configurations,” Applied Sciences, vol. 12, p. 1260, 3 Jan. 2022, ISSN: 2076-3417.

3. P. R. Sanchez and H. Zhang, “Evaluation of a CNN-based modular precision sprayer in broadcast-seeded field,” Sensors, vol. 22, p. 9723, 24 Dec. 2022, ISSN: 1424-8220.

4. P. R. Sanchez and H. Zhang, “Precision spraying using variable time delays and vision-based velocity estimation,” Smart Agricultural Technology, p. 100253, May 2023, ISSN: 27723755

Chapter 2

Literature Review

2.1 Introduction

An agricultural robot operating in an unstructured agricultural environment is a multifaceted machine. Its development inevitably requires integrating concepts in agricultural mechanization, mechatronics, modular robotic systems, and machine learning. Thus, the review was divided into sections providing an overview of these multidisciplinary concepts.

The first section of this chapter presents an overview of field agricultural machinery. The second section briefly examines general robot concepts and classification in relation to robots for agricultural field operations. The third section discusses current techniques in precision spraying. The fourth section provides an overview of deep-learning approaches in weed detection. The fifth section reviewed the module-based design and past agricultural robots implementing a modular design approach. Finally, the last section summarizes the findings of the literature review.

2.2 Field Agricultural Machinery

Agricultural mechanization is the application of any machine to accomplish an operation involved in agricultural production, aiming to reduce human effort and increase operational efficiency. Professional and standardizing organizations, countries, and the United Nations Food and Agriculture Organization classify agricultural machines based on field operations [106]. For example, the International Commission of Agricultural Engineering (CIGR) categorizes field agricultural machines into (1) two-wheel tractors for wetland or dryland farming; (2) four-wheel tractors; (3) tillage machinery, (4) seeders and planters; (5) fertilizer distributors; (6) pest control equipment; and (7) harvesters and threshers [107].

Similarly, countries and professional organizations draft and implement standards to promote the safety of users, provide a common measure and method of performance rating, and enable compatibility of machines between different manufacturers. The International Organization for Standardization (ISO), through its technical committee on tractors and machinery for agriculture and forestry (ISO/TC 23), publishes specifications and performance standards for agricultural machines. These standards include design safety requirements, specifications, and test methods for tractors and implements, including the electronic and electrical aspects [108].

The drafting and adoption of ISO standards are facilitated by collaborating standard organizations from different countries. For example, in the case of field machinery in the United States, the American National Standards Institute (ANSI) works with the American Society of Agricultural and Biological Engineers (ASABE), the Society of Automotive Engineers (SAE), and ISO in establishing standards for agricultural machinery.

2.3 Agricultural Field Robots

A robot is an actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment to perform intended tasks [109]. Unlike field agricultural machines, which are generally classified based on operations, robots can be classified in numerous ways. For example, based on motion control, a robot can be fully autonomous, semi-autonomous, teleoperated, remote-controlled, or automated [110].

In addition, robots can also be classified by (1) type of base or mount or positioning; (2) environment at which robots operate; (3) mechanism of interaction with the environment; (4) application field; and (5) level of autonomy [111, 112]. Since the development of robots started from industrial applications, ISO classifies robots as either industrial or service robots [109]. All robots not used for industrial applications fall under the service robot category of ISO. Based on this classification scheme, agricultural robots are considered service robots.

Like agricultural machinery classification, the Institute of Electrical and Electronic

Engineers (IEEE) also classifies robots based on the application field [113]. These categories are (1) aerospace, (2) consumer, (3) disaster response, (4) drones, (5) education, (6) entertainment, (7) exoskeletons, (8) humanoids, (9) industrial, (10) medical, (11) military and security, (12) research, (13) self-driving cars, (14) telepresence, and (15) underwater.

Various research reviewed the current status of agricultural robot development for field operations. These studies were summarized in Table 1. Many recent developments focused on mechanical or chemical weed control and harvesting. Commonly identified limitations of agricultural robots for field operations from the review articles are (1) low productivity and robustness in unstructured environments, (2) high cost, and (3) lack of a standard framework to integrate subsystems. Xie et al. (2022) [114] stated that developed systems lacked scalability and versatility. These findings also agree with the identified problems presented in Chapter 1.

Table 1

Reviewed Articles on Agricultural Robots

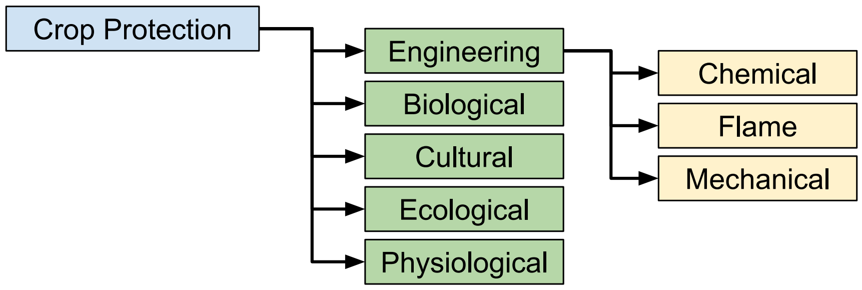
Operations	Identified Problems	Reference
Weed control, seeding, disease and insect detection, crop monitoring, spraying, fruit picking	Low productivity and detection accuracy	[115]
Weed control, spraying, field scouting	Lack of database, low accuracy, and productivity	[116]
Navigation, transplanting and seeding, pruning and thinning, weed control and disease monitoring, harvesting	Lack of standard framework for subsystem integration, low performance in unstructured environments, low productivity and high-cost of developed systems	[38]
General (Navigation and software infrastructure)	Costly connectivity and software infrastructure	[100]
General (Sensor and actuator technologies)	Need to increase scalability, versatility, operate in an unstructured environment, a standard framework to integrate sub-systems	[114]

2.4 Precision Sprayers

Precision sprayers are robotic agricultural equipment that applies site-specific rates of liquid chemicals, including pesticides and fertilizers. Sprayers are widely used for crop protection. Crop protection methods and technologies aim to provide an environment free from weeds, pests, and diseases to crops [106]. These methods and technologies can be implemented using biological, cultural, ecological, physiological, and engineering control Figure 2.

Figure 2

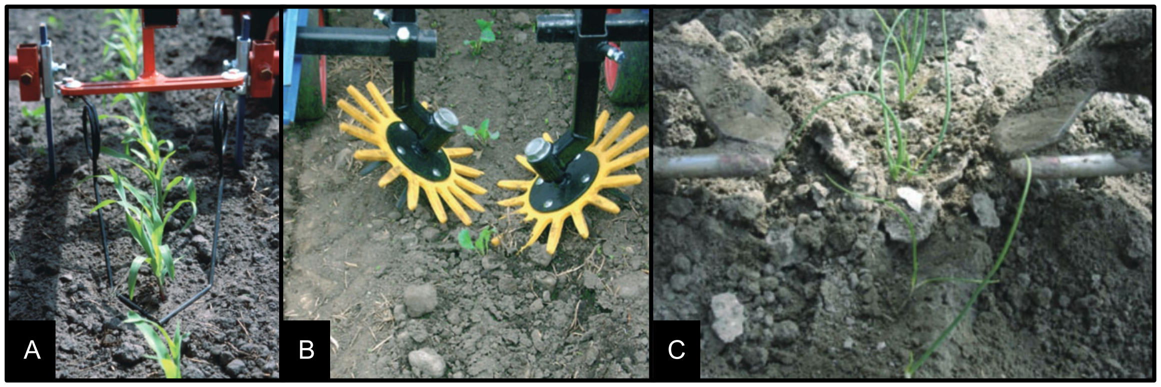
Types of Controls in Crop Protection



Engineering control can be divided into mechanical, chemical, and flame control. Mechanical weeders are hand tools or tractor-drawn implements that eliminate weeds from agricultural land by uprooting or cutting [106]. They are highly effective in controlling weeds between plant rows, as shown in Figure 3, but can damage closely spaced crops for intra-row weed control [117]. Due to this limitation of mechanical weeders, chemical weed control remain popular among farmers.

Figure 3

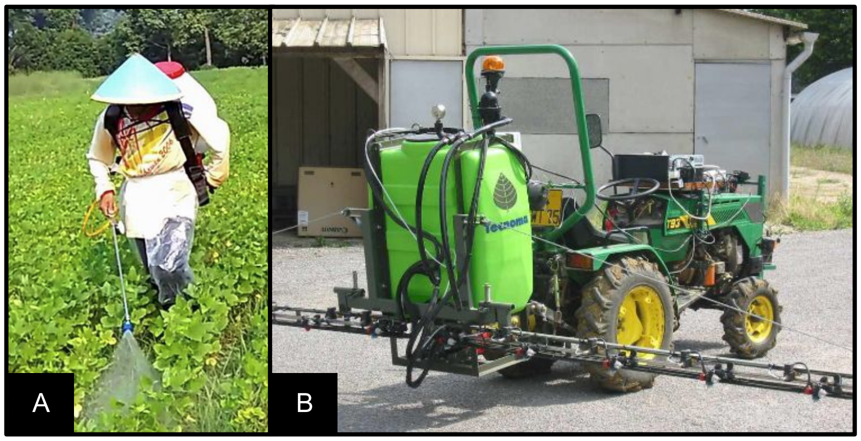
Common Weed Control Mechanisms (Adapted from [118]): (a) Torsion, (b) Finger, and (c) Pneumatic Blower



A chemical sprayer is a machine for applying liquid plant protection products and fertilizer by atomization into the form of droplets [119]. Figure 4 illustrates examples of agricultural sprayers. Knapsack sprayers are commonly used for handheld spraying. Boom sprayers with nozzles at 0.5-m spacing commonly apply pesticides [117]. Nonetheless, recent concerns regarding the over-application of chemicals through uniform spraying have encouraged the development of precision sprayers.

Figure 4

Examples of Agricultural Sprayers: (a) Knapsack Sprayer [81] and (b) Tractor-Drawn Precision Boom Sprayer [120]



ISO has several standards for spraying equipment, but particular standard specifications and test methods for precision sprayers are unavailable [106]. A general method of nozzle sprayer calibration is presented in ASAE EP367.2 MAR1991 (R2017) [121]. Similarly, the only standard specification and test methods for hand-held spraying equipment are standard specifications (PAES 112:2000) [122] and testing (PAES 113:2000) [123] knapsack sprayers in the Philippine Agricultural Engineering Standard.

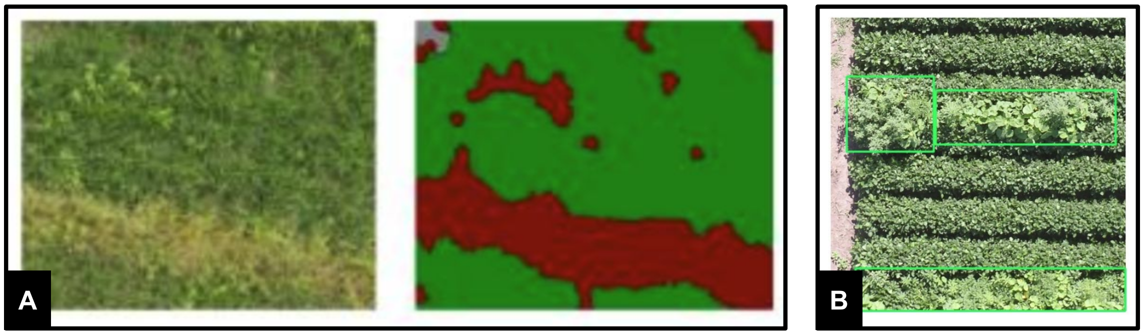
2.4.1 Non-Real-Time Precision Sprayers

During the early stages of development, precision sprayers target weeds using pre-generated weed maps. These weed maps contain global coordinates (latitude and longitude) of target weeds and were often generated using remote sensing, such as satellite images

[124, 125]. Recently, unmanned aerial vehicles (UAV) became attractive options because of the higher spatial resolution images [126]. Using UAV images, a more detailed field stratification can be employed than satellite images [127, 128]. Figure 5 illustrates examples of weed maps generated using UAV. Ground-based or UAV sprayers are then equipped with GPS with or without RTK to determine when the nozzle coincides with the specified weed location in the pre-uploaded map [129]. Nonetheless, non-real-time precision spraying suffers from the temporal differences between weed maps and field conditions during spray application.

Figure 5

Some Weed Detection Techniques Using UAV for Patch Spraying: (a) Segmentation [127], and (b) Bounding-Box [128]



2.4.2 Real-Time Precision Sprayers

Real-time precision sprayers perform on-site weed detection and spraying in a single pass. Slaughter et al. (2008) [130], Allmendinger et al. (2022) [67], and Meshram et al. (2022) [131] reviewed existing methods for precision spraying. In general, real-time precision sprayers employ two additional systems in addition to the components of conventional sprayers. These additional systems include (1) weed detection and (2) spray control systems, illustrated in Figure 6. Additionally, Table 2 summarizes the weed detection and spray control systems of precision sprayers from previous studies.

Figure 6

Common Precision Sprayer Framework

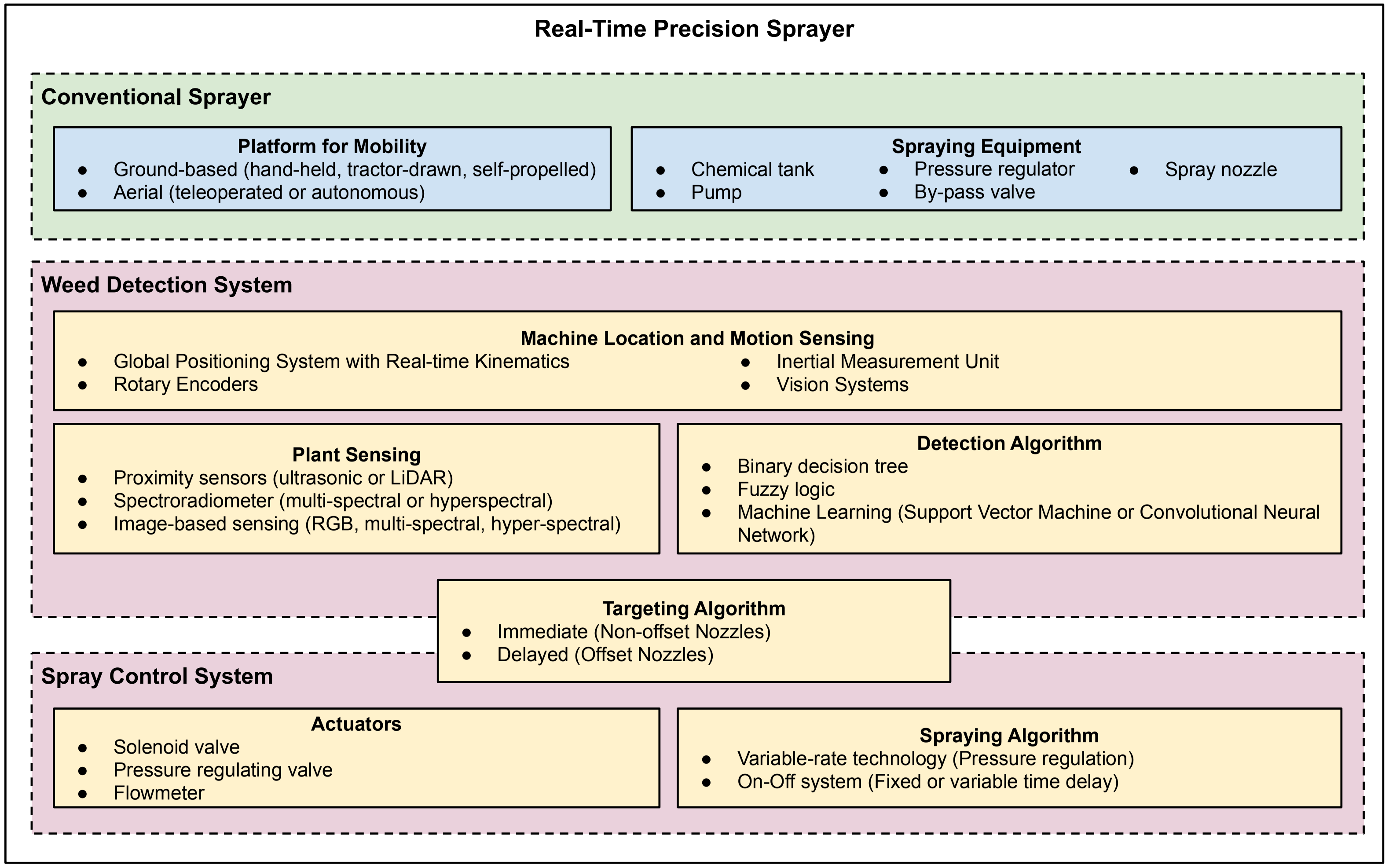

Real-Time Precision Sprayer
Conventional Sprayer
Platform for Mobility
Ground-based (hand-held, tractor-drawn, self-propelled)
Aerial (teleoperated or autonomous)
Spraying Equipment
Chemical tank
Pump
Pressure regulator
By-pass valve
Spray nozzle
Weed Detection System
Machine Location and Motion Sensing
Global Positioning System with Real-time Kinematics
Rotary Encoders
Inertial Measurement Unit
Vision Systems
Plant Sensing
Proximity sensors (ultrasonic or LiDAR)
Spectroradiometer (multi-spectral or hyperspectral)
Image-based sensing (RGB, multi-spectral, hyper-spectral)
Detection Algorithm
Binary decision tree
Fuzzy logic
Machine Learning (Support Vector Machine or Convolutional Neural Network)
Targeting Algorithm
Immediate (Non-offset Nozzles)
Delayed (Offset Nozzles)
Spray Control System
Actuators
Solenoid valve
Pressure regulating valve
Flowmeter
Spraying Algorithm
Variable-rate technology (Pressure regulation)
On-Off system (Fixed or variable time delay)

Table 2

Summary of Previous Research on Real-Time Precision Spraying of Weeds

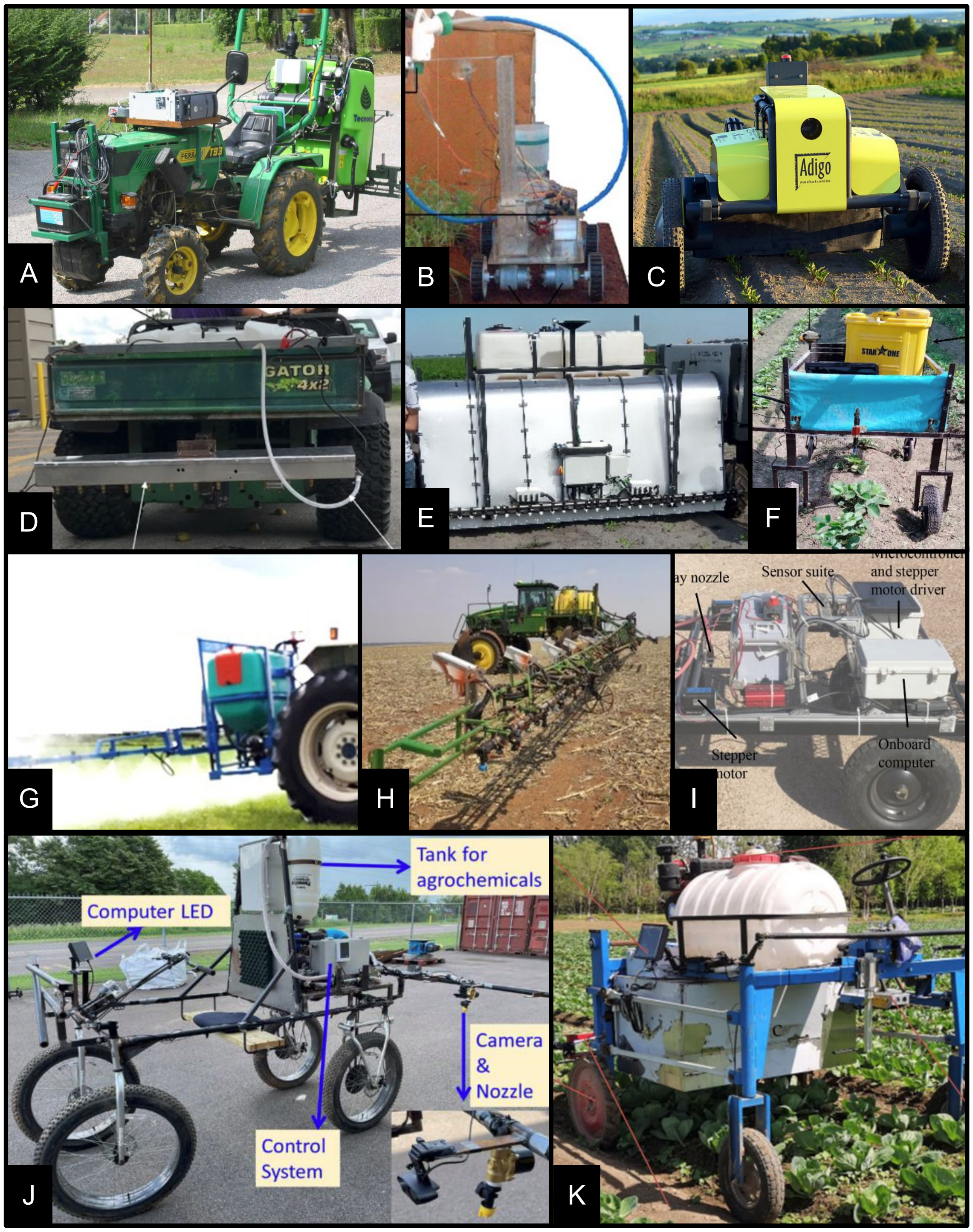
Crop	**Location and Motion Sensors**	**Plant Sensors**	**Computation Hardware**	**Detection Algorithm**	**Nozzle Position**	**Spraying Actuators**	**Spraying Algorithm**	**Spray Pattern**	**Reference**
Simulated crop rows	Speed Sensor	CCD Camera	Intel Celeron Computer	Gabor Filter using Fourier Transform	Offset (Odometry)	Electro-pneumatic valve	Variable Valve On/Off	Spot	[120]
Blueberry	Velocity from GPS	Ultrasonic	Intel i7 Computer	Distance-based Thresholding	Offset (FTD)	Solenoid valve	Fixed Valve On/Off (Variable Flowrate)	Spot	[132]
Simulated field	N/A	RGB Camera	Raspberry Pi 3B	Color-based Thresholding	Non-Offset (FTD)	Pump without valve	Fixed Valve On/Off	Spot	[133]
Carrot	GPS	Multispectral Camera	Industrial Computer	NDVI-based Thresholding	Non-Offset (FTD)	Solenoid valve	Fixed Valve On/Off (Variable Flowrate)	Patch	[58]
Carrot	Wheel encoder	RGB Camera	Nvidia Jetson TK1	SVM	Offset (Odometry)	Solenoid valve	Fixed Valve On/Off	Spot	[59]

Crop	Location and Motion Sensors	Plant Sensors	Computation Hardware	Detection Algorithm	Nozzle Position	Spraying Actuators	Spraying Algorithm	Spray Pattern	Reference
Blueberry	GPS	RGB Camera	Intel i7 Computer	Color-based Thresholding	Offset (FTD)	Solenoid valve	Fixed Valve On/Off	Spot	[134]
Simulated crop rows	RTK-GPS	RGB Camera	Nvidia GTX 2080/Jetson TX2	YOLOv3	Offset (VTD)	Solenoid valve	Fixed Valve On/Off	Spot	[79]
Sugar beets	RTK-GPS	RGB Camera	Nvidia Tesla M10 GPU	YOLOv3	Non-Offset (FTD)	Solenoid valve	Fixed Valve On/Off	Spot	[75]
Strawberry	N/A	RGB Camera	Nvidia GTX 1080	AlexNet, VGG-16, GoogleNet	Non-Offset (FTD)	Solenoid valve	Fixed Valve On/Off	Spot	[73]
Simulated field	N/A	RGB Camera	Raspberry Pi 3B	Color-based Thresholding	Non-Offset (FTD)	Solenoid valve	Fixed Valve On/Off	Spot	[135]
Soybean	N/A	RGB Camera	Nvidia GTX 1050	YOLOv3 and Faster R-CNN	Non-Offset (FTD)	Solenoid valve	Fixed Valve On/Off	Spot	[136]
Soybean and Maize	N/A	Weed-it Spectrometer	Unknown	Spectrance-based Thresholding	Non-Offset (FTD)	Solenoid valve	Fixed Valve On/Off	Patch	[137]

Crop	Location and Motion Sensors	Plant Sensors	Computation Hardware	Detection Algorithm	Nozzle Position	Spraying Actuators	Spraying Algorithm	Spray Pattern	Reference
Simulated target weeds	N/A	RGB Camera	Nvidia Jetson Nano	DeeplabV3 with MobileNet	Non-Offset (Cartesian)	Solenoid valve	Fixed Valve On/Off	Spot	[88]
Winter Pea	IMU	Stereo RGB Camera	Nvidia Jetson Xavier AGX	YOLOv5 and SORT	Offset (Odometry)	Solenoid valve	Fixed Valve On/Off	Spot	[102]
Potato	Analog speedome-ter	RGB Camera	Nvidia GTX 1050	YOLOv3	Non-Offset (FTD)	Solenoid valve	Fixed Valve On/Off	Spot	[86, 74]
Cabbage	Wheel encoder	RGB Camera	Nvidia Jetson Xavier NX	YOLOv5	Offset (Odometry)	Solenoid valve	Variable Valve On/Off	Spot	[76, 103]

Figure 7

Some Existing Vision-Based Precision Sprayers: (a) [120], (b) [133], (c) [59], (d) [79], (e) [75], (f) [73], (g) [135], (h) [137], (i) [102], and (j) [86], and (k) [76]

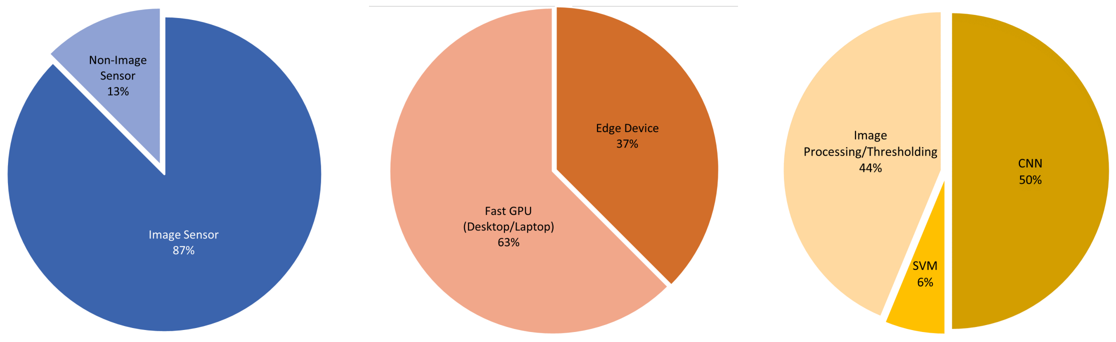


2.4.2.1 Weed Detection System. The weed detection system (WDS) determines the presence of a single target plant or patch of target plants within its effective sensing region. Past studies often used image-based (e.g. RGB and multi-spectral cameras) or non-image-based (e.g. ultrasonic, optical, or spectral) sensors [52, 49]. A detection algorithm then processes the sensed data.

Figure 8a shows that image-based sensors (87%) were commonly used for weed detection than non-image-based sensors (13%). This situation can be mainly attributed to the complex and unstructured characteristics of agricultural field environments, which makes non-image-based sensors challenging to implement. Further, Figure 8b shows that most precision sprayers utilized fast desktops or laptops (63%) to process the sensor data and minimize the time delay for real-time spraying. In contrast, low-power devices were only used by 6 systems (37%) and were mostly implemented with threshold-based detection algorithms. Finally, as illustrated in Figure 8c, machine-learning-based detection algorithms were mostly employed due to the complex nature of agricultural field environments.

Figure 8

Distribution of Technology Types for Weed Detection from 16 Reviewed Studies

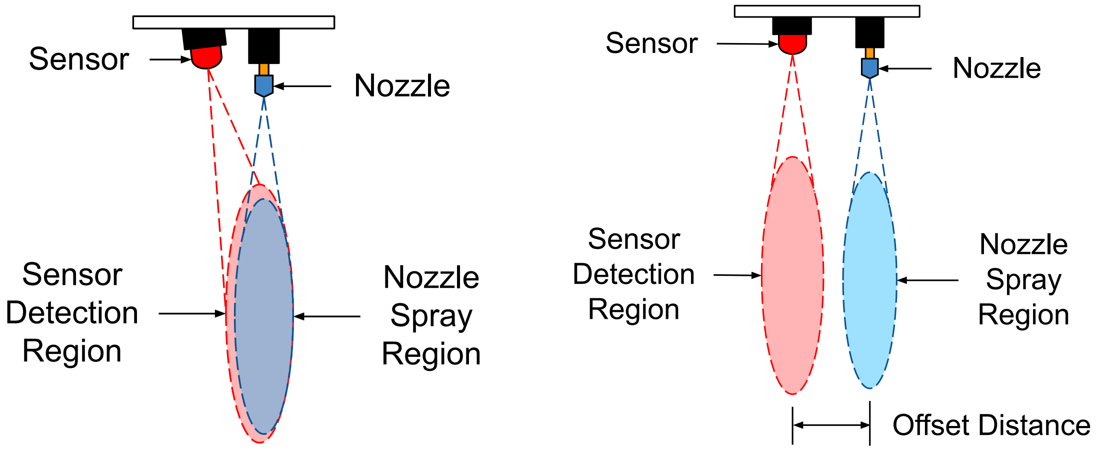


(a) *Sensors* **(b)** *Computation hardware* **(c)** *Detection algorithm*

2.4.2.1.1 Non-Image Sensors. Non-image-based sensors often utilize decision tree structures by comparing measured values of the sensor with a pre-determined threshold value [132, 137]. The main advantage of this approach is its simplicity, which only requires short processing times. Thus, the detection and nozzle spray regions often coincide (non-offset), as shown in Figure 9a.

Figure 9

Types of Nozzle Layout with Respect to the Sensors



(a) *Non-Offset* **(b)** *Offset*

Due to the short overall processing time equal to a fixed-time delay (FTD), solenoid valves can be triggered immediately when a target plant is detected. However, Table 2 showed increasing use of image-based sensing and machine-learning algorithms for weed detection in recently developed precision sprayers. Research showed that spectro-metric- and optoelectronic-based systems were challenging to implement in field scenarios due to high variability in spectral characteristics of plants at different growth stages and weather conditions [55, 56]. Further, proximity sensors could only discriminate plant species when significant height differences among target and non-target plants are present [132].

2.4.2.1.2 Image Sensors. Image-based sensing provides more information, as measured color or spectra can be associated with a spatial coordinate. However, real-time image-based detection requires complicated and computationally expensive algorithms, such as machine-learning-based algorithms, for robust plant discrimination [57]. For example, a study comparing the performance of different deep-learning techniques in detecting weed and lettuce showed that R-CNN had the highest precision but a relatively higher variable inference time than YOLOv3 and SVM [138]. Traditional image processing techniques for weed detection, such as color-based thresholding with a decision tree or fuzzy logic algorithm, proved to be only effective when discriminating between soil and vegetation [58].

Due to limited computational power, the compromise between detection accuracy and speed presents a challenge in deploying real-time weed detection models. Thus, despite the increase in computational power of portable computers and a large number of research in applying CNN for weed and crop discrimination, most real-time robotic weeders still use color-based thresholding techniques, resulting in faster but less precise classification than CNN models [139, 140, 49]. As summarized by Table 2, image-based precision sprayers with machine learning algorithms employ GPU-accelerated computation, while image-processing sprayers utilize CPU-based computation.

Similar to precision sprayers with non-image-based sensors, some image-based sprayers utilize a non-offset nozzle layout. A trigger signal is sent to the microcontroller of the solenoid valve when a weed is detected in a specified image region of interest (ROI). From Table 2, most precision sprayers that utilized image sensors and non-offset nozzle layout implemented color-based thresholding [98, 58, 135] for shorter processing times compared to machine-learning approaches. Although [86] also utilized a non-offset nozzle layout, they did not quantify their targeting accuracy. [73] similarly used a non-offset layout but had their precision sprayer operate at 1 $\frac{km}{h}$ to perform accurate spraying.

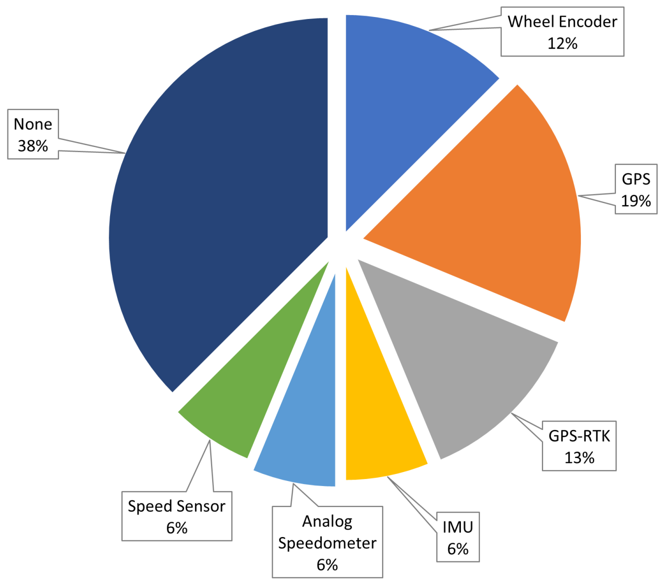
When stationary, the ROI represents the effective spray area of the nozzle [92]. However, the spray area may not completely overlap ROI when the sprayer is moving due to the long processing times of the detection algorithm. This situation worsens with increasing speed. Thus, other vision-based precision sprayers mount the nozzle at a certain offset distance from the camera (Figure 9b) to increase the allowable time delay.

The main advantage of image over non-image sensors is the availability of plant coordinates in the image frame. The plant frame coordinates can be used for the delayed valve triggering or targeting, as shown in various past systems in Table 2. Although Zaman et al. (2011) [132] described that ultrasonic could be used for delayed valve triggering, they did not report the targeting accuracy of their system. Thus, the effectiveness of proximity sensors for delayed valve triggering could not be determined.

2.4.2.1.3 Location and Motion Sensing. In delayed targeting, nozzles can be offset from the detection region and the instance when the target plant is within the nozzle spray region can be estimated [79, 103]. To estimate this instance, precision sprayers typically use GPS with or without RTK (Figure 10) for estimating the global coordinate (latitude and longitude) of target weed plants or patches [67, 49]. Nonetheless, RTK-GPS is utilized for position and velocity estimations because of their accuracy. For example, Akkamis et al. (2021) [94] compared low-cost GPS for velocity measurements at 1 to 7 Hz update frequencies and obtained 0.07 to 0.09 $\frac{m}{s}$ error at 1.55 to 4.36 $\frac{m}{s}$ constant velocities, respectively. However, the accuracy of RTK-GPS relies on the differential correction data transmitted by the reference base stations, which are not always available in rural areas. Without a reference signal, location estimates using GPS can vary from RTK-corrected measurements by 0.5-m on average [93].

Figure 10

Distribution of Sensors Technologies for Location and Motion Sensing of Precision Sprayers from 16 Reviewed Studies



Further, high-resolution RTK-GPS receivers are expensive, ranging from 1300 to 4500 USD for a single antenna and 8700 to 12,500 USD for a double antenna [95]. Low-cost GPS receivers with RTK, such as NEO-M8P, can provide centimeter-level position measurements, but this accuracy was achieved under open-sky conditions [141]. Given that the high cost of entry is among the main barriers to PA technology adoption [42], alternative low-cost velocity estimation methods are needed. Other vision-based precision sprayers employed wheel encoders to predict the instance that the target plant would coincide with the nozzle spray region [59, 76], most likely by counting the required encoder ticks before opening and closing the nozzle. However, one limitation of wheel encoders is wheel skidding, which has been shown to cause early nozzle opening in the study of [76].

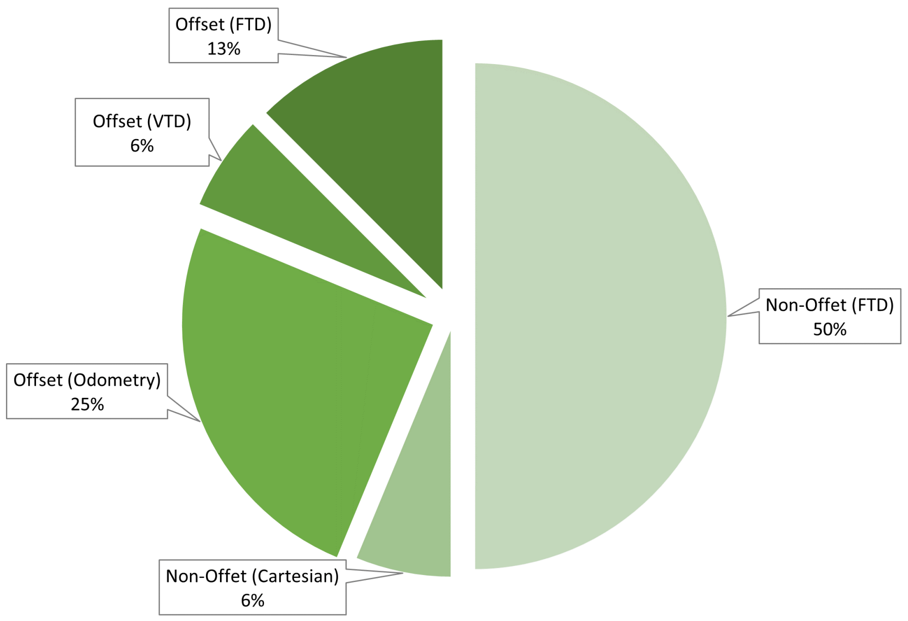
Other vision-based precision sprayers employed wheel encoders to predict the instance that the target plant would coincide with the nozzle spray region [59, 103, 76] by counting the required encoder ticks before opening and closing the nozzle. Next to GPS-based motion sensing, wheel encoders were the most common sensors used for motion estimation of precision sprayers (Figure 10). However, one limitation of wheel encoders is wheel skidding, which has been shown to cause early nozzle opening in the study of Fu et al. (2022) [76]. Also, odometry-based prediction of valve triggering does not account for the time delay caused by image processing and valve actuation, which was also shown by Fu et al. (2022) [76] to cause spraying inaccuracies.

Vision-based velocity estimation is also an alternative method. Traditionally, this approach was mostly applied in transportation, as summarized by [104] in their comprehensive review of the different vision-based velocity estimation methods. One of the earliest examples of vision-based velocity estimation was demonstrated by [142]. They implemented monocular vision and traditional image processing to estimate vehicle velocity as it moves across the camera frame, achieving less than 2% errors. In another study, [143] estimated the velocity of a toy train as it moves toward the camera using two convex mirrors to simulate stereoscopic vision. Object detection was performed through background subtraction, scale-invariant feature transform (SIFT), and speed-up robust feature (SURF). The system had 0.28% to 1.44% error at 0.09 to 0.135 second processing time. Nonetheless, precision sprayers with vision-based velocity estimation are yet to be implemented.

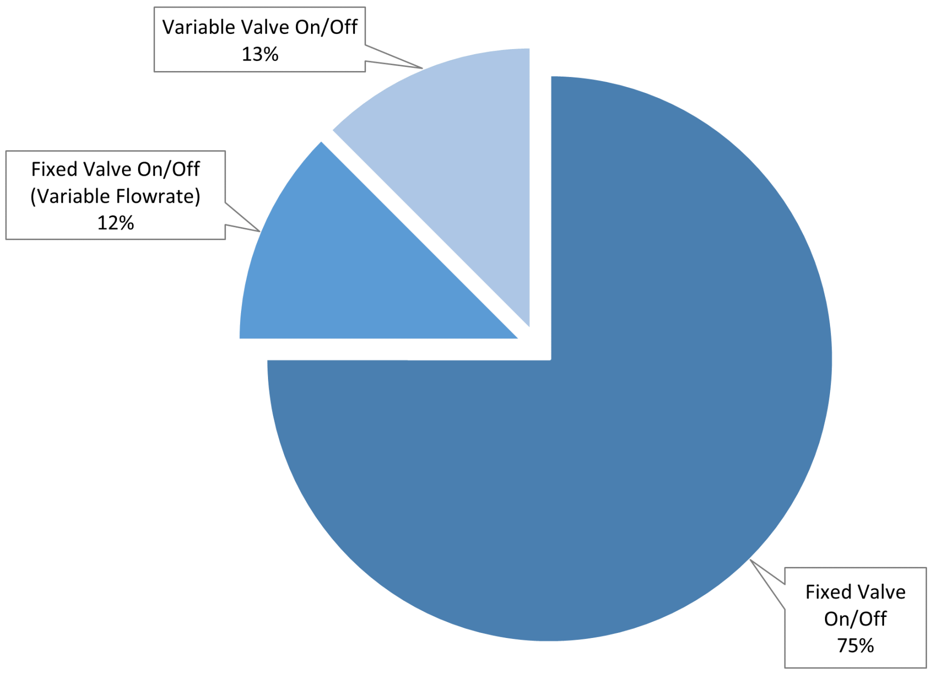
2.4.2.2 Spray Control System. Figure 11a shows that a majority of existing precision sprayers utilized FTD (63%) when triggering the solenoid valves due to the simplicity of implementation. FTD-based valve triggering was followed by odometry (25 %). Further, only Partel et al. [79] implemented valve triggering using VTD. On the other hand, Figure 11b showed that solenoid valves with fixed time On-Off intervals (87%) were the most common control system among precision sprayers. Zaman et al. [132] and Dammer et al. 2016 [58] integrated flow metering with the fixed-time opening of the nozzle to account for velocity variations. Among past studies, only Bossu et al. 2007 [120], Fu et al. 2022 [76], and Zheng et al. 2023 [103] used variable spraying duration through odometry.

Figure 11

Distribution of Targeting and Spraying Algorithms for Weed Control from 16 Reviewed Studies



(a) *Targeting Algorithms*



(b) *Spraying Algorithms*

From these results, much of the research gaps in precision spraying are within valve control. This situation may be due to the wide technical scope of precision sprayer design, which currently focuses on weed detection systems. Thus, using a non-offset nozzle layout and FTD valve triggering became the most common method for valve control.

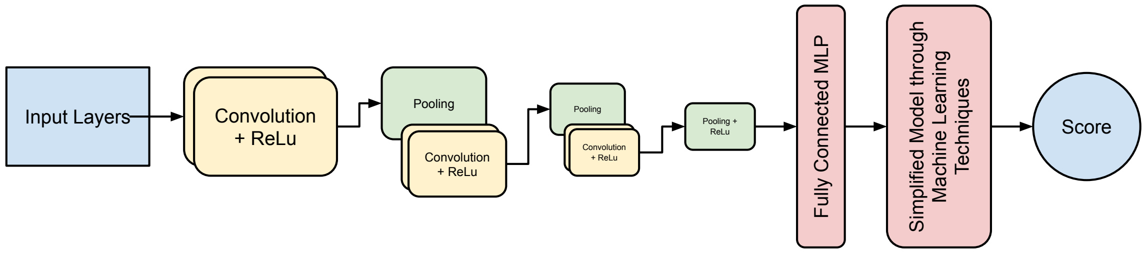
2.5 MVS-CNN

CNN is a particular DL technique defined as a deep, feed-forward artificial neural network (ANN) with deeper layers into the network and various convolutions [68]. CNN has better feature representation compared to other machine learning techniques, such as Support Vector Machines (SVM) and Random Forests, due to its capability to extract features from raw data at hundreds of feature layers linked by millions of learnable parameters to map the input predictors to the output class labels [144].

A simple CNN object detection architecture is composed of three types of layers: (1) convolutional, (2) pooling, and (3) fully connected layers [145]. Figure 12 illustrates a typical architecture of CNN for classification. The function of the convolutional layer is to perform dot-matrix operations between the inputs and weights, resulting in a feature score. For example, in the case of RGB images, the input image is a three-layered matrix or tensor with each dimension representing the color channel, x-dimension, and y-dimension. The tensor is sampled using a 3D filter or kernel that applies weights on each element of the tensor [145]. The initial weight values are usually randomly generated. However, pre-trained models for specific applications have recently become widely available. Transfer learning is the process of using CNN models with initial weight values as starting points for training. Transfer learning is often used to greatly reduce the training times during the development of CNN models for other applications during training [61].

Figure 12

Convolutional Neural Network for Object Classification



The feature map is generated by sliding each kernel from left to right and then top to bottom, usually with a stride size of one, resulting in a matrix with the same dimension as the input image or layer. The process is repeated per feature, resulting in a three-dimensional feature map per convolution. Commonly, a series of convolutions are performed, using the result of each convolution as input to the next convolution layer. The number of convolutions depends on the architectural design of the detection model. At the end of each set of convolutions is a ReLu activation function.

At the end of each set of convolutional layers is a pooling layer. The pooling layer aims to reduce the size of the feature map. Usually, a 2x2 pooling kernel with a stride the same size as the pooling kernel and max-pooling is used, resulting in a feature map with reduced width and height and the same depth (the features are retained). Then, depending on the architecture, sets of convolution and pooling are performed. These operations are performed until the feature space is reduced to a small width and height but with considerable depth.

The final layer is a fully connected multi-layer perceptron and a ReLu activation function to reduce the final three-dimensional feature map into a feature vector. Finally, machine learning techniques such as Softmax regression or SVM are performed on the feature vector to generate the object classification [71].

On the other hand, a modern object detector comprises a backbone, optional neck, and head [146]. The backbone generates feature maps through convolutions and pooling [147]. Modern detectors with a neck also feed the feature map before pooling it into a fully connected network to generate feature models. This process minimizes the loss of features during pooling. On the other hand, the head calculates a final prediction box out of the hundreds of prediction boxes generated based on each feature model from the neck. The complicated structure of CNN requires sizeable computational power. Hence, most accurate modern CNNs do not operate in real-time and need several Graphics Processing Units (GPUs) for training [146].

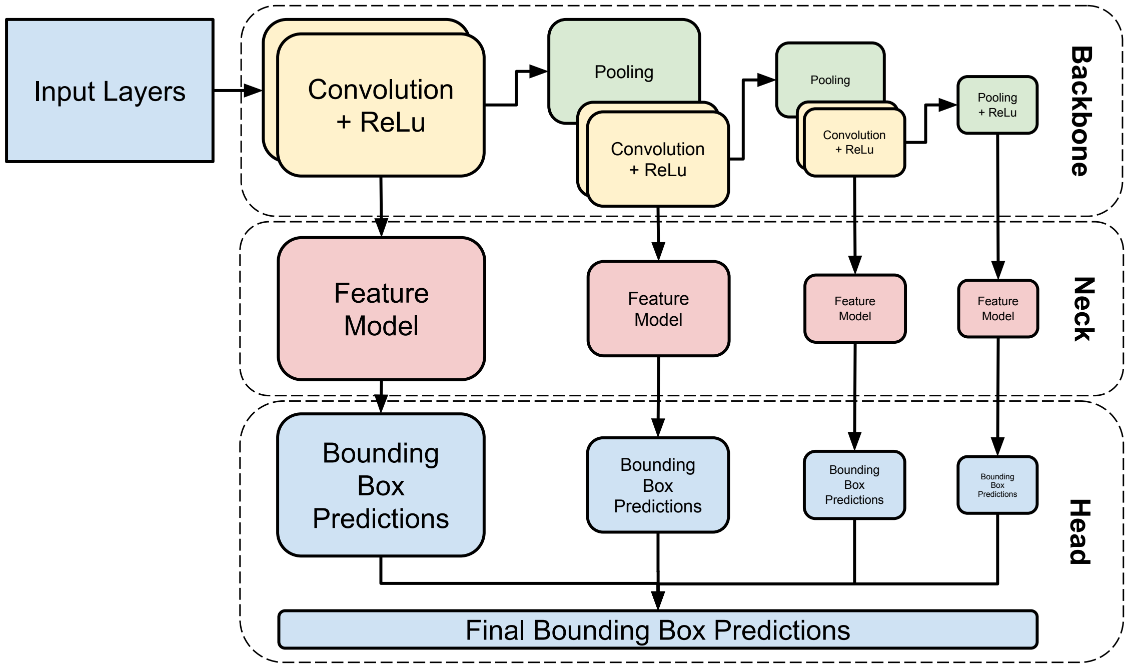
A study that compared the performance of different deep-learning techniques in weed and lettuce detection showed that R-CNN had the highest precision but a relatively higher variable inference time than YOLOv3 and SVM. [138]. A paper in detecting patches of weeds using aerial images using SSD and Faster R-CNN with Inception V2 as feature extractors showed relatively the same precision and inference time [128].

2.5.1 One- and Two-Stage CNN Architectures

Object detectors can be classified into one- or two-stage, as shown in Figure 13. The CNN directly examines the entire image in one-stage object detectors, such as SSD and YOLO, resulting in shorter calculation times [63, 65]. On the other hand, two-stage object detectors, such as the Region-based Convolutional Neural Network (R-CNN), initially extract fix number of regions (region proposals) from the original image in the first stage and classify each region using CNN in the second stage [66, 148].

Figure 13

General Architecture of a Modern CNN for Object Detection



Each object detector has its advantages and disadvantages [149]. One-stage sensors are commonly used in applications that require fast processing time despite limited detection performance. In contrast, two-stage object detectors are used in high-accuracy and non-real-time applications. A study using YOLOv3 for weed detection demonstrated that the availability of multiple frames in real-time weed detection enabled their system to detect 96% of the weeds despite only detecting 57% of the weeds in their dataset [75].

2.5.2 SSD MobileNet

SSD performs bounding box predictions using a default set of bounding boxes with different aspect ratios at different feature map resolutions [63]. Specifically, the prediction is implemented using three critical concepts: (1) using a convolutional feature layer and generating multi-scale feature maps to detect an object of a specific class at various sizes; (2) utilizing convolutional predictors to generate scores for the presence of object and shape offsets for the default bounding box; and (3) employing multiple default boxes and aspect ratios at each feature map to efficiently approximate the possible output box size and location for an object. Bounding boxes for objects with a probability of less than the threshold are disregarded. Finally, non-maximum suppression is implemented to the remaining bounding boxes so that only the bounding box with the maximum probability for an object of a particular class remains.

SSD was initially implemented with VGG-16 as its feature extractor [63] but later switched to MobileNet due to approximately 3% computational cost VGG-16 [61]. MobileNet, currently known as MobileNetV1 (MB1), was designed for mobile and embedded vision applications and implemented a depthwise separable filter, resulting in 9 times less computational cost than standard convolution filters [64]. In a depthwise separable filter, the standard convolution filter is replaced by depthwise and pointwise convolution. The standard convolution filter, which has $D_k \times D_k \times M$ dimensions for N channels, is replaced by M number of $D_k \times D_k \times 1$ depthwise convolution and N number of $1 \times 1 \times M$ pointwise convolution.

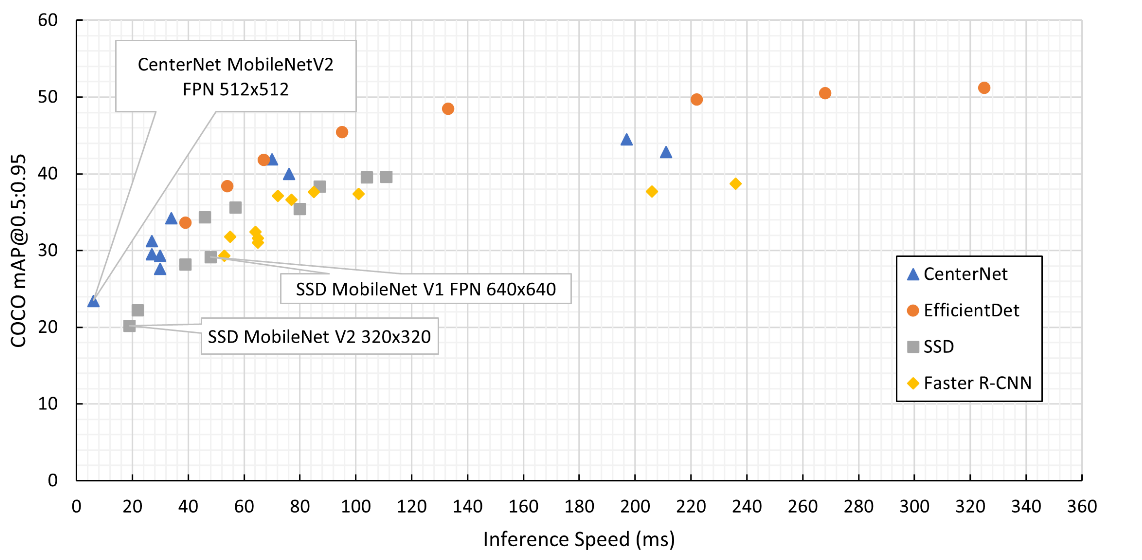
Aside from a depthwise separable filter, MB1 also uses two hyperparameters, width (α) and resolution (ρ) multipliers, to further adjust the network size and reduce the computational cost of the MB1 architecture. This network size adjustment is implemented by multiplying α to M and N and ρ to the feature map size, D_k. In the study of Huang et al. in 2017 [61], their results showed that models that used MB1 exhibited the fastest inference times. Furthermore, they demonstrated that SSD with MobileNet (SSD-MB1) had the quickest inference time among the different architecture and feature extractor combinations.

A recent modification of MB1, named MobileNetV2 (MB2), implemented pointwise before depthwise convolution, expanded the resulting layers by a factor called expansion ratio (t), and added a third pointwise-like convolution without ReLu activation function for its convolution filter [150]. The introduction of the third convolution filter and expansion ratio stemmed from a linear bottleneck layer and expansion of the output layer sizes of the depthwise and pointwise convolutions, respectively. Using a modified version of SSD, known as SSDLite, they showed that MB1 and MB2 had relatively the same *mAP*, but the MB2 was 25% faster than MB1 on SSDLite.

Figure 14 shows the *mAP* and speed of different object detection architectures when evaluated on COCO 2017 dataset. The graph shows that two-stage networks, such as Faster-RCNN, are slower than one-stage networks, such as SSD and CenterNet. The figure also indicates that CenterNet with MobileNetV2 FPN feature extractor was the fastest, followed by SSD with MobileNetV2 (SSD-MB2). SSD-MB1-FPN had a relatively higher *mAP* but was 2.5 times slower than SSD-MB2. However, the larger 640×640 image input for SSD-MB1-FPN compared to the 320×320 of SSD-MB2 likely caused the higher *mAP* but longer inference time.

Figure 14

Summary of mAP and Inference Speed of Selected Object Detection Architectures on COCO 2017 Dataset [151]



2.5.3 YOLO

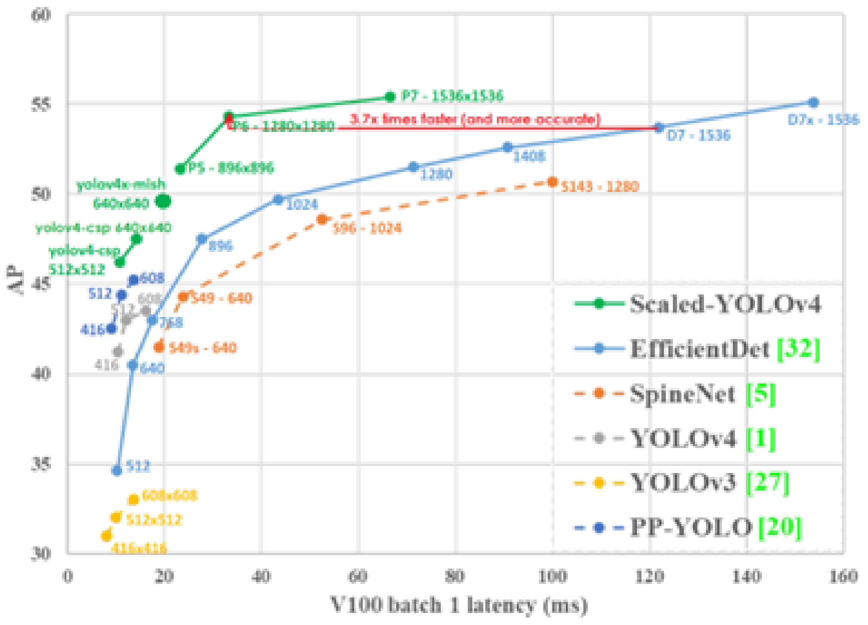
YOLO detects objects in a single forward pass by dividing the input image into $S \times S$ grid cells and employing a predicting vector to each cell [65]. Parameters for predicting object presence, sizing B amount of bounding boxes, and classifying objects compose the predicting vector. These parameters include the object center probability being inside the cell (Pr_{Object}), the bounding box center relative to the grid cell origin (x, y), and the bounding box dimensions relative to the image size (w, h). Each bounding box per cell has these five parameters resulting in $B \times 5$ elements. The remaining C number of elements of the predicting vector comprise conditional probabilities of the object class in the bounding box ($Pr_{Class_i|Object}$). Hence, the final predicting layer of YOLO is $S \times S \times (B \times 5 + C)$ tensor.

Later, YOLO9000, also known as YOLOv2, improved the detection performance of YOLO by incorporating batch normalization, high-resolution classifier, convolutional anchor boxes, dimension clusters, fine-grained features, and multi-scale [65]. YOLOv3 further improved YOLO9000 by implementing (1) logistic regression objectness score prediction for each bounding box; (2) logistic classifiers instead of softmax; (3) cross-entropy loss; (4) FPN-like algorithm for scaling; and (5) increased depth of the Darknet feature extractor from 19 to 53 convolutional layers [152]. At present, the two most recent variants of YOLO are YOLOv4 [146] and YOLOv5 [153], which are derivatives of YOLOv3.

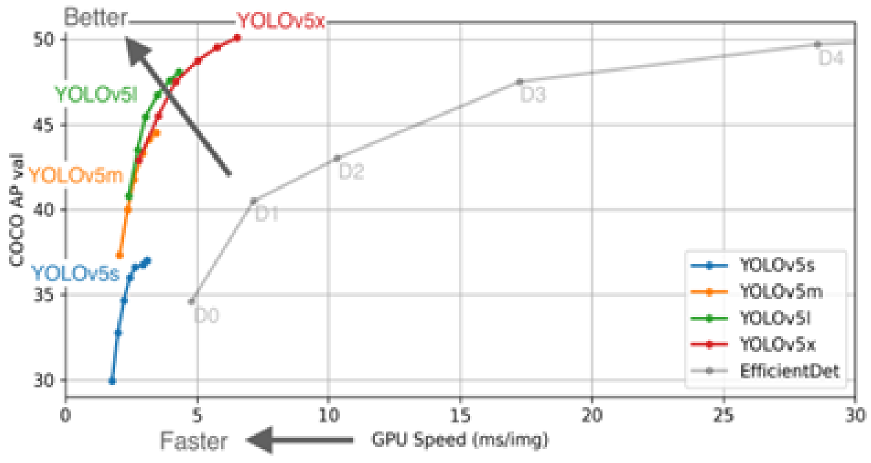
In the study of Wang et al. in 2020 [154], they incorporated Scaling Cross Stage Partial Network to YOLOv4 (Scaled-YOLOv4-CSP) and compared their model with other object detection models. Figure 15a compares the performance of YOLO with other object detectors using COCO 2017. The graph shows that YOLOv4 models have significantly higher *mAP* than EfficientDet at 512 network size. On the other hand, Figure 15b shows that most YOLOv5 models were faster and had higher *mAP* than EfficientDet. Using the performance of EfficientDet as the baseline, we infer that YOLOv5 is faster than YOLOv4, while YOLOv4 has better detection performance than YOLOv5. Among the fast YOLOv4 models, Scaled-YOLOv4-CSP has the highest accuracy. On the other hand, YOLOv5s is the fastest among the YOLOv5 models.

Figure 15

Comparison of YOLO Architectures with Other Object Detection Architectures



(a) *YOLOv4 [154]*



(b) *YOLOv5 [153]*

2.5.4 *MVS-CNN in Weed Detection*

Table 3 summarizes the performance of different CNN models for weed monitoring. A review article on using convolutional neural networks in agriculture showed that CNN models yielded 86.20-97% accuracy in identifying weeds [68]. On the other hand, studies comparing the performance of CNN models with other DL techniques showed that CNN models performed better than random forest and support vector machines [155, 138].

Table 3

Past MVS-CNN Applications in Agricultural Field Operations

Plant	CNN Architecture	Performance		Reference
		Measure	Value	
Soybean and weeds	R-CNN	F1	66%	[128]
	SSD	F1	67%	
	ConvNets	Precision	99.50%	[155]
Lettuce and weeds	Mask R-CNN	F1	94%	[138]
	YOLOV3	F1	94%	
Turfgrass and weeds	Detectnet	F1	99%	[156]
	VGGNet	F1	85-86%	
	GoogleNet	F1	75%	
Crop and weeds	VGG16	F1	81.6-85.1%	[157]
Various weed species	ResNet	Accuracy	88-97.2	[91]

2.5.4.1 Effect of Dataset Size. There is also a considerable variation in the dataset size and class distribution used for training object detection models. Research on weed and crop detection using 375 training samples of UAV multispectral images had weed and crop class distributions of 64.80% and 35.20%, respectively [157]. On the other hand, in a study on weed and carrot detection using Random Forest as a classifier, 494 training samples composed of 67.21% weeds and 32.79% carrots were used [158]. The weed detection study in turfgrass utilized 4,550 training class samples, having 42.41% with weeds and 57.59% without weeds [156]. Evaluating the cited studies showed that models trained on datasets with low sample sizes and significant class imbalance showed lower performance than models trained on more samples and balanced class sample distribution.

Class imbalance can be reduced by either downsizing the class with more samples or upsizing the one with few counts. A study comparing performances of different DL algorithms to detect soil, broadleaves, and soybeans showed that a model train using an unbalanced but larger dataset has higher overall precision than a downsized, balanced, but smaller-sized dataset [155].

2.5.4.2 Spraying Accuracy and Operating Velocity. Due to limited computational power, the compromise between detection accuracy and inference speed presents a challenge in deploying CNN models for real-time precision spraying. Thus, as shown in Table 4, despite the increase in computational power of computers and a large number of research in applying CNN for weed and crop discrimination, most real-time precision sprayers had to operate at walking velocities. Farooque et al. (2023) tested their system at 5.0 $\frac{km}{h}$ but the accuracy of their sprayer was not specified.

Table 4

Examples of Past MVS-CNN Applications in Real-Time Precision Spraying

Crop	**Planting Layout**	**Hill Spacing, m**	**Hardware**	**Travel Speed, $\frac{m}{s}$ ($\frac{km}{h}$)**	**Weed Spraying Accuracy**	**Wrong Spray Rate**	**Reference**
Strawberry	Row	1	GTX 1080 (Desktop)	0.28 (1.0)	94%	0%	[73]
Soybean	Row	Unspecified	GTX 1050 (Desktop)	0.5 (1.8)	78%	Unspecified	[136]
Potato	Row	Unspecified	GTX 1050 (Desktop)	1.39 (5.0)	Unspecified	Unspecified	[86]
Sugarbeets	Row	0.19	Tesla M10 (Server)	Unspecified	96%	3%	[75]
Simulated Field	Row	1	GTX 1070 Ti (Desktop)	Unspecified	78%	8%	[79]
Winter Pea	Row	0.2	Jetson Xavier AGX	0.4 (1.44)	Unspecified	Unspecified	[102]
Cabbage	Row	0.4	Jetson Xavier NX	0.7 (2.52)	98.91%	Unspecified	[76]

The published research also showed few studies investigating small computational board performance in real-time weed detection. Among these few studies is on the development of weed-detecting robots in sugarcane fields [159]. This robot used Raspberry Pi to implement a fuzzy real-time classifier on processed images to detect weeds at high accuracy and short processing time. On the other hand, a robotic farm implement used a Raspberry Pi 3B+ to implement a custom feature extractor based on U-Net, MobileNets, DenseNet, and ResNet showed short inference time but low detection rates ($\approx$ 60%) [78]. Finally, a study that developed a real-time sprayer using YOLOv3 on Nvidia Jetson TX2 showed about 40% missed spray due to slow inference speed [79]. The most promising configuration was demonstrated by Fu et al. 2022 ([76]) on targeted spraying of cabbage using YOLOv5 on an Nvidia Jetson Xavier NX. Their system sprayed 98.91% of cabbage at 0.7 $\frac{m}{s}$.

2.6 Modularity

Based on system reconfigurability, robots can be fixed or modular. In robotics, modularity refers to a set of characteristics that allow systems to be separated into discrete components and recombined into a larger system [160]. Other standard definitions of concepts in modular service robots can be found in ISO 22166-1:2021 [160].

The concept of modularity is not new in robotics. Currently, classification schemes based on the geometry of modular robotic systems are already being proposed [97, 161]. Various modular software frameworks for different types of robots are also existing. The goals of these frameworks are to minimize redundancy in development and provide a standard scheme for component organization and communication [162]. A widely used robotics software framework (RSF) that focuses on mobile robot navigation, depth perception, and planning is Robot Operating System or ROS [163, 164]. Likewise, Yet Another Robot Platform or YARP is a popular software framework with modular components designed for humanoid robotics [165].

2.6.1 Module-Based Hardware

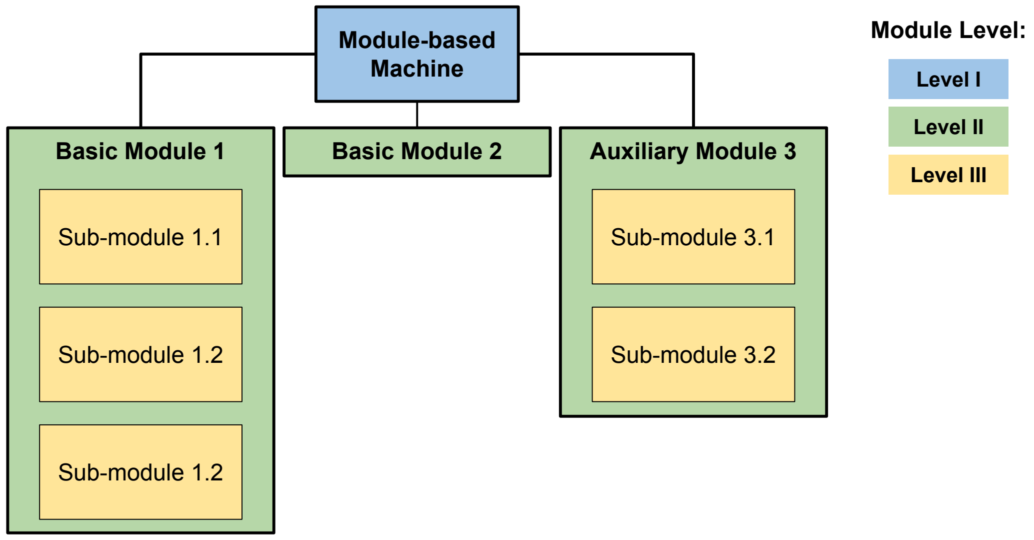
ISO defines a module as a component or assembly of parts with defined interfaces accompanied by property profiles to facilitate system design, integration, interoperability, and reuse [160]. Modular robots can be classified as chains, lattices, trusses, and variable shape systems [97, 161]. Others have proposed a three-classification system: chains or trees, lattice, and hybrid [166]. However, these classification schemes were based on experimental robotic systems with only mechanical linkages and had no specific functions.

A study by Gauss et al. [96] proposed a design framework for module-based machinery. They have divided the engineering design process into three stages: (1) planning, (2) concept development, and (3) system-level design. Planning aims to establish target specifications based on identified functional requirements (FR) of different hierarchical importance. On the other hand, concept development aims to identify the different design modules represented in a hierarchical tree. Finally, the system-level design phase selects the appropriate design modules, configuration structure, and details.

The resulting modular configuration structure is often called a system or hardware architecture [167, 168]. It is typical for hardware architecture to be represented in a hierarchical tree with each level corresponding to a degree of modularity, as shown in Figure 16. In agricultural machinery, a basic module is a mechanical or robotic unit that performs a specific agricultural field operation. Several basic modules can then be combined using a tree structure to create a modular robotic implement or self-propelled agricultural robot.

Figure 16

A Generic Three-Level Modular Hardware Architecture



ISO identified six types of modules based on function: (1) actuator; (2) communication; (3) computing module; (4) infrastructure; (5) sensing; (6) and supervisor modules [160]. Modules with or without sub-components are called compound or primitive modules, respectively [96]. Basic modules are the main components that deliver critical FRs, while auxiliary modules represent components with non-critical FRs. Modules are physically connected through a mechanical interface. A mechanical interface is a physical means of connecting with other modules to transmit physical forces and facilitate module function and configuration [160].

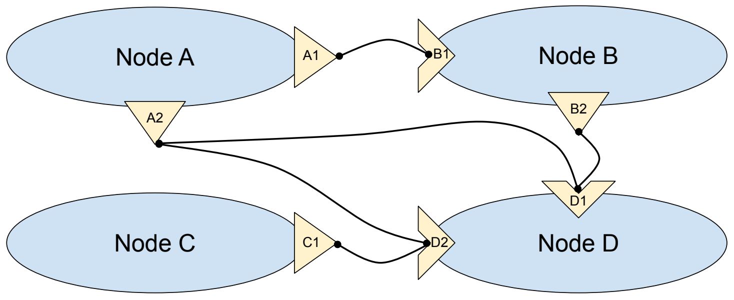
2.6.2 Robotics Software Framework

A Robotics Software Framework (RSF) is a set of software tools and libraries that provide a virtual component model, communication middleware, and protocol to manage the state and the life cycle of the components [169]. RSF is often referred to as middleware in robotics despite providing management, development, simulation, modeling tools, and hardware driver and algorithms on top of communication architecture [163, 170]. Well-known open-source RSFs targeting industrial and service robotics include Player, Orca, Orocos, ROS, YARP, Open Robotic Technology Middleware, Open Platform for Robotic Services, Open Robotic Development Kit, SmartSoft, and Robotic Construction Kit [169].

The central aspect of RSF is the implementation of distributed architecture that allows synchronous or asynchronous communication of nodes [170]. In ROS, a node is a virtual representation of a component. Synchronous communication is facilitated by sending or receiving messages from other nodes through ports or services. Each ROS node has an in-port and out-port, which allow nodes to synchronously read and write messages, respectively (Figure 17). These ports are dynamic communication buffers that can be created or destroyed. Messages can be read by the receiving port synchronously through polling or asynchronously through callback functions. On the other hand, ROS services employ an on-demand request and response model. A server node is a global procedure with specific arguments. A client node sends a request message to a server node. After completing the procedure, the server node sends a response message to the client node.

Figure 17

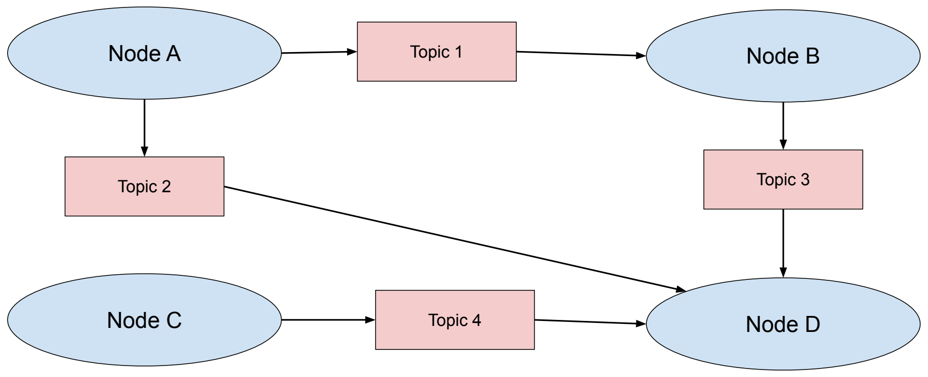
The Synchronous Communication of ROS Nodes Using Port Mechanism



On the other hand, ROS implements asynchronous communication using topic or event communication mechanisms Figure 18. A topic is a specific channel for a particular type of message. If a node is a sensor module, the node sends sensor readings as messages under the “sensor reading” topic. Other components subscribed to the “sensor reading” topic, such as a controller or actuator, can read published messages. Finally, nodes in the events mechanism directly post their messages to subscribed nodes using a communication buffer, and subscribed nodes read messages through callbacks.

Figure 18

The Asynchronous Communication of ROS Nodes Through the Topic Mechanism



2.6.3 Modular Agricultural Robots

A review of past studies shows very few agricultural robots implement a modular approach. The developed robots can be divided into two categories: (1) partial and (2) fully modular. Robots with partial modular components usually have a modularized manipulator design, as shown in Table 5. These designs implement a replaceable tooling system or vary the degrees of freedom of the robotic manipulator.

On the other hand, fully modularized designs have distinct parts that perform specific functions, as shown in Table 6. The degree of modularization varies from each study using the system presented in Figure 16. In Thorvalds II, which is the most mature among the modular agricultural robots, the system had up to level 3 of modularization, as each distinct function is a module. The rest of the studies employed up to level 2 modularization.

Table 5

List of Agricultural Robots with Modular Sub-Components

Description	Fixed Components	Modular Component	Software Framework	Interfaces
Modular and multifunctional agricultural robot system for specialty crops [171]	Power supply, control unit, precision sprayer, sensing unit	Manipulator	ROS	CAN, AD-Converters, ethernet
Teleoperated robotic system with modular end effectors for greenhouse watermelon [172]	RF data modem, wireless image processing system, transfer unit, control unit (PLC)	Manipulator	Custom software	Digital I/O, RF signal
Modular agricultural robot [173]	Platform and steering	Tools (sensors, controller, communication, GPS)	Custom software	UART, GPIO, USB
Reconfigurable agricultural robot for orchard [174]	Navigation platform	Manipulator	Custom software	Wired connection (no specific description)

Table 6

List of Fully-Modularized Agricultural Robots

Description	**Modules**	**Sub-components**	**Software Framework**	**Connection Interfaces**
Autonomous field robot for individual plant phenotyping [168]	Navigation	controller, RTK-GPS	Custom framework based on Linux Operating System	USB, RS232, Ethernet, AD-Converters
	phenotyping control system	Gigabit switch, control unit		
	Speed and steering control	Gigabit switch, control unit, motor/hydraulics, rubber wheels		
	Sensors	Gigabit switch, Ethernet I/O, PC, camera, NIR, Triangulation sensors, Light curtain		
	Communication	WLAN, Router		
Multifunctional agricultural robotic platform for upland [175]	Sensor Unit	GPS, Camera, IMU	Custom framework	UART (RF), CAN, GPIO
	Navigation Unit	Driver, motor, rubber heels		
	Working Unit	Driver, Motor, Tool		

Description	Modules	Sub-components	Software Framework	Connection Interfaces
Conceptual multifunctional agricultural robot for strawberry field [176]	Central Unit	Computational unit, motor driver, RC receiver, control unit	Custom software running on Linux Operating System	Ethernet, WiFi, USB
	Power source	24VDC Battery		
	Communication	RC transmitter, WiFi, Bluetooth		
Multi-functional modular and reconfig-urable Agricultural Robot (Thorvald II) [177]	Drive	Motor and transmission	Custom software	Ethernet, USB
	Steering	DC motor, transmission, two-channel motor controller		
	Motor controller	Weather-proof case, two-channel motor controller		
	Passive wheel	Caster wheels, rubber wheels		
	Suspension	Shock absorbers		
	Sensor interface	Sensor attachment		
	Sensor mounting	Aluminum mounting frame		
	Frame	Tube		

Description	**Modules**	**Sub-components**	**Software Framework**	**Connection Interfaces**
Modular robotic tool carrier [178]	Track module	Controller, electric motor, transmission, rubber tracks	Frobomind based on ROS	RSxxx, USB, Ethernet, CAN
	Power module	Battery (48V 100Ah), electrical generator		
	Frame	Metal bars		
Modular agricultural robotic system [179]	Wheel module	Steering, drive, passive	ROS	RS232, USB
	Suspension module	Shock absorbers		
	Power Module module	Battery (12 pcs. of 38.4V 60-Ah), batter management system		
	Controller module	Nvidia Jetson Xavier		
	Frame module	Aluminum extrusions, 3D-printed components (PLA)		
	Sensor interface	Sensor attachment		
	Actuator interface	Actuator attachment		

The standard connection interfaces used for communication were Universal Serial Bus (USB), ethernet, Wireless Fidelity (WiFi), or CAN. Most of the studies also implemented a custom software framework that is not publicly released. Some studies utilized ROS as a base software framework to develop their custom-made software framework, such as in the study of Xu et al. (2022) [179], Oberti et al. (2014) [180] and Jensen et al. (2014) [181]. However, none of the studies incorporated a CNN-based object detection model in their framework.

There is also a confusing nomenclature to name each component. For example, the navigation unit comprised the RTK-GPS for a modular plant phenotyping robot [168]. In another study, the navigation unit referred to the wheels and motors [175]. Other studies refer to wheels, transmission, and motor assembly as the drive or track units. Hence, this situation requires the need to have a standardized naming system for agricultural components.

2.7 Review Summary

This chapter reviewed recent design approaches and techniques for precision spraying. Overall, the review of recent research and developments in precision spraying showed a growing direction toward using image-based sensors and CNN models for the real-time detection of weeds. Specifically, the following findings were identified:

1. Existing CNN-based precision sprayers employed fixed configuration and utilized fast, high-energy-consuming, and expensive GPUs.

2. To increase the feasibility of CNN-based object detection, one-stage object detection models and optimization using TensorRT was shown to effectively reduce the computational cost of CNN while providing high detection accuracy in agricultural field environments. However, despite these improvements, most CNN-based precision sprayers have lower productivity than standard boom spraying operations.

3. Non-offset nozzle layout and FTD-based valve actuation offer the simplicity of implementation but require fast processing speed to prevent missed spray at high velocities. Thus, this method was mainly employed on non-image-based or image-based sensors that utilized traditional image processing for segmentation. As a result, CNN-based systems that used Non-offset nozzle layout and FTD-based valve actuation had high missed sprays, were tested at low velocities, or did not quantify targeting accuracy.

4. Offset nozzle layouts and delayed valve triggering were commonly employed for vision-based CNN sprayers but required auxiliary sensors for motion estimation. On the other hand, precision sprayers mostly use RTK-GPS for motion sensing. However, as mentioned in the previous chapter, RTK-GPS for motion sensing is not an attractive method in under-yielding countries due to the expensive cost of the technology.

5. A modular precision sprayer with MVS-CNN remains unexplored. Most past studies on modular agricultural robots developed mobile platforms without specific functions. Early studies utilized custom software frameworks, while recently developed systems use a custom software framework based on ROS.

Chapter 3

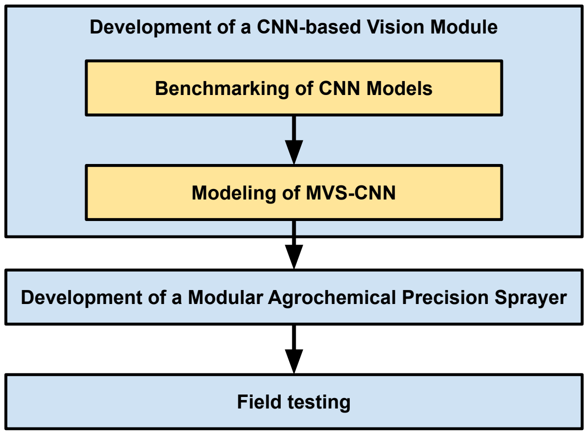
Materials and Methods

3.1 Overview

The development of the MAPS was divided into four phases: (1) performance benchmarking of one-stage CNN object detection models for weed detection on different hardware; (2) modeling of MVS-CNN for plant detection; (3) development of the MAPS; and (4) field testing. Figure 19 illustrates the three main phases of the study and sub-components. The first two steps aim to develop a CNN-based vision module that is low-cost, energy efficient, and reusable. Thus, benchmarking and modeling of MVS-CNN were implemented to gain an understanding of the factors affecting detection performance.

Figure 19

Flowchart of the Development Process of the Modular Agricultural Robotic Sprayer


Development of a CNN-based Vision Module
Benchmarking of CNN Models
Modeling of MVS-CNN
Development of a Modular Agrochemical Precision Sprayer
Field testing

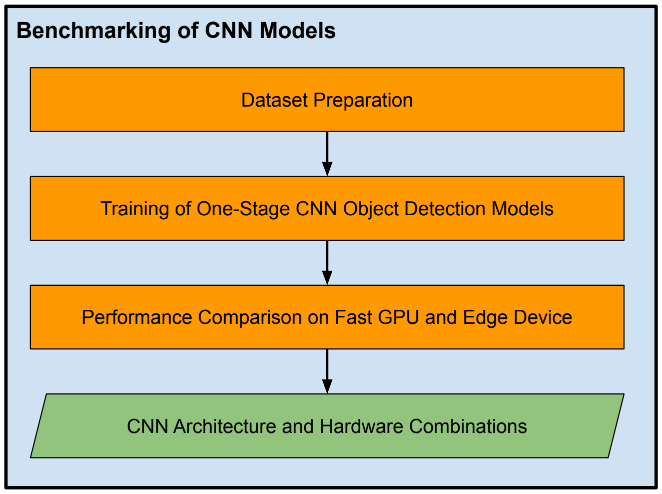
The third phase integrated a sprayer module, vision-based velocity estimation, and valve control based on queuing and dynamic filtering of variable time delays (VTD) to the previously developed vision module to form the scalable unit (SU) of the MAPS. The vision-based velocity estimation and valve control parameters of the SUs were optimized in simulated field conditions. Finally, after tuning, the MAPS was tested in the field during the fourth phase to obtain its actual performance. A separate section on CNN model development was dedicated as a recurring step in all phases of the research.

3.2 Development of CNN-Based Vision Module

The combinations of one-stage CNN object detection models and different hardware systems for weed detection were initially evaluated (Figure 20). Cost-effective CNN architecture and hardware combinations for weed detection were identified based on ease of (1) data preparation, (2) training, (3) detection performance, and (4) processing time. Scaled-YOLOv4-CSP, YOLOv5s, SSD-MB1, and SSD-MB2 were trained and tested on the collected image dataset of a mulched onion field. The loading time, inference time, and cost efficiency of the selected models on a mobile laptop with a powerful GPU and a low-power embedded device were evaluated. The results were finally compared to determine which combination was best suitable for our application.

Figure 20

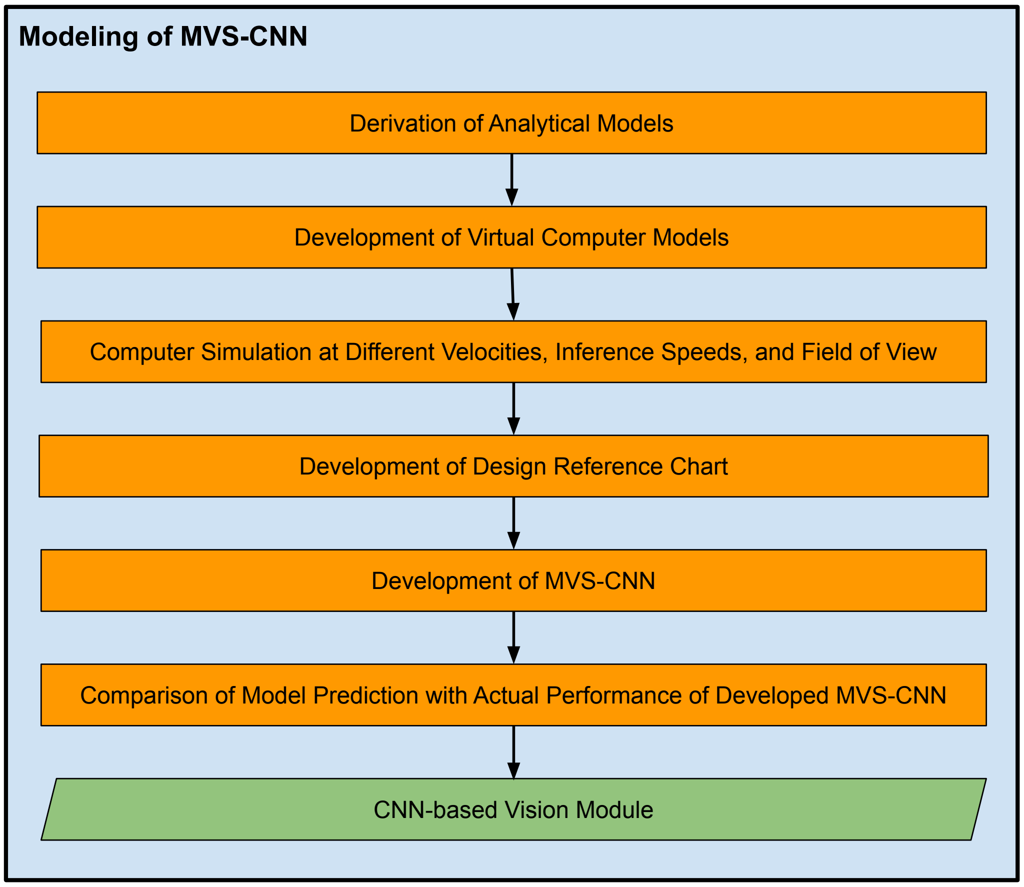
The Steps in the Benchmarking of Selected CNN Models on Different Hardware


Benchmarking of CNN Models
Dataset Preparation
Training of One-Stage CNN Object Detection Models
Performance Comparison on Fast GPU and Edge Device
CNN Architecture and Hardware Combinations

After determining the suitable combination of CNN architecture and hardware for weed detection, a CNN-based vision module was developed. Its feasibility for real-time precision spraying was evaluated based on theoretical and simulation models (Figure 21). The derived analytical models and computer simulations are discussed in detail in Chapter 4.

Figure 21

Steps in Modeling the Detection Rate of an MVS-CNN for Plant Detection

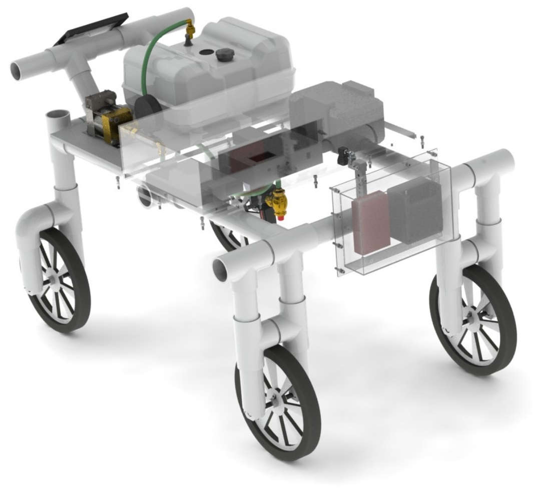

Modeling of MVS-CNN
Derivation of Analytical Models
Development of Virtual Computer Models
Computer Simulation at Different Velocities, Inference Speeds, and Field of View
Development of Design Reference Chart
Development of MVS-CNN
Comparison of Model Prediction with Actual Performance of Developed MVS-CNN
CNN-based Vision Module

3.3 Development of the MAPS

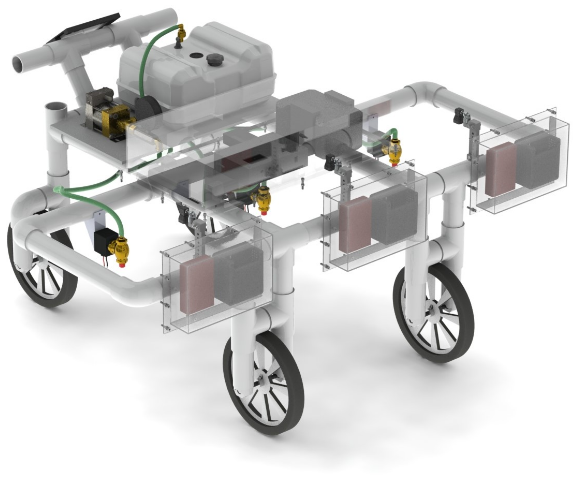
Figure 22 shows a render of the developed MAPS. Three key innovations were implemented in the design to increase $\vec{v}_{travel}$ while using low-cost and energy-efficient devices for CNN-based plant detection: (1) modular hardware and software architecture; (2) vision-based velocity estimation; and (3) valve control through queuing of variable-time delays (VTD). The detailed list of components and assembly drawings are included in Appendix C. A prototype was then developed by following the steps in Figure 23.

Figure 22

The Precision Sprayer Prototype with the Push-Type (a) Base Unit and (b) Extended Configuration



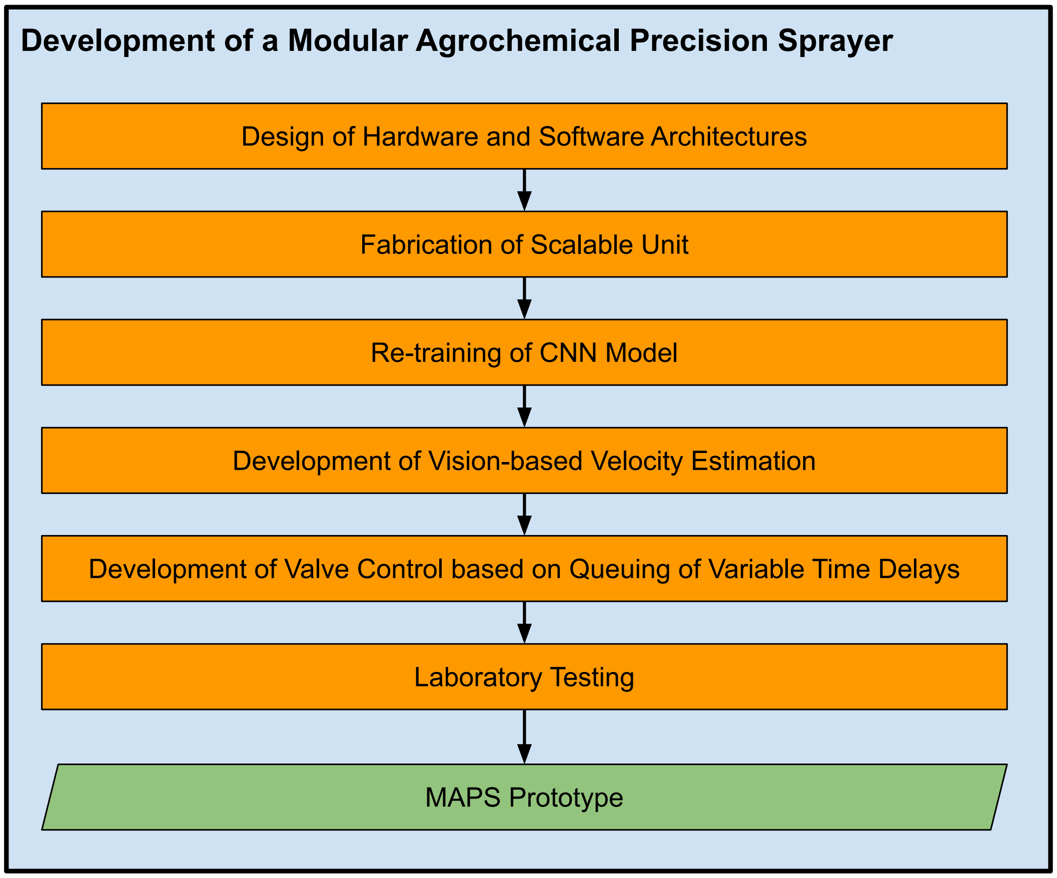
(a) *Base unit*



(b) *Extended configuration*

Figure 23

The Specific Steps in the Development of the MAPS


Development of a Modular Agrochemical Precision Sprayer
Design of Hardware and Software Architectures
Fabrication of Scalable Unit
Re-training of CNN Model
Development of Vision-based Velocity Estimation
Development of Valve Control based on Queuing of Variable Time Delays
Laboratory Testing
MAPS Prototype

3.3.1 Modular Hardware and Software Architecture

The design of the MAPS implemented modular hardware and software to enable component reuse and system reconfiguration. Four hardware modules (vision, sprayer, central, and power), as detailed in Figure 24, were held together by a push-type frame to form a base unit. The components of the scalable unit are shown in Figure 25.

Figure 24

Hardware Schematic of the Vision-Based Precision Sprayer with Modular Components

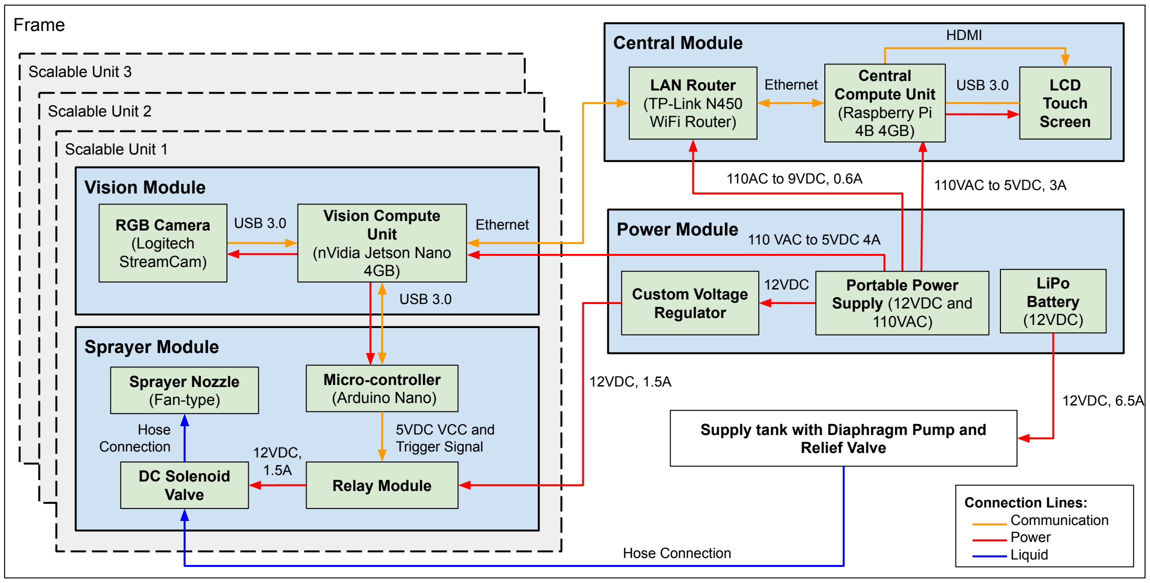
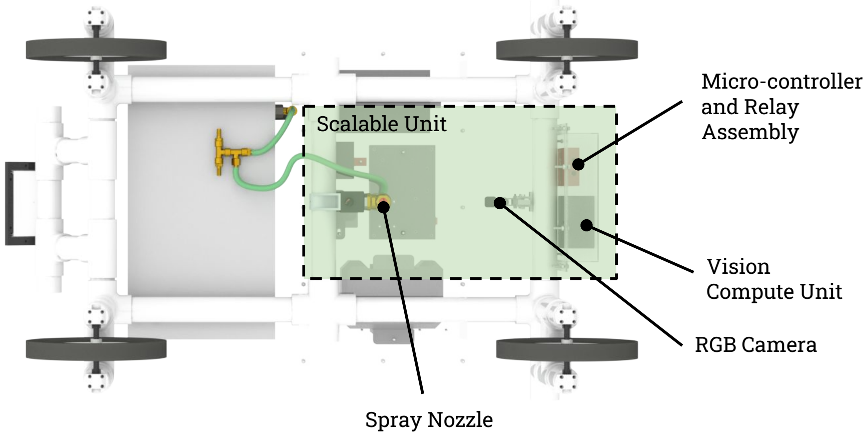


Figure 25

The Bottom View of the MAPS

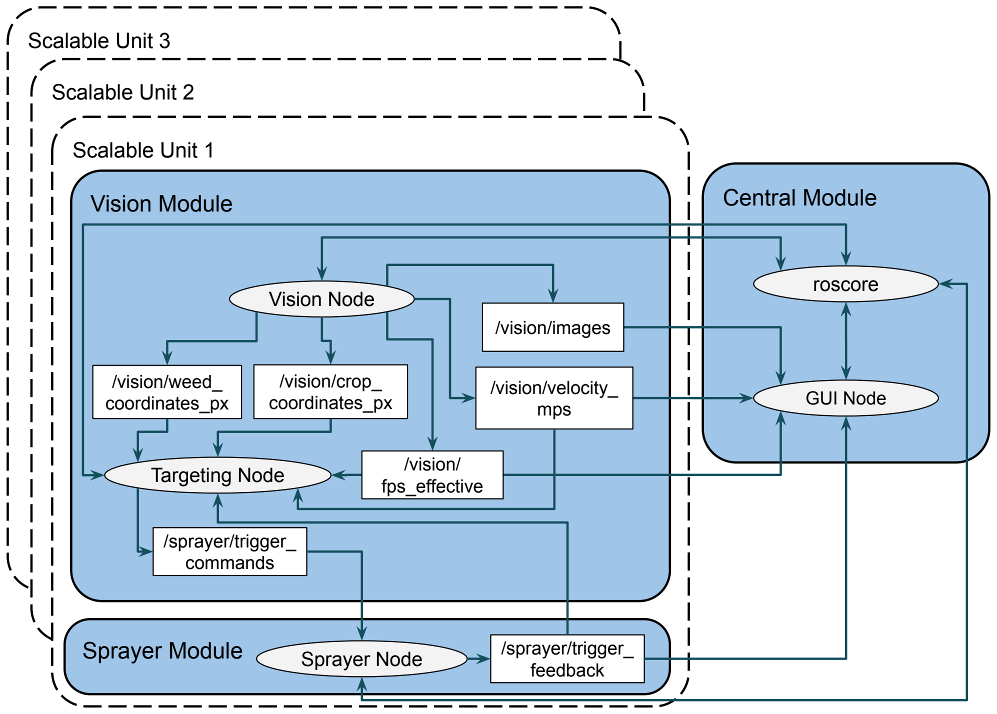


Multiple SUs can be mounted on the base unit through extension structures for broader field coverage, as illustrated in Figure 22b. The number of SUs is only limited by the structure design and the capacity of the central and power modules. The current manually pushed prototype in the test was limited to three modules with the consideration of human power. The same design can be easily expanded to unlimited modules as long as the power and maneuverability are allowed by the prime mover, such as a tractor or unmanned vehicle.

The software architecture (Figure 26) follows a modular structure based on the Robot Operating System (ROS Melodic Morenia). The scripts for each ROS node were written in Python 2.7. Asynchronous data communication between nodes was implemented through the publisher and subscriber model of ROS. The subscriber nodes execute a call-back function when a new message is published in a specific ROS topic. Each callback function had a dedicated processing thread that allowed nodes to fetch messages from multiple topics asynchronously.

Figure 26

General Software Architecture of the MAPS



3.3.1.1 Vision Module.

The vision module was composed of a webcam (Logitech StreamCam) connected to a CUDA-capable low-power device (Nvidia Jetson Nano 4GB), serving as the vision computing unit via Universal Serial Bus (USB). The webcam captures 1280 $px \times 720$ px image at $30fps$ and streams in real-time to the Jetson Nano, which hosts the vision and targeting nodes. The vision node publishes the pixel weed and crop coordinates, $\vec{v}_{travel}$, effective FPS ($fps_{effective}$), and the processed image showing detected plants in their respective ROS topic using tracking and velocity estimation algorithms. The targeting algorithm is hosted in the targeting node. It subscribes to the weed coordinates, travel velocity, and $fps_{effective}$ topics and publishes trigger commands to another ROS topic. On the other hand, processed images showing plant detections can be resized to minimize the transmission size of image data displayed in the Graphical User Interface (GUI).

3.3.1.2 Sprayer Module.

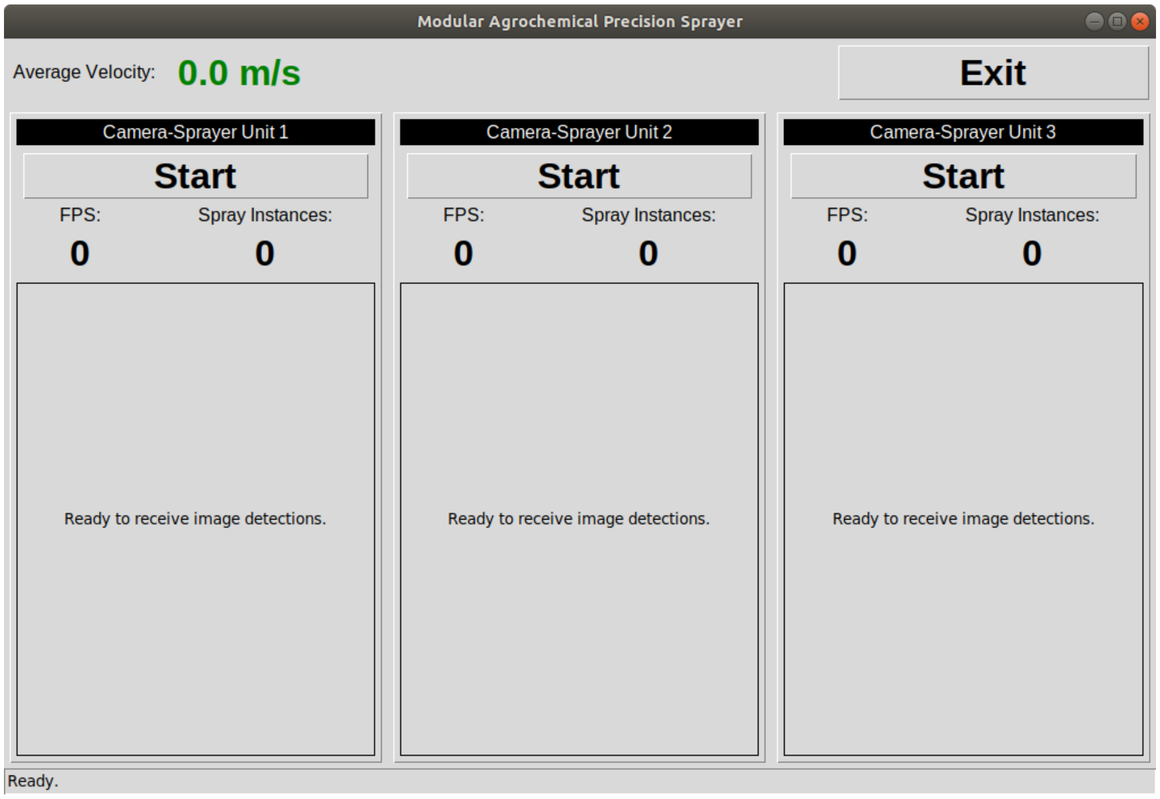
The sprayer module communicates with the vision module through USB. It was composed of an Arduino Nano micro-controller (ATmega328P), a 5-VDC relay module (Arceli KY-019), a 12-VDC solenoid valve (US Solid USS2-00006), and a fan-type sprayer nozzle (Solo 4900654-P). Each solenoid valve was connected to a centralized power sprayer (NorthStar Spot Sprayer) with a 10-gallon capacity, 12-VDC diaphragm pump, and a relief/back-flow valve to maintain an operating pressure of 550 kPa.

The Arduino Nano hosted the sprayer node, interfacing with the communication network via ROSserial, and subscribed to the trigger command topic. It closed the Normally Open (NO) terminal of the relay module when a value of 1 was published to the trigger command topic. Closing the circuit and opening the solenoid valve for a fixed spraying duration. After each spray event, the Arduino Nano checked for the new published value in the trigger command topic.

3.3.1.3 Central Module. The central module was composed of a network hub (TP-Link N450 WiFi Router TL-WR940N), a central computing unit (Raspberry Pi 4B 4GB), and a touchscreen (EVICIV 7-Inch Portable USB Monitor). The network hub served as the connection interface among the edge computing devices that comprised the MAPS. The Raspberry Pi hosted the roscore and GUI nodes that facilitated the launching, monitoring, and terminating of each SU. Figure 27 shows the prototype GUI to control the MAPS. Launching and closing of the slave nodes were accomplished through roslaunch.

Figure 27

Graphical User Interface of the MAPS



Ready to receive image detections.

Ready to receive image detections.

Ready to receive image detections.

Ready.

3.3.1.4 Power Module. The power module comprised a dedicated 20-Ah 12-VDC LiPo battery (Miady LFP16AH) to power the diaphragm pump and a 296-Wh portable power supply (NEXPOW Portable Power Station) to power the rest of the components. Table 7 summarizes the power consumption of each system component. With three SU, the MAPS had a peak power consumption of 160W. Each SU consumed a maximum of 25W during spraying and 4W when nodes were not running. The central module was consuming 9.3W during operation and 3W when idle. On the other hand, the power sprayer consumed a maximum of 75W when the pump was enabled.

Table 7

The Rated Voltages and Currents of Individual Components of the MAPS

Device	Rated Voltage, VDC	Rated Current, A
Raspberry Pi 4B+ 4GB	5	3.0
Wireless Router	5	0.6
Jetson Nano 4GB	5	4.0
Solenoid Valve	12	1.5
Diaphragm Pump	12	6.5

3.3.2 Vision-Based Velocity Estimation

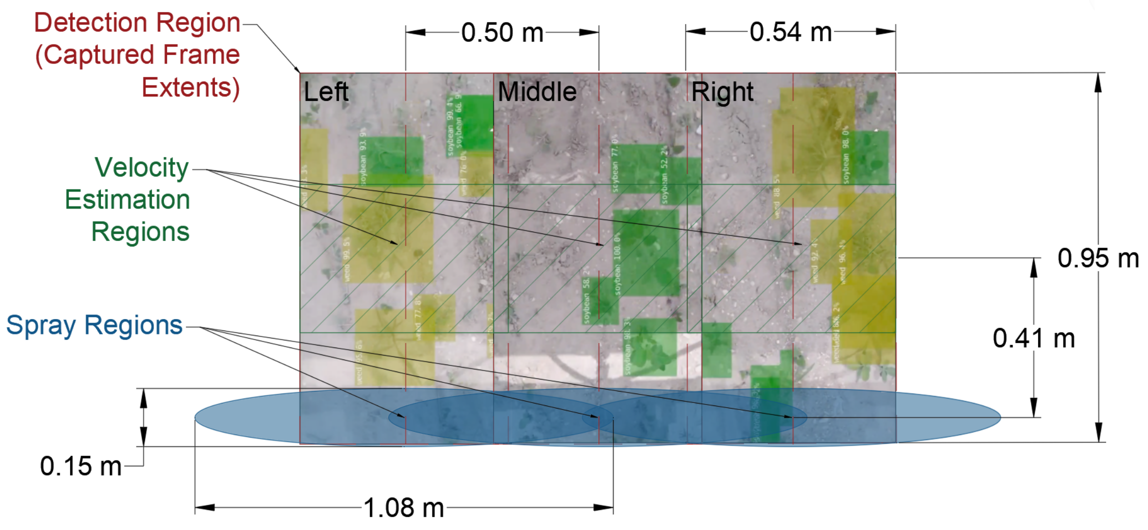
The vision modules used the virtual crop and weed detection bounding box to estimate the travel velocity in a local coordinate system. In this "what you see is what you detect" approach, the vision module can combine plant detection and velocity estimation. It can also easily correct any spraying error with real-time feedback from the incoming video streams based on queuing of variable time delays for valve actuation. Compared with wheel encoders [59] or global positioning systems with real-time kinematics [79], the vision module could be potentially more accurate, faster in obtaining feedback, and more capable of accommodating uneven terrain. The development of the vision-based velocity estimation algorithm is discussed in detail in Chapter 6.

3.3.3 Valve Control by Queuing of VTDs

The effective spray regions covered by the nozzles were positioned away from the velocity estimation region (Figure 28). This approach will decrease the need for computing power while increasing the permissible time delay between detection and spraying to allow for a higher sprayer moving speed than when the detection, velocity estimation, and sprayer regions coincide. However, since the travel velocity can vary during actual field operation, valve control using VTD was implemented. The details of this approach were similarly discussed in Chapter 6.

Figure 28

The Relative Positions of the Detection, Velocity Estimation, and Spray Regions of the MAPS


Detection Region (Captured Frame Extents)
Velocity Estimation Regions
Spray Regions
0.50 m
0.54 m
Left
Middle
Right
0.95 m
0.41 m
0.15 m
1.08 m

3.3.4 Specification and Cost Summary

Table 8 summarizes the technical specification of the MAPS. The prototype had three SUs, separated by 0.5 m and mounted on a push-type frame. The overall cost of all components of the sprayer was approximately USD 2,100. Figure 29 summarizes the cost distribution of each component of the MAPS. The detailed cost of each component is summarized in Table 9.

Table 8

The Technical Specification of the MAPS with Three SUs

Description	**Value**	**Unit**
Fluid Pressure	550	kPa
Nozzle Delivery Rate	1.6	$\frac{L}{min}$
Nozzle Spraying Time	0.2	s
Nozzle Spray Pattern Width	1.08	m
Nozzle Height	0.45	m
Nozzle Spacing	0.5	m
Effective Spray Width	2.08	m
Max. Ground Speed	3.54	$\frac{m}{s}$
Theoretical Field Capacity	2.65	$\frac{ha}{h}$
Camera Resolution	1280×720	px
Average Inferencing Rate	19	fps
Power Consumption	160	W
Min. Operating Time	1.85	h

Figure 29

Cost Distribution of Each Component of the MAPS

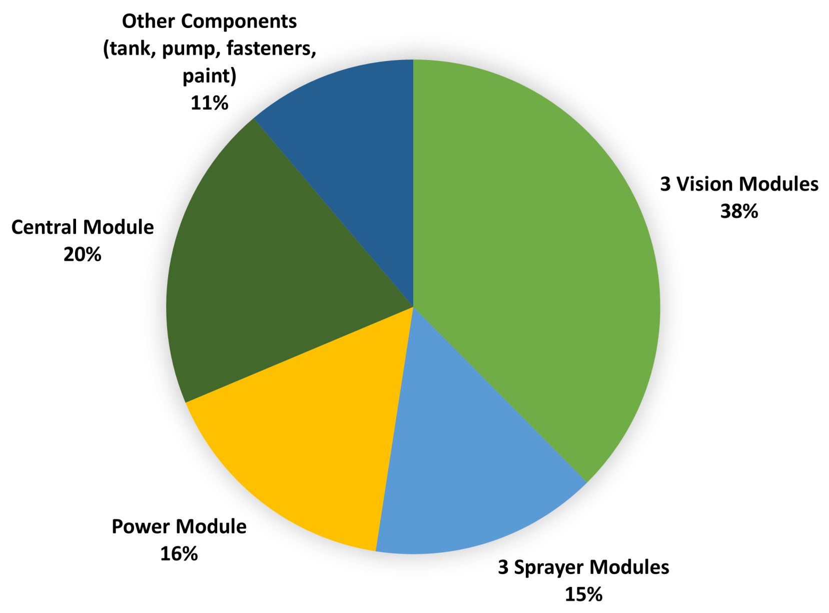


Table 9

Detailed Cost of Each Component of the MAPS

Module	Cost per Module, USD	Unit	Cost, USD
Vision	261.73	3	785.19
Sprayer	103.54	3	310.61
Power	339.18	1	339.18
Central	421.11	1	421.11
Other Components (tank, pump, fasteners, paint)	233.77	1	$233.77
Total Cost			$2,089.86

3.4 Field Testing

Field testing was performed on a soybean field with a broadcast-seeded layout Figure 30 and each step is summarized in Figure 31. Images of soybeans and weeds were collected to form the training and validation dataset of the CNN model for field testing. The spraying accuracy and spray volume reduction of the MAPS on rows with different weed populations were then evaluated.

Figure 30

The Broadcast-Seeded Soybean Field

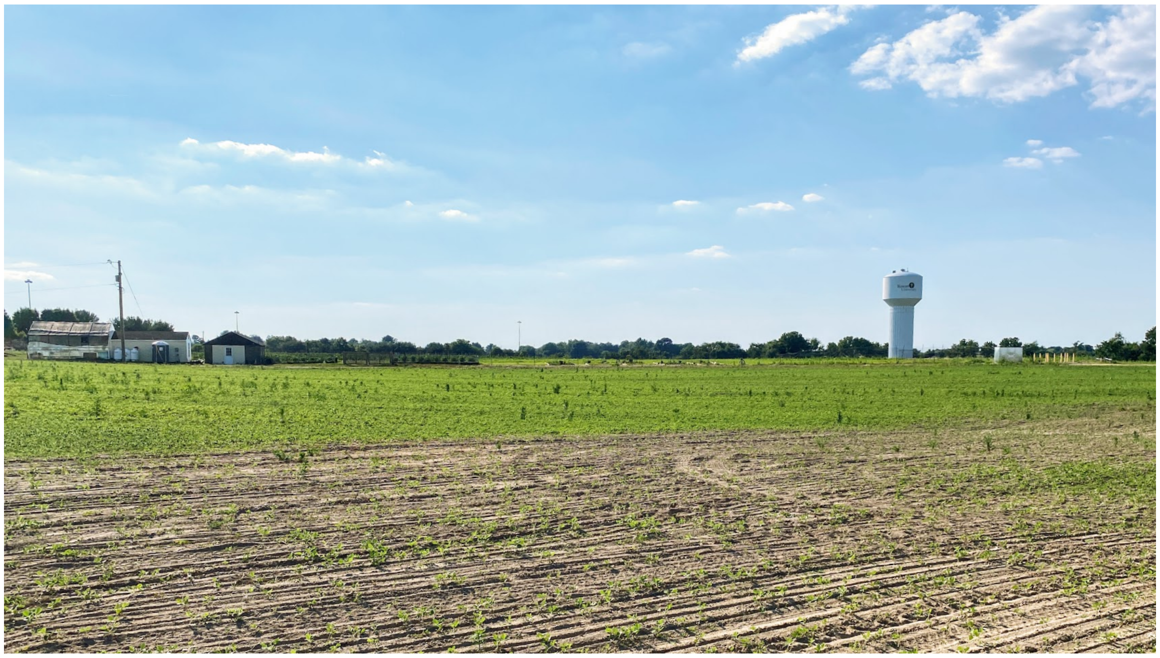
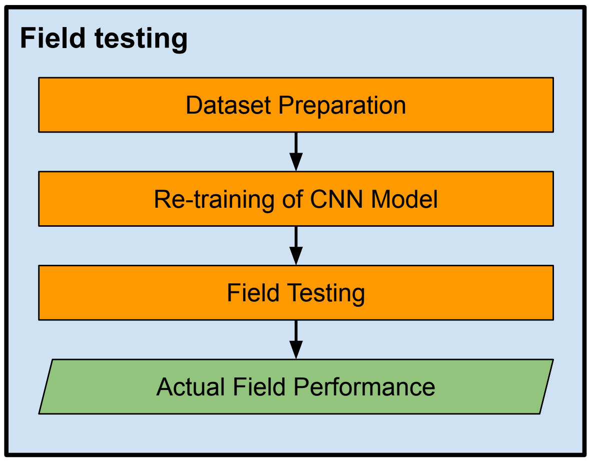


Figure 31

Field Testing Flowchart

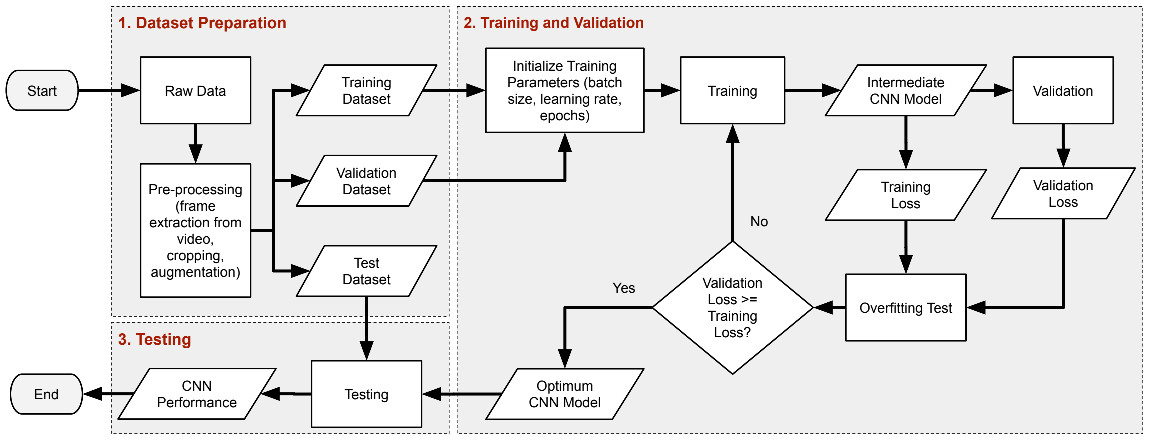


3.5 CNN Model Development

Figure 32 summarizes the general steps for the CNN model development. These steps were (1) dataset preparation, (2) training and validation, and (3) testing. In dataset preparation, sample images of the objects in the operating were collected and annotated. The annotated dataset was split into training, validation, and test sub-datasets. The dataset was then fed to known CNN object detection model architectures and the model was trained until overfitting. Overfitting was determined when the validation loss increased and exceeded the training loss. Finally, the performance of each trained model was determined using the test dataset.

Figure 32

CNN Development Flowchart



3.5.1 CNN Hardware

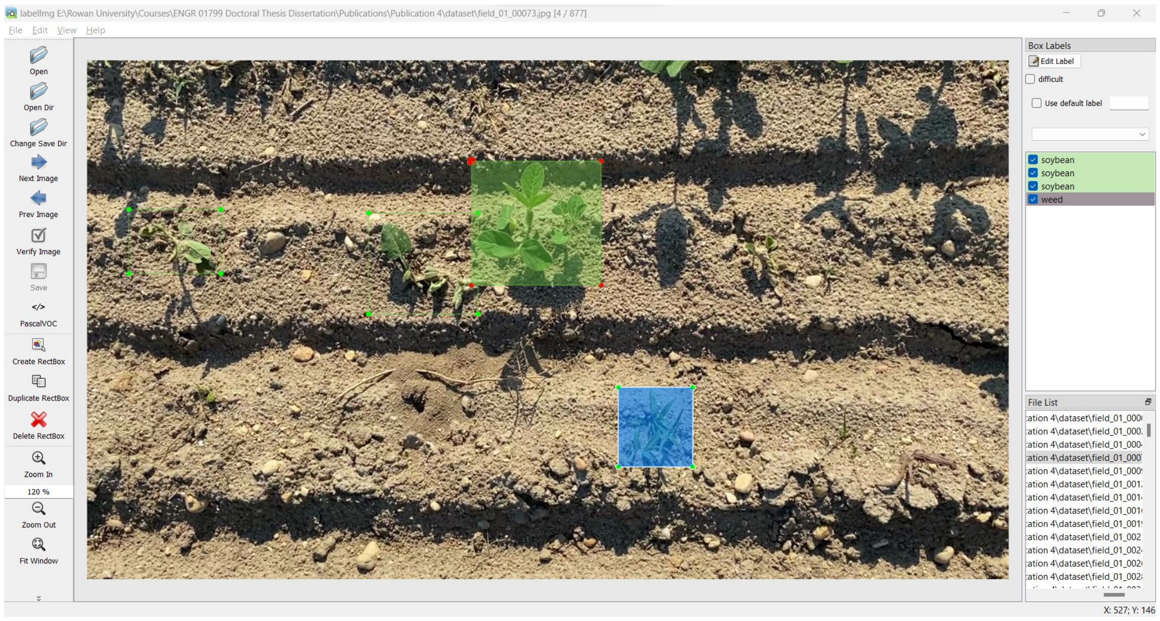
A laptop (ThinkPad T15g Gen 1 laptop) and a low-power device (Jetson Nano) were used to benchmark the processing time of each CNN model. The laptop had a 10th Gen Intel Corei7 10750H (6-core Comet Lake CPU @ 2.60 to 5.0 GHz), 16GB DDR4-3200 (SK Hynix), Nvidia GeForce RTX 2080 SUPER Max-Q (3,072-core Turing GPU @ 735 to 975 MHz, 8GB GDDR6, 80W TDP). The Nvidia Jetson Nano had an ARM A57 CPU (4-core @ 1.43 GHz), a shared 2GB DDR4 memory, and a 128-Core Nvidia Maxwell GPU (10W TDP). As mentioned, this study aimed to identify combinations of CNN architecture and hardware configurations suitable for field robotics. Hence, Nvidia Jetson Nano 2GB was selected due to the available CUDA cores, allowing low-cost and energy-efficient GPU-based inferencing.

3.5.2 Dataset Preparation

Three datasets were prepared for the development of the MAPS. Each dataset was annotated in YOLO format using labelImg [182], as shown in Figure 33. Since CNN was used for detecting weeds, a general binary classification was implemented among the datasets. The datasets were then divided into training, validation, and test datasets.

Figure 33

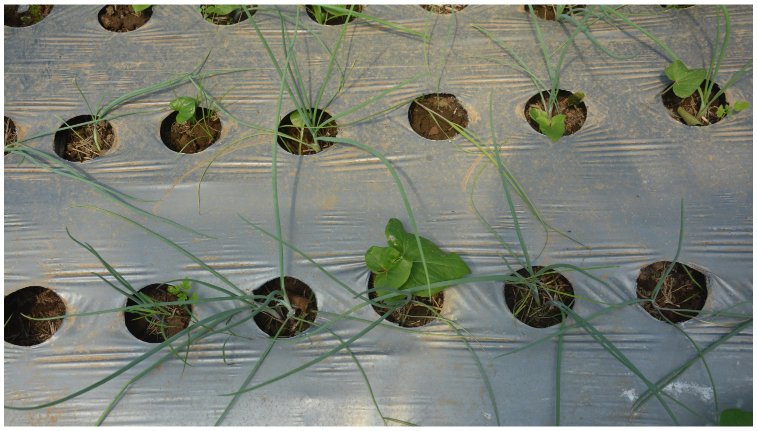
Sample Dataset Annotation Using LabelImg



The first dataset, composed of field images of mulched onion plots (Figure 34a), was used for preliminary benchmarking of existing one-stage object detection models for weed detection. Onion plants growing through the holes of the polyethylene mulch were labeled as "with weeds" or "without weeds".

Figure 34

Sample Raw Images from the Three Datasets Used in the Development of the MAPS



(a) *Onion Dataset*

(b) *Artificial Plant Dataset*

(c) *Soybean Dataset*

Artificial plant images then composed the second dataset, as shown in Figure 34b. The grass and broad-leaf artificial plants were annotated as "weed" and "crop", respectively. This second dataset was used to simulate and prototype the vision and sprayer modules. Finally, field images of soybeans composed the third dataset (Figure 34c). Soybean plants were annotated as soybean, while all non-soybean plants were annotated as "weed". The soybean dataset was then used in the field testing of the MAPS.

3.5.3 Training and Validation

The research explored four one-stage CNN architectures as potential models for weed detection due to their fast inference speed. These CNN models include (1) Scaled-YOLOv4-CSP, (2) YOLOv5s, (3) SSD-MB1, and (4) SSD-MB2. Scaled-YOLOv4-CSP, YOLOv5s, and SSD-MB1 were trained, validated, and implemented using PyTorch machine learning framework [183]. On the other hand, TensorFlow2 [184] was used for SSD-MB2. Table 10 summarizes the reference links of the CNN algorithms used. Each code was implemented in Google Colab.

Table 10

Reference URLs of the CNN Algorithms

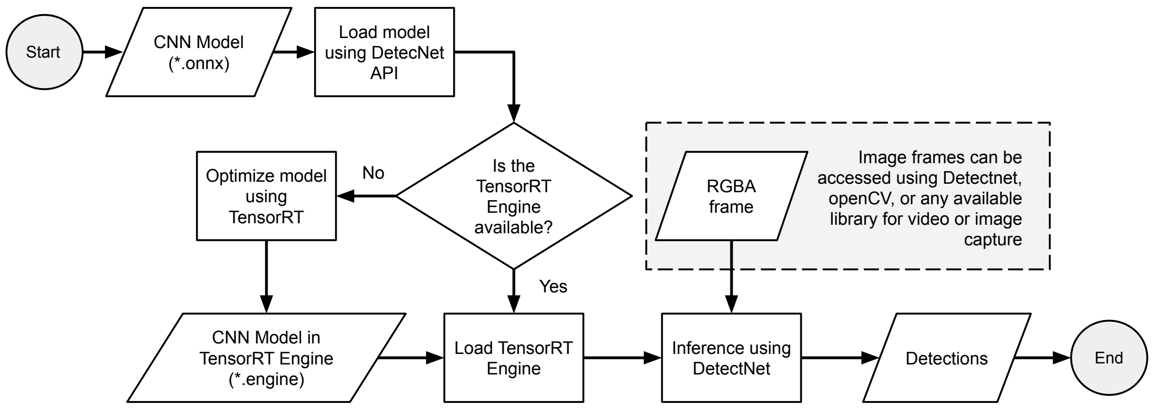
CNN Architecture	Training Algorithm URL
Scaled-YOLOv4-CSP	https://github.com/WongKinYiu/ScaledYOLOv4
YOLOv5s	https://github.com/ultralytics/yolov5
SSD-MB1	https://github.com/dusty-nv/pytorch-ssd
SSD-MB2	https://github.com/tensorflow/models

3.5.4 TensorRT Optimzation

The Nvidia Jetson platform has a native API for object detection, known as jetson-inference [185]. Jetson inference provides high-level GPU-accelerated functions for CNN-based object detection that runs on CUDA, cudnn, and GStreamer. These functions include optimizing supported CNN object detection models using TensorRT, connecting to video-capture devices through gstreamer, and CNN-based inference using the tensorRT-optimized model. TensorRT is a CUDA- and cudnn-based neural network inference acceleration engine from Nvidia [90]. Jetson inference requires a supported CNN model to be in ONNX format and automatically optimizes the ONNX-format CNN model into a tensorRT (*.engine) when the CNN model is initially used. The workflow of using jetson-inference API for object detection is shown in Figure 35.

Figure 35

The General Procedure on CNN Model Optimization Using TensorRT in Nvidia Jetson Platform


Start
CNN Model (*.onnx)
Load model using DetecNet API
Is the TensorRT Engine available?
No
Yes
Optimize model using TensorRT
RGBA frame
Image frames can be accessed using Detectnet, openCV, or any available library for video or image capture
CNN Model in TensorRT Engine (*.engine)
Load TensorRT Engine
Inference using DetectNet
Detections
End

3.5.5 Detection Performance

Ruigrok et al. (2020) [75] recommended (1) image-level and (2) application-level evaluations when testing the weed detection performance of a CNN model for weed detection. Image-level evaluation represents the common method of determining CNN detection performance by standard datasets, as described in the papers of Padilla et al. 2021 [186]. This method is commonly used during non-real-time evaluation. Image-level uses annotated images, specifically the validation and test datasets. A sample count is equal to an instance in an image. For example, an object is accounted twice when present in two distinct images in the dataset.

On the other hand, application-level evaluation is commonly used by precision sprayers with MVS-CNN [79, 73, 74, 75, 86, 76]. Unlike image-level, evaluation is based on the real-time detection of plants. During real-time detection, a sample plant can be detected multiple times in several frames. However, a plant is only accounted for once despite being present in multiple images.

3.5.5.1 Image-Level Evaluation. Precision (p) is a measure of classification performance. It is defined as the ratio of the number of true positive (TP) detections and the sum of true and false positive (FP) detections, as shown below:

$$p = \frac{TP}{TP + FP} \tag{1}$$

Recall (r) is an indicator of the detection rate of the model. It quantifies the ratio of the number of correctly detected objects and all detections, which include both correct and incorrect detections, as expressed below:

$$r = \frac{TP}{TP + FN} \tag{2}$$

Intersection over Union (IoU) measures the correctness of the predicted bounding box (*PBB*) in detecting each object class. It is the ratio of the area covered by the intersection and union of the predicted bounding box and actual bounding box (TBB) in the sample images, as shown by the following equation:

$$IoU = \frac{area(PBB \cap TBB)}{area(PBB \cup TBB)} \tag{3}$$

Average Precision at IoU (AP^{IoU}) is the average of the area under the precision-recall curve at a particular IoU threshold. AP^{IoU} was calculated using the definition of Common Object in Context (COCO) performance metrics [187]. The total sampling points of recalls and precision, I, is the number of spikes in the curve. A spike in a curve is considered whenever the maximum precision value drops. The area is then calculated as the product of the difference between the recalls at the new spike, r_{i+1}, and the previous spike, r_i, and the precision where the latest spike, $p_{interp}(r_{i+1})$, occurred. AP^{IoU} can be expressed mathematically as:

$$AP^{IoU} = \sum_{i}^{I} (r_{i+1} - r_i) p_{interp}(r_{i+1}) \tag{4}$$

AP^{IoU} is a performance metric for a specific category or class, n. Therefore, mean average precision at a threshold IoU (mAP^{IoU}) is calculated to represent the performance for all detection classes and expressed as:

$$mAP^{IoU} = \frac{1}{N} \sum_{i}^{I} AP_n^{IoU}; \in \{class\ 1, class\ 2, ..., class\ N\} \tag{5}$$

Similarly, $mAP^{0.5:0.95}$ is the average of mAP^{IoU} at IoU from 0.5 to 0.95 at a 0.05 interval. Mathematically, $mAP^{0.5:0.95}$ can be expressed as:

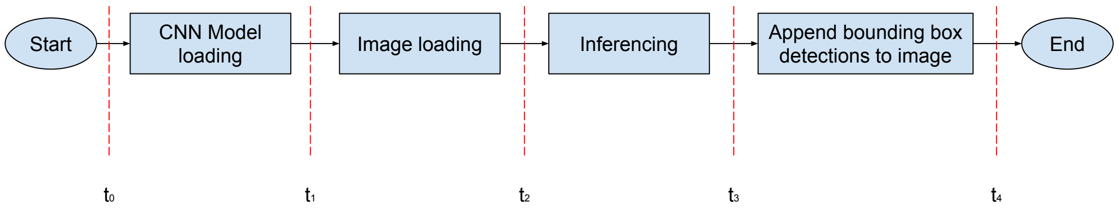
$$mAP^{0.5:0.95} = \frac{1}{11} \sum_{IoU \in \{0.5, 0.55, ...0.95\}} mAP^{IoU} \tag{6}$$

Based on existing CNN detection algorithms [154, 153], image-level inference time (Equation 7), in seconds, was measured before (t_2) and after (t_3) applying CNN-based detection (Figure 36).

$$\Delta t_{inference} = t_3 - t_2 = \frac{1}{fps_{image}} \tag{7}$$

Figure 36

Image Level Processing Times



3.5.5.2 Application-Level Evaluation. Application-level evaluation evaluates the precision (p_d) and recall (r_d) of the MVS-CNN on the total number of samples. A subscript was added to the symbols representing precision and recall to distinguish the application from image-level detection performance. Equation 1 and Equation 2 were similarly used to calculate p_d and r_d, respectively. Detected and correctly classified plants were considered true positives (TP), while detected and incorrectly classified plants were categorized as false positives (FP). Undetected plants of a different class were considered true negatives (TN). Finally, missed detections were classified as False Negatives (FN). The following rules were also adapted from Ruigrok et al. (2020) [75] in counting TP, TN, FP, and FN:

1. Each plant counts as one detection, even if represented in multiple images.
2. If a single bounding box covers multiple plants of the correct class, each of these plants is a *TP*.
3. If a single bounding box overlaps with one or multiple plants of incorrect class, each

incorrectly detected plant is counted as FP.

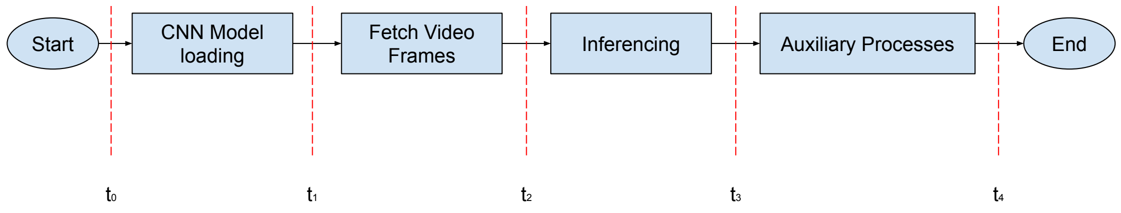
4. If a plant is detected as multiple smaller correctly classified plants, it is only counted as one TP

5. If a plant is incorrectly detected as one or multiple potato plants, it is only counted as one FP.

Additionally, application-level inference speed (*fps*) was the rate of processing an entire loop composed of (1) fetching a frame from the video stream, (2) inferencing, and (3) auxiliary processes, as summarized in Figure 37. In this study, the auxiliary processes include vision-based velocity estimation, calculation and queuing valve opening schedules, and processed image transmission.

Figure 37

Application Level Processing Times



The application-level or effective inference speed ($fps_{effective}$) was then calculated using the following equation:

$$fps_{effective} = \frac{1}{t_4 - t_1} \tag{8}$$

3.6 Statiscal Analysis

R Software was used for statistical analyses. Analysis of Variance and T-tests were used to compare the significant difference among means. When the differences were significant, the means and variance of treatments were compared using Tukey's Honestly Significant Difference (THSD). All statistical analyses were compared at a 95% confidence interval.

Chapter 4

Benchmarking of One-Stage CNN Object Detection Models for Weed Detection[1]

4.1 Introduction

In this chapter, we evaluated combinations of one-stage CNN object detection models and different hardware systems for weed detection. We collected images of mulched onion (*Allium cepa* L.) plots to serve as our dataset in our tests.

Farmers commonly use mulches in combination with mechanical weed or herbicide spraying to minimize weed growth in onion plots [189, 190]. Weeds compete with crops for water, nutrients, light, and space, inhibiting crop growth and development [191]. If left unmanaged, these unwanted plants have the highest capacity to reduce yield [192]. Further, weeds can obstruct humans and farm machinery during operations, reducing field operation efficiency and harvest quality [193].

A robot commonly implements weed detection through a machine vision system (MVS) that utilizes cameras and image processing to recognize plant image features [50]. Current developments in robotic weed control focus on targeted herbicide spraying and mechanical weeding through robotic systems [194, 101]. However, the varying field conditions make weed detection using MVS using traditional image processing techniques difficult [125]. Unlike industrial settings, agricultural production systems are subjected to variable weather, ecological, and topographic conditions. Moreover, the intricate morphological and texture features at different growth stages of plants introduce additional levels of complexity [49].

In recent years, research and development in deep learning resulted in abundant convolutional neural network (CNN) architectures for object detection [61]. This situation enabled and promoted the application of CNN in image analysis problems in agriculture [52, 62]. Nonetheless, the complicated structure of CNN requires sizeable computational

[1]Some parts of this chapter were published in [188].

power. Hence, most accurate modern CNNs do not operate in real-time and need very high computing power to operate [146].

The need for real-time CNN object detection models led to the development of one-stage object detection architectures. Unlike two-stage object detectors that extract region proposals as input to the CNN [148, 66], one-stage object detectors use the entire image to perform dense predictions, resulting in shorter inference time but at lower detection accuracy [63, 148]. However, despite these accomplishments, most agricultural robots utilizing one-stage CNN architectures for plant detection operate below standard operating velocities and field capacity to be accurate [78].

Further, existing studies on weed and crop image classification using machine learning techniques showed a considerable variation in the dataset size and class distribution used in training object detection models. Dataset size varied from 375 to 4550 total samples while weed and crop distribution varied from approximately 7:3 to 1:1 ratios from past studies [157, 158, 156]. Nonetheless, models trained on datasets with low sample sizes and significant class imbalance showed lower performance than models trained on more samples and balanced class sample distribution.

Therefore, this chapter presents our work identifying cost-effective CNN architecture and hardware combinations with optimum detection performance and processing time for weed detection. The effect of using data augmentation to reduce class imbalance on the detection accuracy of the trained model was also evaluated. Four one-stage CNN object detection models (Scaled-YOLOv4-CSP, YOLOv5s, SSD-MB1, and SSD-MB2) were trained, and their performance on a mobile laptop with a powerful GPU (Nvidia RTX2080 Super Max-Q) and a low-power embedded device (Nvidia Jetson Nano 2GB) were evaluated. The training, detection performance, loading time, inference time, and cost efficiency of the four CNN models were compared and assessed to determine which combination was best suitable for our application.

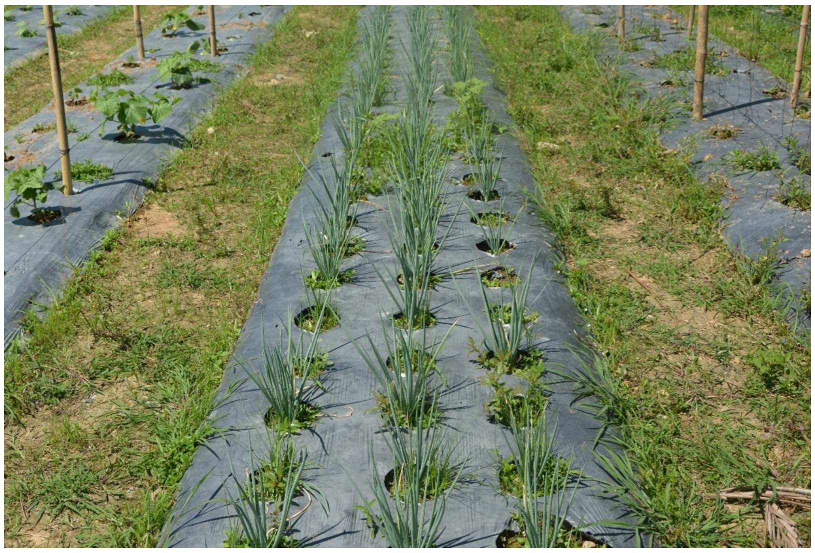
4.2 Methodology

4.2.1 Dataset Preparation

Figure 38 shows the perspective view of the mulched onion plots in Baybay, Leyte, Philippines, intercropped with eggplants. The RGB images of mulched onion plots were captured 51 to 60 days after planting on February 11, 2021, using a Nikon D5200 DSLR hand-held at 1-m height to capture the top view of the field. A total of 292 images at 6000 x 4000 image resolution were collected. Removing duplicates reduced the number of raw images to 153.

Figure 38

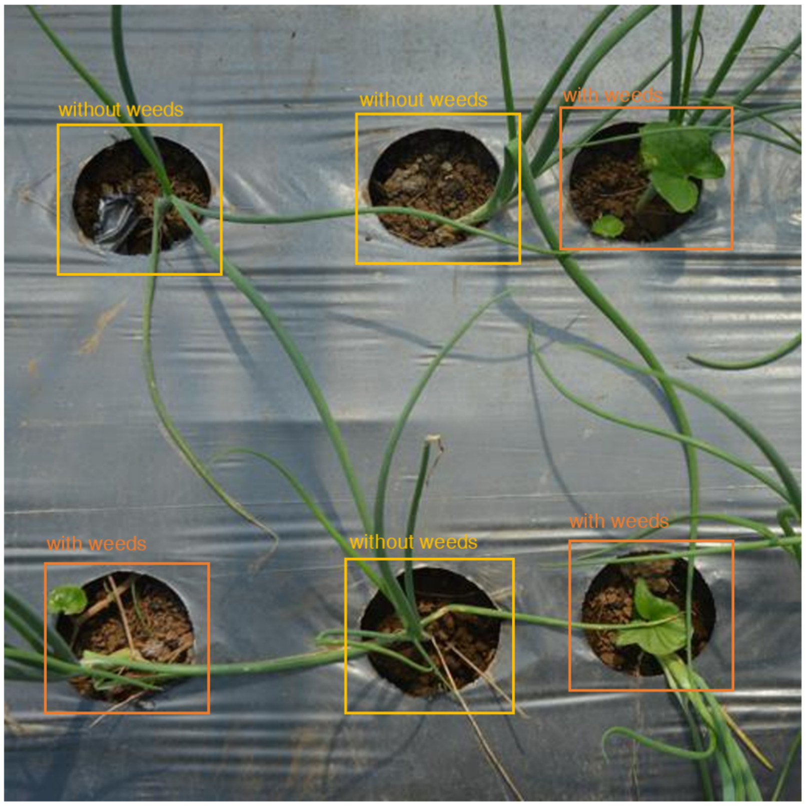
Onion Plant Beds with Black Polyethylene Mulch



A Python script was written to partition and resize the raw images into four sections at 512 x 512 image resolution. The resulting images were annotated with bounding boxes using LabelImg [182]. In addition, the classes “with weeds” and “without weeds” were assigned to holes having and not having the presence of weeds, respectively. Figure 39 shows a sample of the processed image with class annotations.

Figure 39

Cropped Image of a Mulched Onion Bed with Class Annotations



In this study, two datasets were prepared using the processed images to determine the effect of data imbalance on the performance of the models. The first dataset was composed of the processed images randomly divided into training, validation, and test groups, consisting of approximately 60%, 15%, and 25% of the cropped images, respectively. The resulting class distribution for the training dataset with weeds and without weeds were approximately 70% and 30%, respectively, and was highly unbalanced. Table 11 summarizes the count and distribution of these classes as stratified into training, validation, and test samples.

Table 11

Class Distribution of Samples in the Unbalanced Dataset

Group	Image Count	Class		Group Total
		With Weeds	Without Weeds	
Training	382	1524	640	2164
Validation	96	397	149	546
Test	131	622	199	821

Class imbalance can be reduced by either (1) downsizing the class with more samples or (2) upsizing the class with fewer counts. For example, a study comparing performances of different deep learning algorithms to detect soil, broadleaves, and soybeans showed that using an unbalanced but larger dataset increased overall precision than downsizing to a more balanced but smaller dataset [155]. However, the study did not demonstrate the effect on performance using a balanced model through upsizing.

This study performed data augmentation by image rotation, flipping, and random exposure adjustments to generate additional 192 images and reduce the class imbalance in the first dataset. These new images were added to the training and validation group to create a second dataset. To observe changes in detection performance by reducing the class imbalance, the same set of test images from the first dataset was used to test the second dataset. The second dataset then had training, validation, and test group distributions of approximately 65%, 17%, and 18%, respectively. The resulting class distribution for the training dataset for "with weeds" and "without weeds" was about 55% and 45%, respectively. Table 12 shows the detailed distribution of samples after reducing the dataset imbalance.

Table 12

Class Distribution of Samples in the Balanced Dataset

Group	Image Count	Class		Group Total
		With Weeds	Without Weeds	
Training	533	1662	1324	2986
Validation	137	465	326	791
Test	131	622	199	821

4.2.2 Training and Validation

Scaled-YOLOv4-CSP [154], YOLOv5s [153], SSD-MB1 [195], and SSD-MB2 [196] were trained for weed detection due to their short inference times. PyTorch was used for training Scaled-YOLOv4-CSP, YOLOv5s, and SSD-MB1 models while TensorFlow 2 was used to train SSD-MB2.

The initial annotations in YOLO format were used to create the tfrecords for training the SSD-MB2 in TensorFlow 2. In detail, the YOLO annotations (*.txt) were first converted to Pascal VOC (*.xml). The XML annotations were then converted into CSV using a Python script. CSV was the required annotation format of TensorFlow 2 in creating the tfrecords (*.record). All scripts for conversion can be found in a GitHub repository [197].

Pre-trained models on Common Objects in Context (COCO) 2017 from each meta-architecture were starting models in the transfer learning. A cloud computer with an Nvidia Tesla V100 SXM2 GPU with 16GB VRAM and a dual-core Intel Xeon at 2.0GHz with 13.3 GB of RAM was used for training and validation. 24 images per batch were used. Default values of the learning rate and momentum, as set in the respective code repositories of the CNN architectures, were utilized. Overfitting tests were performed to determine the maximum number of epochs used for each combination of the dataset and CNN architecture. Three replications of training and validation were then performed for each CNN architecture and dataset combination.

4.2.3 Performance Testing

4.2.3.1 Detection Performance. In this chapter, an image-level evaluation of the detection performance of the trained CNN models was performed. The $mAP^{0.5}$, and $mAP^{0.5:0.95}$, as described in detail in Chapter 4 and COCO website [187], were used to quantify the overall precision and recall of the tested CNN models. The $mAP^{0.5}$ and $mAP^{0.5:0.95}$ were evaluated using the default validation Python scripts provided in each model's respective repositories.

4.2.3.2 Processing Time. The loading and inference times were evaluated in two different hardware systems capable of Compute Unified Device Architecture (CUDA) for accelerated machine learning (Lenovo ThinkPad T15g Gen 1 laptop and Jetson Nano). Loading time was measured from the initial time of loading the model to the processing of the first frame, as initial testing showed that the loading time of the first frame was longer than the rest of the test frames.

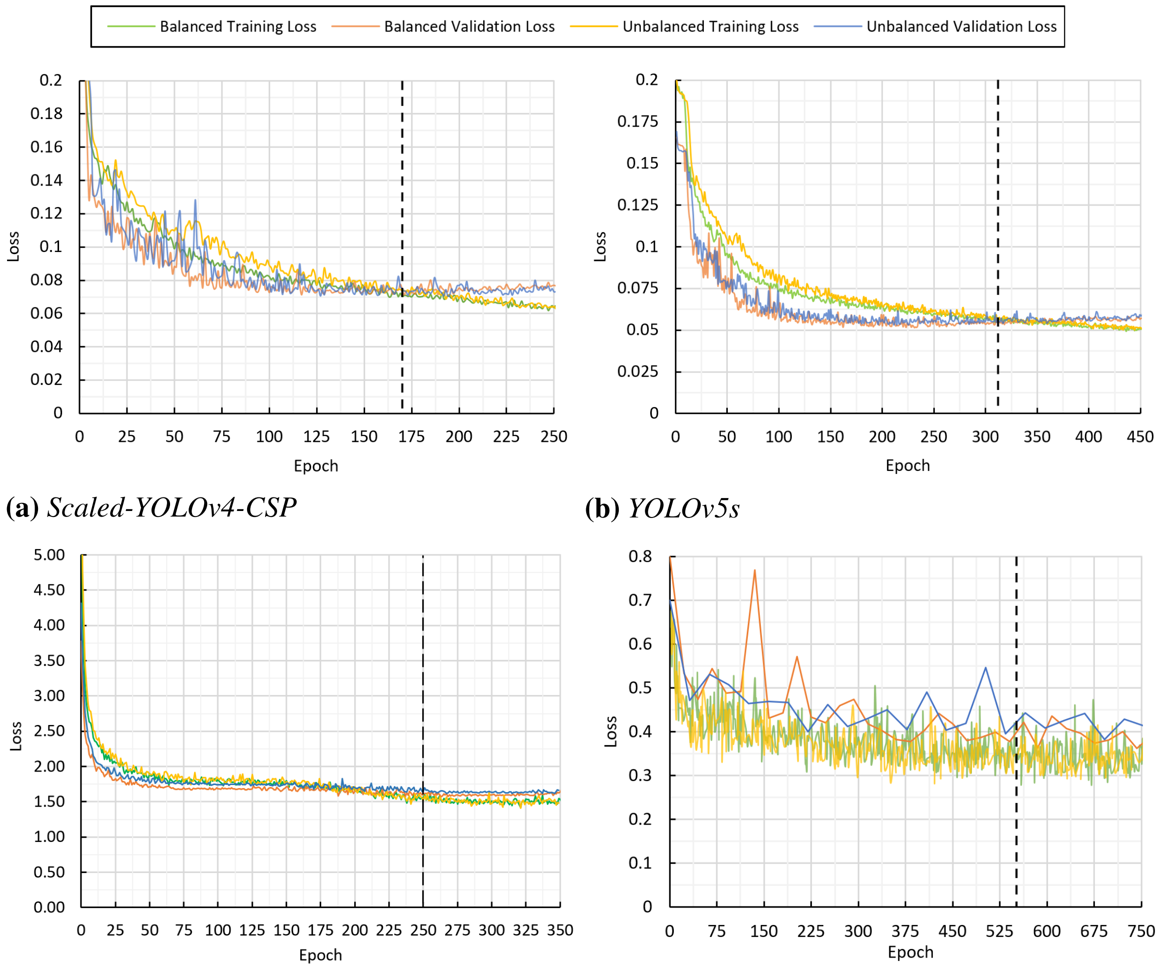
4.3 Results and Discussion

4.3.1 Training of Weed Detection Models

Figure 40 summarizes the training and validation losses of the models. In Figure 40a, the Scaled-YOLOv4-CSP model started to overfit after 170 epochs. The validation losses for both datasets remained increasing while the training losses continued decreasing, an overfitting indicator. Thus, we considered the trained model 170 epochs as the optimum model for the testing.

Figure 40

Training and Validation Losses of Tested CNN Models with the Vertical Broken Lines Representing the Epoch at Overfitting



(a) *Scaled-YOLOv4-CSP* **(b)** *YOLOv5s*

(c) *SSD-MB1* **(d)** *SSD-MB2*

The training of the YOLOv5s model showed that the validation loss started to increase at 250 epochs for the balanced and unbalanced datasets, as illustrated in Figure 40b. Continuing the training showed that the losses converged at 310 epochs, after which the training losses started to be less than the validation losses. Thus, we considered that the model started to overfit, and the models at 310 epochs were used during testing.

Figure 40c illustrates the losses of training SSD-MB1 over 350 epochs. The graph illustrates that losses converged at about 200 epochs. However, the validation loss continually decreased and plateaued at about 250 epochs for both datasets. On the other hand, the training loss continually decreased. Thus, we considered model overfitting at this point and used the trained model at 250 epochs for the testing.

Finally, Figure 40d shows the training and validation losses of the SSD-MB2 model over 750 epochs using balanced and unbalanced datasets. Overfitting was not observed in the first 500 epochs, as both losses continued to decrease. However, after 550 epochs, the validation loss was observed to plateau, whereas the training loss continually decreased. This continued decrease in the training loss increased the difference between the training and validation losses and is an indicator of overfitting. Therefore, the model at 550 epochs was used during performance evaluations of the selected models.

Based on the training results of each model, it can be inferred that the epoch where overfitting started to occur was not highly affected by the ratio of class samples but mainly depended on the object detection architecture. However, it can also be said from the results that after overfitting, running more epochs during training would not increase and may result in a slight decrease in performance.

Table 13 shows the training time and resource consumption of the tested object detection models. The results showed that Scaled-YOLOv4-CSP required the most GPU memory during training and generated the largest model file size since the algorithm extracts more features than the other algorithms. Nonetheless, in general, YOLO-based models took the shortest time to train. On the other hand, the resource consumption of the two SSD-based models was similar. However, SSD-MB2 required the longest training time as more epochs were needed to arrive at an optimum model.

Table 13

Average Resource Consumption of the Tested Object Detection Models During Training

Model	Unbalanced			Balanced		
	Training Time, h	**GPU Memory, GB**	**File Size, MB**	**Training Time, h**	**GPU Memory, GB**	**File Size, MB**
Scaled-YOLOv4-CSP	0.284	10.70	105.50	0.360	10.70	105.50
YOLOv5s	0.259	2.78	14.40	0.379	2.78	14.40
SSD-MB1	0.670	3.99	26.45	0.887	3.99	26.45
SSD-MB2	0.740	4.00	27.50	1.031	4.00	27.50

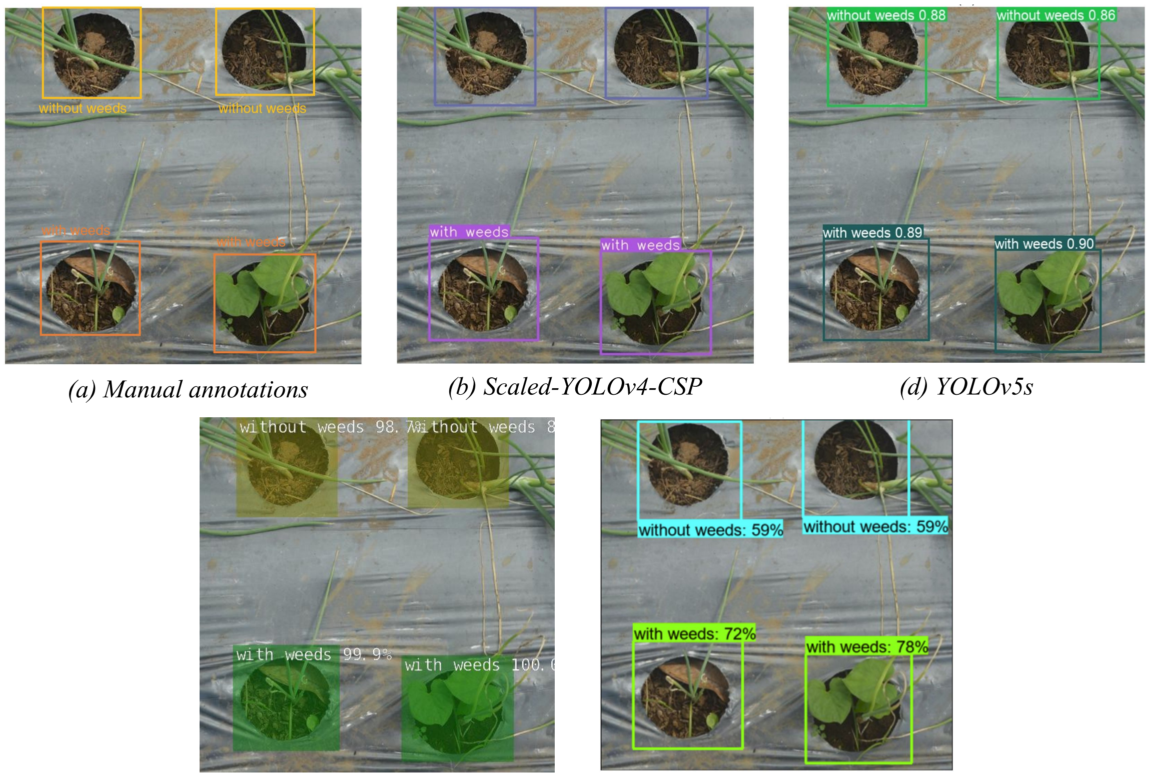
The file size of the model and GPU memory consumption during training were not affected by the number of training samples since the training loads the specified batch of images per training step. Still, they depended on the type of object detection algorithm, as extensive networks extract more features than object detectors with a smaller network. However, in general, increasing the dataset increased the total training time due to the increased number of steps per epoch. In addition, the models of the same family have similar training times. Overall, the YOLO-based models were faster to train than the SSD-based models.

4.3.2 Detection Performance

The trained models were then used to perform detections on the test dataset. Figure 41 illustrates a sample test image with manual annotations and detections from the trained models. For weed spraying, the main objective is to determine if weeds were present in the openings in the polyethylene mulch, accurately size the bound box, and effectively transmit the center coordinate of the bounding box with weed presence to the targeting algorithm. For these reasons, weed recognition and bounding box size and location should be as accurate as possible. Furthermore, the inference time should be as low as possible to prevent missed regions and maximize the forward travel velocity of the robot sprayer.

Figure 41

A Sample Detections Using the Trained CNN Object Detection Models on a Mulched Onion Plot



(a) Manual annotations *(b) Scaled-YOLOv4-CSP* *(d) YOLOv5s*

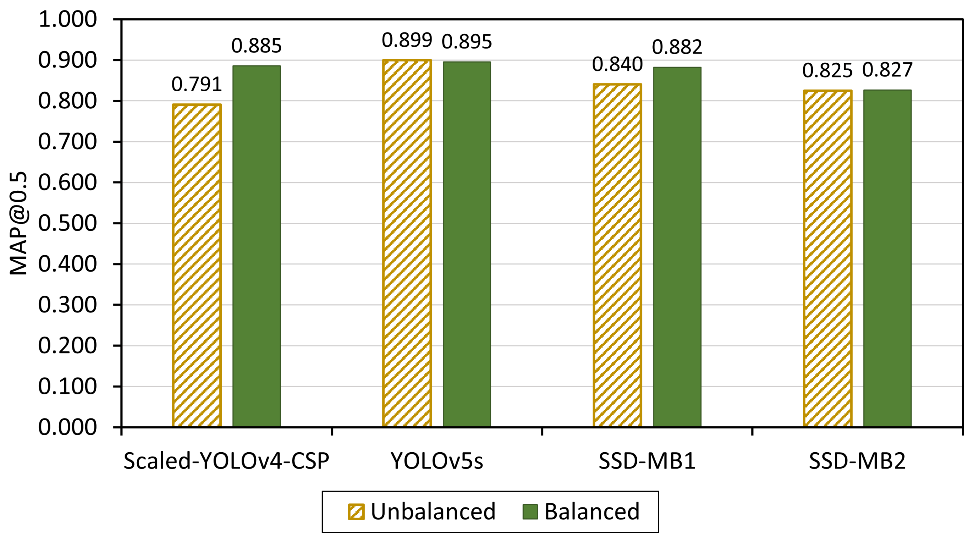
(d) SSD-MB1 *(e) SSD-MB2*

Figure 42 summarizes the average weed detection performance in mulched onion plots of the three models trained on unbalanced and balanced test datasets. For the balanced dataset, Figure 42a revealed that YOLOv5s had the highest $mAP^{0.5}$ at 0.899 followed by SSD-MB1 (0.0.840) and SSD-MB2 (0.825). Scaled-YOLOv4-CSP had the lowest $mAP^{0.5}$

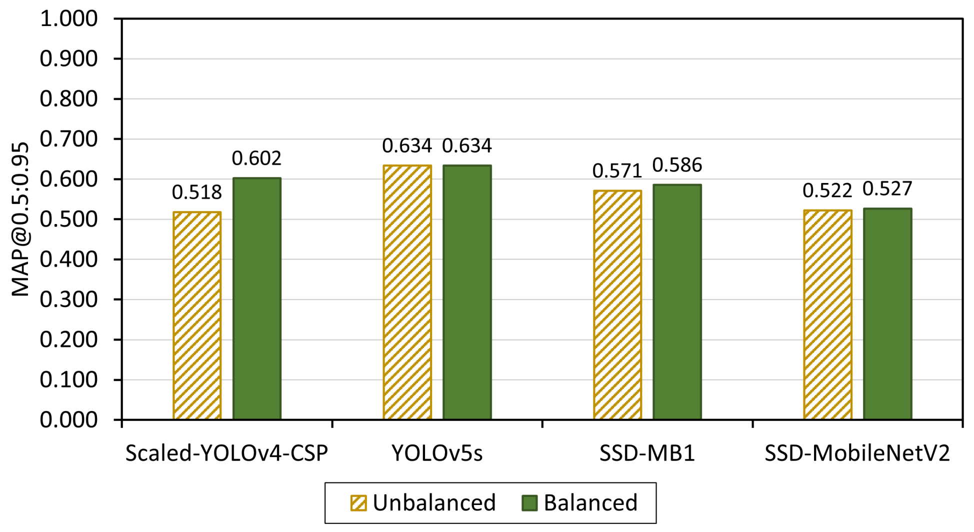
at 0.825. Similarly, Figure 42a also showed that the YOLOv5s model had the highest $mAP^{0.5}$ among the tested models when trained on the balanced dataset.

Figure 42

Average (a) $mAP^{0.5}$ and (b) $mAP^{0.5:0.95}$ of Tested CNN Models on the Test Dataset



(a) $mAP^{0.50}$



(b) $mAP^{0.50:0.95}$

However, there was no improvement in the $mAP^{0.5}$ of YOLOv5s at 0.895 despite training on a balanced dataset. In contrast, the $mAP^{0.5}$ of Scaled-YOLOv4-CSP (0.885) and SSD-MB1 (0.882) improved greatly, resulting in a minimal difference with the YOLOv5s model. SSD-MB2 had minimal improvement despite training on a balanced dataset, resulting in the lowest $mAP^{0.5}$ (0.827) among the models.

We observed the same order of performance among the models at the 0.5 to 0.95 IoU threshold (Figure 42b). YOLOv5s still had the highest $mAP^{0.5:0.95}$ among the tested CNN models in the unbalanced and balanced datasets. Again, there was no improvement observed in the $mAP^{0.5:0.95}$ of YOLOv5s, which remained at 0.634 despite using a balanced dataset. SSD-MB2 still had the lowest performance in the balanced dataset, and the improvement was also minimal, from 0.522 to 0.527. Scaled-YOLOv4-CSP also had the lowest $mAP^{0.5:0.95}$ (0.518) in the unbalanced dataset. YOLOv5s was still followed by SSD-MB1 (0.571) and SSD-MB2 (0.522) in the unbalanced dataset. Similarly, a substantial increase in $mAP^{0.5:0.95}$ in Scaled-YOLOv4-CSP was observed, increasing to 0.602 and surpassing SSD-MB1 (0.586), when trained on the balanced dataset.

Overall, YOLOv5s had the highest *mAP* among the models. YOLOv5s and SSD-MB2 did not benefit from data augmentation and using a balanced dataset. We also observed minimal differences at $mAP^{0.5}$ among the performance of Scaled-YOLOv4-CSP, YOLOv5s, and SSD-MB1 when the models were trained on a balanced dataset. Nonetheless, YOLOv5s exhibited a high $mAP^{0.5:0.95}$ compared to other tested models in either dataset. This result shows that the bounding boxes generated by YOLOv5s were more accurate compared to the other models. However, in reference to field robotics, a high $mAP^{0.5:0.95}$ is only important for operations that require very accurate targeting due to the small effective region of the robotic tool. Examples of these operations could be laser-based or point-spraying. However, for field spraying operations with large effective regions, using fan- or cone-type nozzles, $mAP^{0.5}$ would be acceptable.

Analysis of variance (ANOVA) was performed to determine if the observed differences in the *mAP* among the trained CNN models were statistically significant. Table 14 shows the one-way ANOVA results and indicates that the differences in *mAPs* of the CNN models trained on unbalanced datasets were statistically significant.

Table 14

One-Way ANOVA Tests of the $mAP^{0.5}$ and $mAP^{0.5:0.95}$ of Tested Object Detection Models on Unbalanced and Balanced Datasets

Dataset	$mAP^{0.5}$		$mAP^{0.5:0.95}$	
	F-value	**Probability (>F)**	**F-value**	**Probability (>F)**
Unbalanced	6.361	0.0164*	13.14	0.0019*
Balanced	5.863	0.0203*	16.93	0.0008*

* Statistically different at 5% confidence level.

Tukey's Honestly Significant Difference (HSD) was performed to determine which CNN models trained on balanced (Table 15) or unbalanced (Table 16) datasets were statistically different. Tukey's HSD results in Table 15 show that the higher performance of YOLOv5s than the rest of the trained models on unbalanced datasets was only statistically different from Scaled-YOLOv4-CSP and SSD-MB2. In addition, in Table 16, augmenting the dataset and reducing class sample imbalance reduced the *mAP* difference of YOLOv5s and Scaled-YOLOv4-CSP to levels that the difference was no longer statistically significant. The test also showed the minimal *mAP* increase of SSD-MB2, despite training on the balanced dataset, was statistically lower than the rest of the trained models. In general, the statistical analysis showed that YOLOv5s only had a statistically significant advantage against Scaled-YOLOv4-CSP and SSD-MB2 but not with SSD-MB1.

Table 15

Tukey's HSD Tests of the $mAP^{0.5}$ and $mAP^{0.5:0.95}$ of the Tested CNN Object Models Trained on an Unbalanced Dataset

Compared Models	**p-value**	
	$mAP^{0.5}$	$mAP^{0.5:0.95}$
SSD-MB1 — Scaled-YOLOv4-CSP	0.288	0.132
SSD-MB2 — Scaled-YOLOv4-CSP	0.570	0.997
YOLOv5s — Scaled-YOLOv4-CSP	0.012*	0.003*
SSD-MB2 — SSD-MB1	0.929	0.175
YOLOv5s — SSD-MB1	0.169	0.068
YOLOv5s — SSD-MB2	0.073*	0.003*

* Statistically different at 5% confidence level.

Table 16

Tukey's HSD Tests of the mAP$^{0.5}$ and mAP$^{0.5:0.95}$ of the Tested CNN Object Models Trained on an Balanced Dataset

Compared Models	**p-value**	
	mAP$^{0.5}$	*mAP*$^{0.5:0.95}$
SSD-MB1 — Scaled-YOLOv4-CSP	0.999	0.709
SSD-MB2 — Scaled-YOLOv4-CSP	0.047*	0.005*
YOLOv5s — Scaled-YOLOv4-CSP	0.943	0.253
SSD-MB2 — SSD-MB1	0.060*	0.021*
YOLOv5s — SSD-MB1	0.884	0.056
YOLOv5s — SSD-MB2	0.022*	0.001*

* Statistically different at 5% confidence level.

Table 17 summarizes the t-test to determine the significant effect of data augmentation on the *mAP* of the trained models. The results show that only Scaled-YOLOv4-CSP had a statistically significant increase in *mAP*. Statistically, training on a balanced dataset improved only the *mAP*$^{0.5}$ of SSD-MB1. Further, the results show that YOLOv5s and SSD-MB2 did not benefit from data augmentation. Hence, it can be inferred that data augmentation is no longer needed for weed detection in mulched onions plots when training CNN object detection models using YOLOv5s and SSD-MB2 architectures to achieve optimum detection performance. This condition can make the dataset preparation significantly less tedious since the additional steps to perform data augmentation and annotate the new set of images can be eliminated.

Table 17

The Paired T-Test of the $mAP^{0.5}$ and $mAP^{0.5:0.95}$ of Tested CNN Models on Unbalanced and Balanced Datasets

Model	$mAP^{0.5}$			$mAP^{0.5:0.95}$		
	t	**Mean Difference**	**p-value**	**t**	**Mean Difference**	**p-value**
Scaled-YOLOv4-CSP	4.325	0.0953	0.04951*	5.2372	0.0800	0.0346*
YOLOv5s	-0.3779	-0.0067	0.7418	0.3780	0.0033	0.7418
SSD-MB1	13.104	0.0417	0.0058*	2.644	0.0146	0.1182
SSD-MB2	-0.1368	0.0067	0.8600	0.1525	0.0033	0.8928

* Statistically different at 5% confidence level.

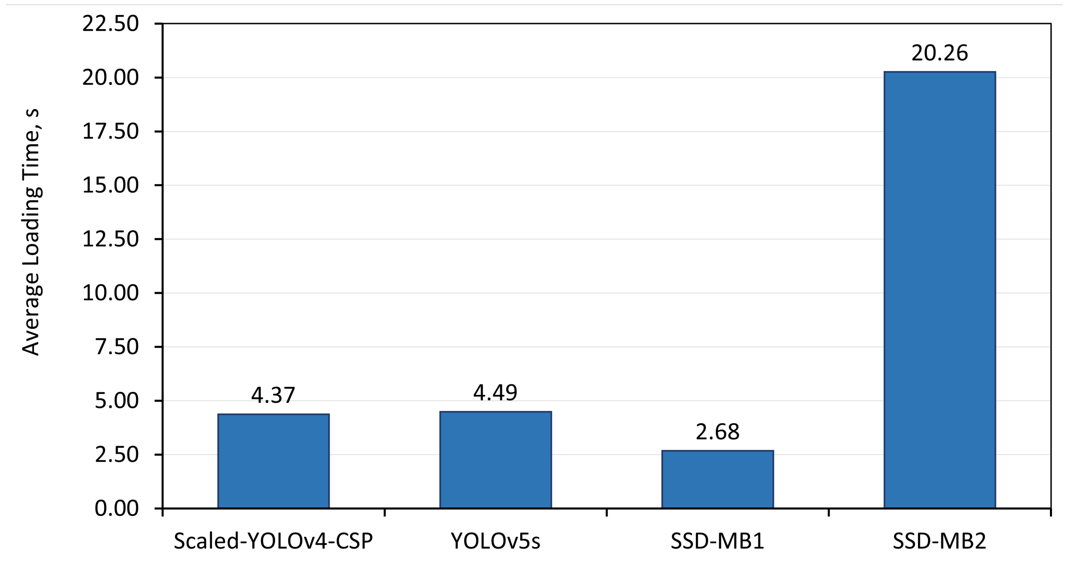
Overall, despite having a higher *mAP* of YOLOv5s compared to the tested models, the performance of YOLOv5s was not statistically different from SSD-MB1. Thus, YOLOv5s and SSD-MB1 deliver the best *mAP* among the tested models.

4.3.3 Processing Time in Different Hardware Setups

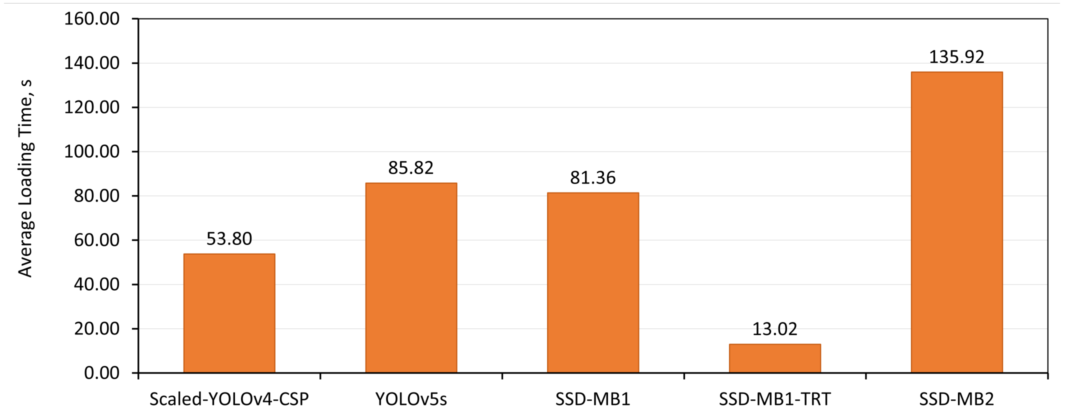
Figure 43 summarizes the loading times of the tested CNN models in the two test hardware. The loading times were measured before loading the model and after processing an initial frame. In the RTX2080 laptop, SSD-MB1 and SSD-MB2 had the fastest (2.68 s) and slowest (20.26 s) loading speeds, respectively. Scaled-YOLOv4-CSP and YOLOV5s took about twice the time to load than SSD-MB1 in TensorRT. Similarly, in the Jetson Nano, the SSD-MB1 optimized in TensorRT (SSD-MB1-TRT) was the fastest to load (13.02 s), while SSD-MB2 remained the slowest (135.92 s). Scaled-YOLOv4-CSP took four times the loading time (53.80 s) of SSD-MB1-TRT while YOLOv5s (85.82 s) and SSD-MB1 (81.36 s) initialized about 6 times that of SSD-MB1-TRT.

Figure 43

The Loading Time, in Seconds, of Trained CNN Models in Test CUDA Devices



(a) *Nvidia RTX 2080 Super Max-Q (80W TDP)*



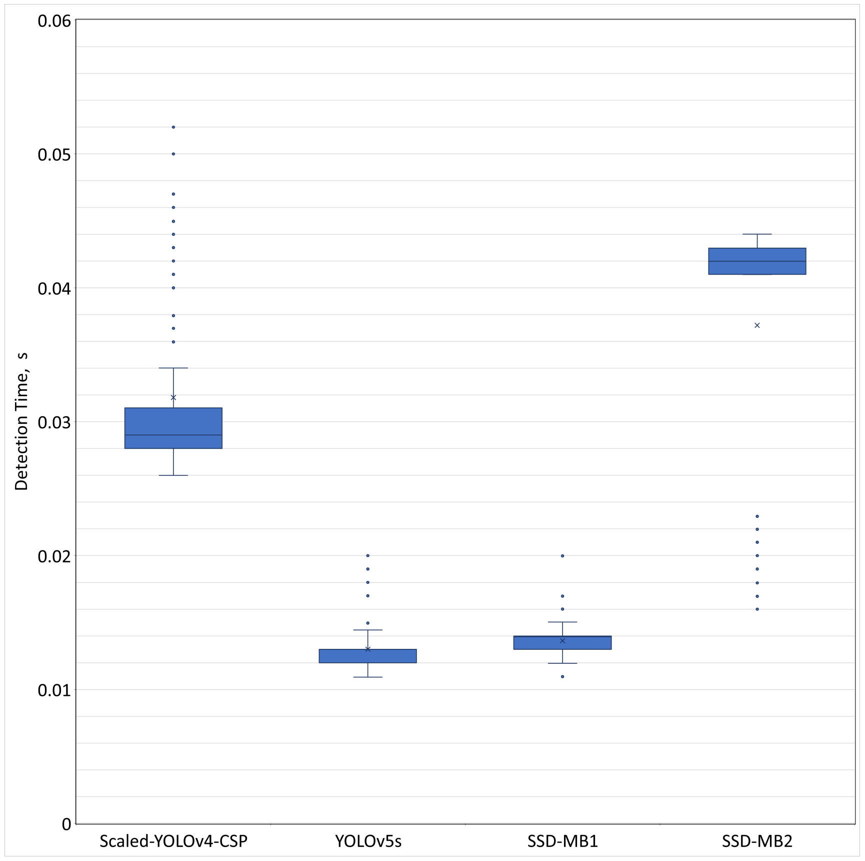
(b) *Nvidia Jetson Nano 2GB (10W TDP)*

Comparing the loading performance between the two hardware, SSD-MB1 without TensorRT optimization loaded 30 times slower in the Jetson Nano than in the RTX2080. In contrast, SSD-MB1-TRT was only about 5 times slower to load in a Jetson Nano than in an RTX2080. Except for the SSD-MB1-TRT, which required approximately 13.2 s to load, the rest of the tested models took approximately 1 to 2 minutes in a Jetson Nano. Conversely, in the RTX2080, all tested models took less than 5 seconds loading time except for SSD-MB2, which required 20 seconds. In reference to field operations, long loading times could be an annoyance to the operator and a source of delays during operation. Thus, when selecting an appropriate CNN model for field operations, the testing showed that Scaled-YOLOv4-CSP, YOLOv5s, and SSD-MB1 in an RTX2080 and SSD-MB1-TRT in a Jetson Nano exhibited compelling loading times.

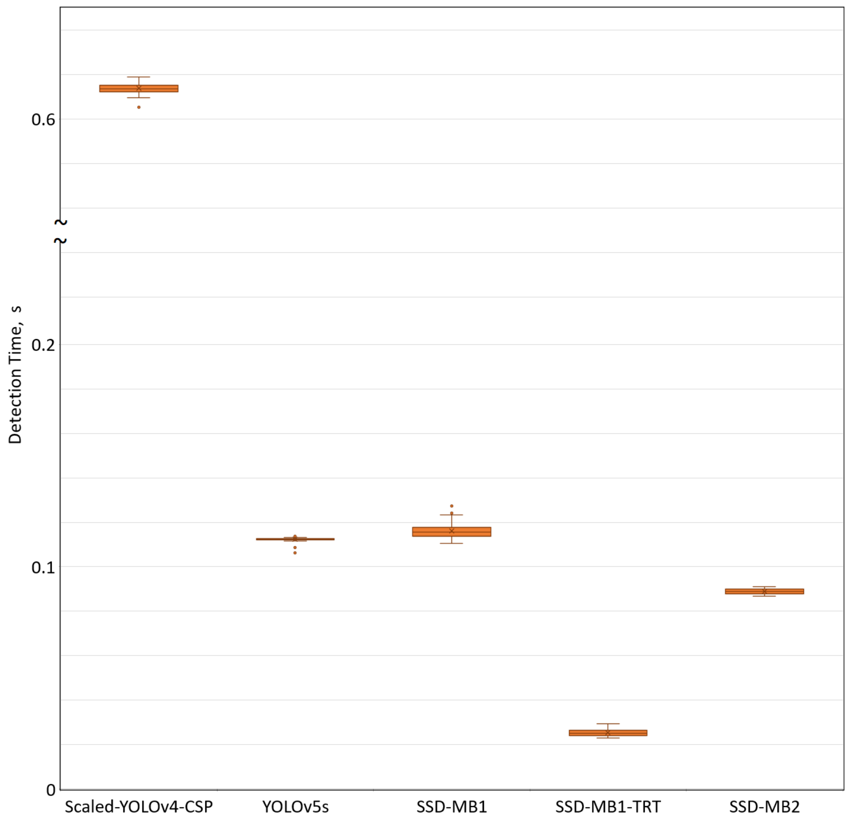
Figure 44 shows the inference time of each detection model running in the two test hardware systems. Inference times were measured before and after detection. Scaled-YOLOv4-CSP, YOLOv5s, SSD-MB1, and SSD-MB2 had mean inference times of 31.81, 13.01, 13.64, and 37.20 ms, respectively, in the RTX 2080 Super. On the other hand, the Scaled-YOLOv4-CSP, YOLOv5s, SSD-MB1, SSD-MB1-TRT, and SSD-MB2 had mean inference times of 613.73, 112.37, 116.17, 25.33, and 83.18 ms, respectively, in the Jetson Nano. These results showed that YOLOv5s and SSD-MB1 were the fastest, on average, among the four tested models in RTX2080. However, TensorRT optimization of SSD-MB1 resulted in SSD-MB1-TRT being about 4.4 times faster than YOLOv5s in the Jetson Nano. On the other hand, Scaled-YOLOv4-CSP had the highest performance loss when running on low-power hardware. Scaled-YOLOv4-CSP took 21 times longer inference time, on average, in the Jetson Nano than in the RTX2080 compared to only 8.5 times that of SSD-MB1 and 1.9 times that of SSD-MB1-TRT. This longer inference time of the Scaled-YOLOv4-CSP than the other models in the Jetson Nano was most likely due to the limited 2GB RAM. As presented in Table 13, Scaled-YOLOv4-CSP required 2 to 4 times the GPU memory of the other tested CNN models during training. Thus, the 2GB shared memory of the Jetson Nano was most likely insufficient to run the trained Scaled-YOLOv4-CSP model.

Figure 44

The Inference Time, in Seconds, of Trained CNN Models in Test CUDA Devices



(a) *Nvidia RTX 2080 Super Max-Q (80W TDP)*

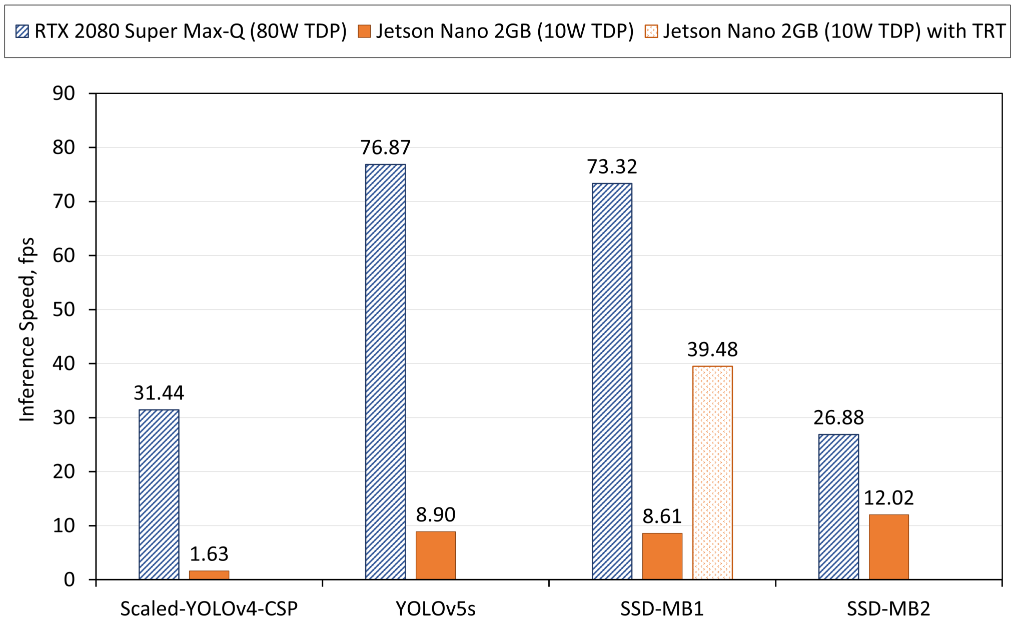


(b) *Nvidia Jetson Nano 2GB (10W TDP)*

Figure 45 compares the image-level inference speed (fps_{image}) of the tested CNN models on the RTX2080 and Jetson Nano. The chart illustrates that SSD-MB1-TRT (39.48 fps) on a Jetson Nano was faster than the rest of the configurations except for SSD-MB1 (73.32 fps) and YOLOv5s (76.87 fps) on the RTX2080. YOLOv5s on the RTX2080 provided the fastest inference speed. However, most weed detection applications in agriculture utilized RGB-imaging systems that had a 30-fps framerate [79, 73, 74, 78]. Thus, CNN and hardware combinations with inference speeds above 30 fps have more than enough speeds to process all of the frames from these imaging systems. Nonetheless, the fps accounted only for the inference process on 512×512 frames and will be slower at higher frame resolutions. The measured values also did not account for the processing times of auxiliary processes that utilized the inference results. These processes may include algorithms for controlling actuators, monitoring the operations, and reading data from other sensors. Nonetheless, these additional processes shall most likely be CPU-based. Thus, the inference speed presented offers a valid reference to compare the computing capability of the GPUs of the tested hardware on different CNN architectures.

Figure 45

The Average Inference Speeds, in fps, of Trained CNN Models in Test CUDA Devices



With respect to processing times, YOLOv5s and SSD-MB1 on an RTX2080 provided the best performance. However, we consider the combination of SSD-MB1-TRT on a Jetson Nano to be a compelling configuration due to its short average loading time of 13.2s and more than 30 fps inference speed at low power requirement. Although SSD-MB1 and YOLOv5s on the RTX2080 were about 1.86 times faster than SSD-MB1-TRT on a Jetson Nano, the Jetson Nano using SSD-MB1-TRT achieved this performance with just 12.5% of the TDP of the RTX2080 test hardware.

4.3.4 Cost Analysis

We procured the laptop with RTX2080 for USD 2626.74 and the Jetson Nano with peripherals for USD 286.60 (Table 18), resulting in the RTX2080 test system costing 9.17 times that of Jetson Nano test system. Figure 46 illustrates the ratio of the inference speed of each CNN model to each test hardware system. Our analysis showed that SSD-MB1 and

YOLOv5s only benefitted from using a low-cost device, such as Jetson Nano, instead of a power RTX2080 laptop. However, the increased cost efficiency using a Jetson Nano 2GB for SSD-MB1 without optimization and YOLOv5s was minimal at only 1.35 times the fps per USD than on an RTX2080.

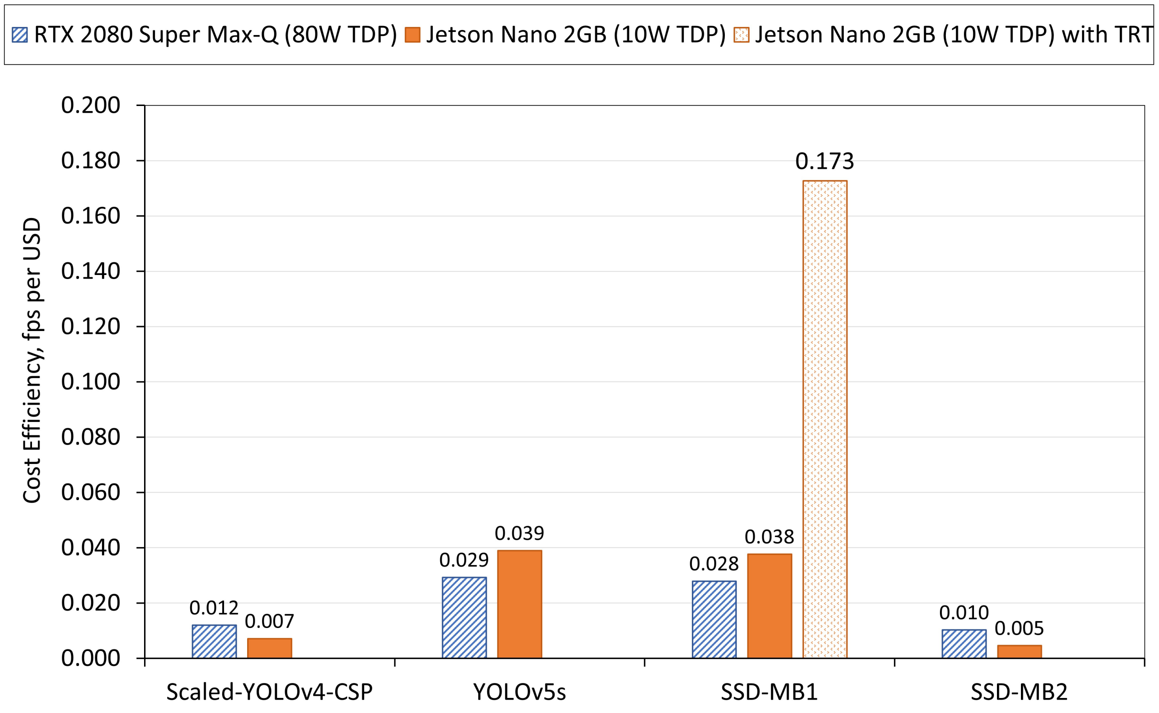
Table 18

Summary of the Component Price of the Jetson Nano Test System

Component	**Cost, USD**
Nvidia Jetson Nano 2GB Developer Kit	62.91
5VDC PWM 40-mm Cooling Fan	7.31
AOC 24B1XHS Monitor	95.95
Logitech K400 Plus Keyboard-Touchpad	21.19
5VDC 4A USB Type-C Power Adapter	12.60
Acrylic Case for Jetson Nano	11.65
Samsung EVO Plus 128GB Micro SDXC	16.99
Total	228.60

Figure 46

Comparison of Cost Efficiency, in fps Per USD, of Trained CNN Models in Test CUDA Devices



The chart revealed that SSD-MB1-TRT on the Jetson Nano test hardware had the highest fps per USD. Furthermore, the chart also showed that SSD-MB1-TRT had the highest benefit in increased cost efficiency (6.17 times) when changing from an RTX2080 to a Jetson Nano due to TensorRT optimization. Finally, Scaled-YOLOv4-CSP and SSD-MB2 showed a decreased cost efficiency when running on a Jetson Nano from the RTX2080 test hardware. These results demonstrate the highest benefit when using a low-power embedded device, such as a Jetson Nano, relative to cost can be attained when optimizing the CNN model with TensorRT.

4.3.5 Performance Summary

In this chapter, we have benchmarked the performance of Scaled-YOLOv4-CSP, YOLOv5s, SSD-MB1, and SSD-MB2 CNN models on high- and low-power hardware systems. Table 19 summarizes the combinations of CNN architecture and test hardware against specific performance criteria during the test. Results showed that SSD-MB1-TRT on the Jetson Nano fulfilled all requirements with high *mAP* and fast processing times at a minimal cost. In contrast, SSD-MB2 proved to be the least compelling architecture in both test hardware, as it exhibited statistically lower *mAP* and slower loading and inference speed compared to other tested models. Further, except for cost efficiency, the combinations of Scaled-YOLOv4-CSP, YOLOv5s, and SSD-MB1 on RTX2080 satisfied the outlined performance requirements. As technology progresses and the cost of these powerful systems lowers, we expect that RTX2080 would be a compelling option.

Table 19

Summary Performance Comparison of Each Test Configuration

Performance	RTX 2080 Super Max-Q (80W)				Jetson Nano 2GB (10W)				
	Scaled-YOLOv4-CSP	**YOLOv5s**	**SSD-MB1**	**SSD-MB2**	**Scaled-YOLOv4-CSP**	**YOLOv5s**	**SSD-MB1**	**SSD-MB1-TRT**	**SSD-MB2**
1. Mean average precision	✓	✓	✓		✓	✓	✓	✓	
2. Fast loading Time	✓	✓	✓					✓	
3. Fast inference time	✓	✓	✓					✓	
4. Cost efficiency						✓	✓	✓	

In the Jetson Nano platform, our tests revealed that model optimization was necessary to satisfy our performance requirements. All models had long processing times when moving from RTX2080 to Jetson Nano despite having acceptable *mAP*. Thus, Jetson Nano provides a cost-effective solution provided that the CNN model can be optimized to reduce processing time.

The next chapter discusses the simulation of plant detection at different inference speeds, travel velocities, and camera configurations. This chapter and the next served as the foundation for designing the modular agrochemical precision sprayer with a CNN-based MVS in the rest of the dissertation.

Chapter 5

Simulation-Aided Development of a Modular MVS-CNN for Plant Detection: Effect of Travel Velocity, Inference Speed, and Camera Configurations[1]

5.1 Introduction

Machine vision systems (MVS) are integral components of field agricultural robots due to the large amount of information that can be extracted from an image scene [50]. MVS was often used to recognize, classify and localize plants accurately for precision spraying [134, 79], mechanical weeding [49], solid fertilizer application [29], and harvesting [199, 70]. However, robust and accurate plant detection using traditional image processing techniques proved challenging due to the vast number of features needed to model and differentiate plant species [29] and work at various farm scenarios [49]. Soil types, crop types, and layouts may deviate significantly among farms while lighting changes accordingly with the weather condition and time of day [98]. Plants also have intricate morphological and texture features at different growth stages that present additional complexity levels [49].

In the last decade, the major developments in deep learning resulted in abundant available state-of-the-art convolutional neural network (CNN) architectures for object detection [61]. Furthermore, pre-trained models and various tools and platforms, such as TensorFlow, PyTorch, and Caffe, have become easily accessible [68]. These conditions enabled and promoted CNN application in image analysis problems in agriculture [62, 52]. However, despite high classification and detection performance, the sizeable computational power requirement of CNN limits its application in real-time operations [61]. As a result, most CNN applications in agriculture were primarily employed in non-real-time scenarios [71, 72, 68] and utilized desktop computer components [74, 73].

Existing MVS-CNN precision sprayers suffer from unoptimized combinations of

[1]Some parts of this chapter were published in [198].

CNN model, computer hardware, camera configuration, and $\overrightarrow{v}_{travel}$ to prevent missed detections. Modeling and simulation are common techniques in engineering to understand the relationship of design parameters and consequently optimize designs. Computer simulations could also save time and cost in robot development as it allows testing of robot software and hardware in a virtual environment [200, 201]. Furthermore, depending on the complexity of the robot simulations, the model can also be implemented using general programming languages and aided by simulation software [202, 203, 201, 204].

Computer simulations were also often used in past studies to characterize the effect of design and operating parameters on the overall performance of agricultural machines for field operations. For example, the simulation of a 2-wheel tractor as a function of engine type, transmission configuration, wheel design, frame structure, and soil mechanical properties allowed researchers to optimize the tractive efficiency [205]. Modeling and simulation were also employed to minimize the vibration of an agricultural boom by optimizing the open-loop gain of a control system for leveling [206]. Computer simulations were also used to estimate the required sprayer spatial resolution of a boom sprayer with MVS-CNN, as influenced by boom section weeds, nozzle spray patterns, and spatial weed distribution [203]. Wang et al. (2018) [201] implemented a computer model to simulate and identify potential problems of a robotic apple-picking arm and developed an algorithm to improve the performance by 81%. Finally, Lehnert et al. (2019) [207] developed a novel multi-perspective visual servoing technique to detect the location of occluded peppers based on a computer-simulated robot-arm-mounted MVS. However, the simulation of an MVS-CNN to illustrate the effect of fps, $\overrightarrow{v}_{travel}$, and camera field of view (S) on actual detection performance (r_d) is yet to be demonstrated.

Chapter 4 evaluated the accuracy and inference speed of popular one-stage CNN architectures for weed detection and showed that a TensorRT-optimized SSD MobileNetV1 (SSD-MB1-TRT) on an Nvidia Jetson Nano provides compelling performance with respect to cost and power consumption. With this previous result, this chapter aims to determine the feasibility of using SSD-MB1-TRT and Jetson Nano for plant detection.

Initially, the process of plant detection as affected by travel inference speed (*fps*), travel velocity ($\overrightarrow{v}_{travel}$), and field of view ($S$) was modeled. We introduced a dimensionless parameter called overlapping rate (r_o). r_o was used to theoretically predict the plant detection rate ($r_{d,th}$) of an MVS as a function of $\overrightarrow{v}_{travel}$, *fps*, and S. To validate $r_{d,th}$, we developed a computer simulation of an MVS for detecting plants in a row-planted field. Simulated plant detections were then performed using published values of $\overrightarrow{v}_{travel}$ and *fps* from existing systems. The obtained simulated plant detection rates $r_{d,sim}$ were then compared to the predicted $r_{d,th}$ and reported detection rates of existing studies.

We then plotted our theoretical model with ranges of $\overrightarrow{v}_{travel}$ of different operations. The generated chart summarizes the applicability of different combinations of $\overrightarrow{v}_{travel}$, *fps*, and S to various agricultural field operations. A reusable and scalable vision module for plant detection based on SSD-MB1-TRT and Jetson Nano platform was developed and tested. The r_d of the developed vision module were compared with $r_{d,th}$ and $r_{d,sim}$. Finally, the theoretical maximum $\overrightarrow{v}_{travel}$ was calculated to determine the feasibility of the developed vision module for precision spraying operations.

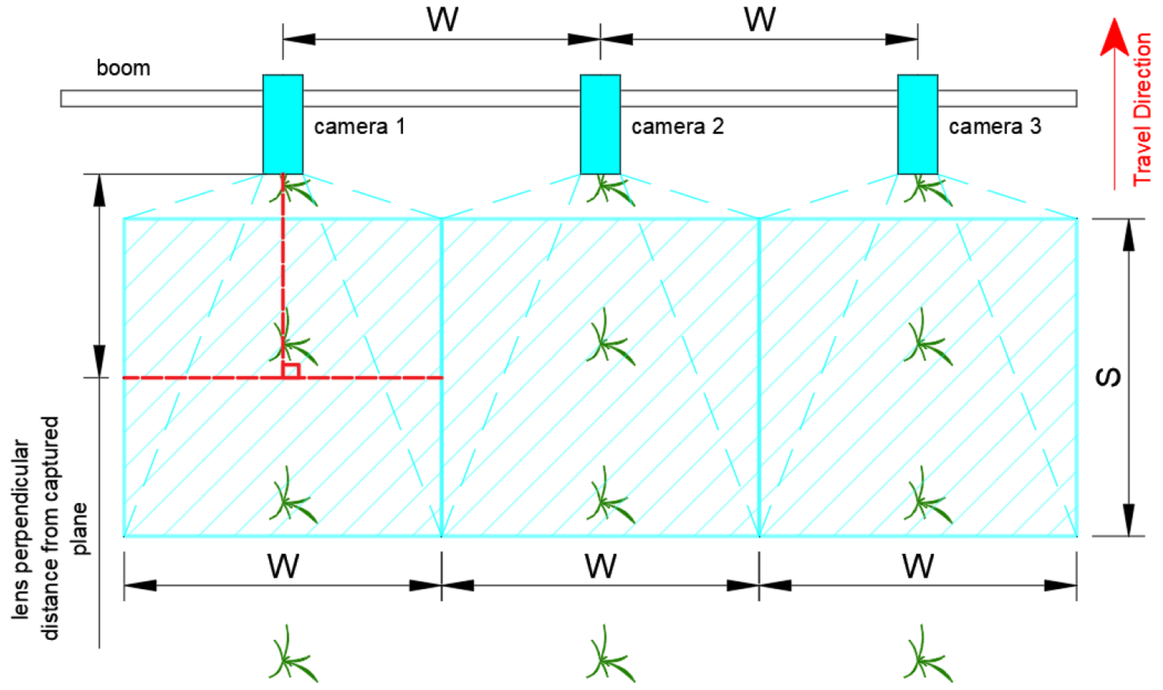
5.2 Materials and Methods

5.2.1 Concept

Cameras for plant detection are typically mounted on a boom of a sprayer or fertilizer spreader [134, 79, 49], as illustrated in Figure 47. They are oriented so that their optical axis is perpendicular to the field and captures top-view images of plants [208].

Figure 47

Camera Mounting Location and Orientation in a Boom (Not Drawn in Scale)



Depending on the distance between the camera lens and captured plane, lens properties, and sensor size, the field of view equals a linear distance. The linear length of the side of a field of view of a frame parallel to the direction of travel was denoted as S, in $\frac{m}{frame}$. For complete visual coverage of the traversed width of the boom, the maximum spacing between adjacent cameras is equal to the length of the side of a field of view perpendicular to the travel direction, denoted as W, in meters per frame [134].

During motion, the traverse distance between two consecutive frames of the camera (d_f), in meters, is equal to the product of $\overrightarrow{v}_{travel}$, in $\frac{m}{s}$, and the time between the frames ($\frac{1}{fps}$), in seconds, shown in Equation 9.

$$d_f = \overrightarrow{v}_{travel} \times \frac{1}{fps} \tag{9}$$

The ratio of S to d_f is proposed as the overlapping rate (r_o), a dimensionless parameter, and is represented by Equation 10.

$$r_o = \frac{S}{d_f} \tag{10}$$

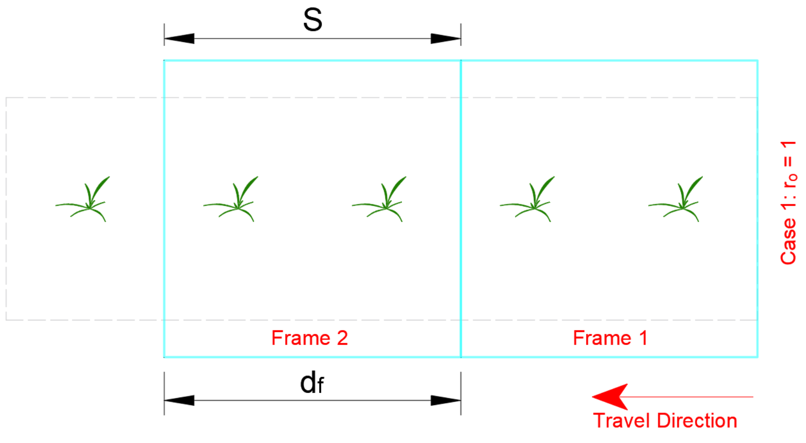
With a single camera, r_o describes the presence of overlap or gap between frames. Depending on r_o, certain regions in the traversed field will be uniquely captured, captured in multiple frames, or completely missed, as shown in Figure 48 and summarized as follows:

- Case 1: $r_o = 1$. When S and d_f are equal, the extents of each consecutive processed frame are side by side. Hence, both gaps and overlaps are absent.
- Case 2: $r_o > 1$. The vision system accounted for all regions in the traversed field, but there is an overlap between the frames. The vehicle can run faster if the mechanical capacity allows it.
- Case 3: $r_o < 1$. Gaps will occur between each pair of consecutive frames, and the camera will miss certain plants. The length of each gap is $d_f - S$ and the gap rate (r_g) the gap rate is calculated as follows:

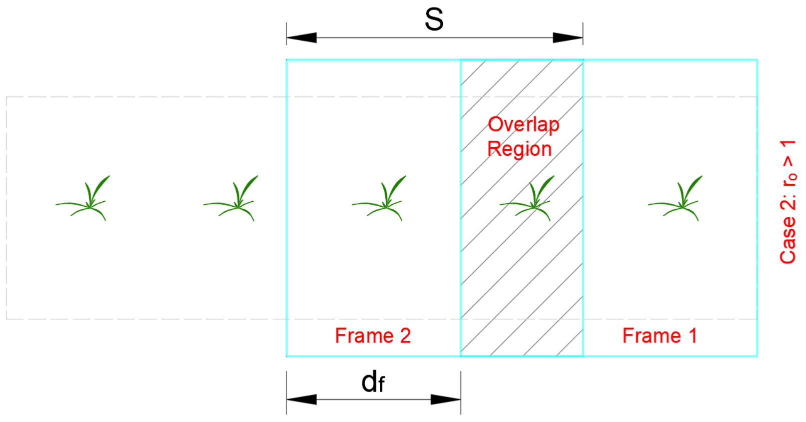
$$r_g = \frac{d_f - S}{d_f} = 1 - r_o \tag{11}$$

Figure 48

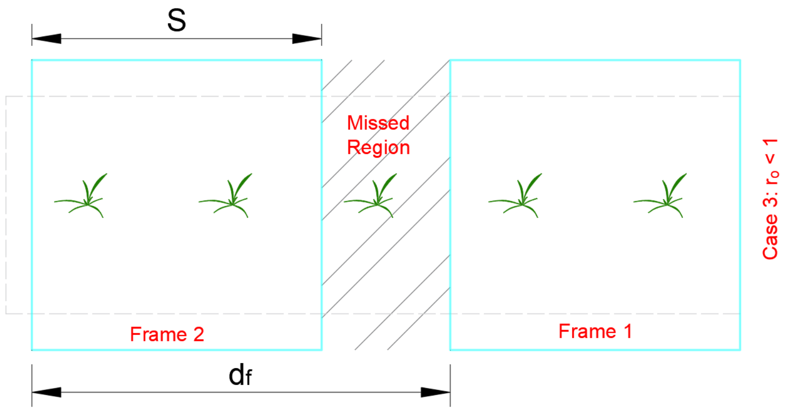
Cases of Gaps and Overlaps in Vision-Based Plant Detection at Different Values of r_o



(a) *No Overlap or Gap ($r_o = 1$)*



(b) *With Overlaps ($r_o > 1$)*



(c) *With Gaps ($r_o < 1$)*

5.2.1.1 Theoretical Coverage Ratio. The maximum detection rate or the theoretical coverage ratio ($r_{d,th}$) can be defined as $\min(1, r_o)$, shown in Equation 12. The $r_{d,th}$ was also a dimensionless parameter.

$$r_{d,th} = \min \begin{cases} 1, & r_o > 1 \\ r_o, & r_o < 1 \end{cases} \tag{12}$$

5.2.1.2 Maximizing Travel Velocity. Setting $r_o = 1$ in Equation 10 will yield Equation 13, which is similar to the equation used by Esau et al. (2018) [134] in calculating the maximum travel velocity of a sprayer. However, equation (5) only describes the maximum forward velocity $\overrightarrow{v}_{travel,max}$ that a vision-equipped robot can operate to prevent gaps while traversing the field as a function of S and fps.

$$\overrightarrow{v}_{travel,max} = S \times fps \tag{13}$$

5.2.1.3 Increasing Maximum Travel Velocity. A consequence of Equation 13 is that increasing S at the same frame rate fps will increase $\overrightarrow{v}_{travel,max}$. Hence, raising the camera mounting height or using multiple adjacent synchronous cameras along a single plant row can increase the effective S of a system. This situation, then, shall increase $\overrightarrow{v}_{travel,max}$ without needing powerful hardware for a faster inference speed. When $r_o < 1$, the number of vision modules (n_{vis}) to prevent missed detection can be calculated using Equation 14. Since r_o represents the fraction of the field a single camera can cover, the inverse of r_o represents the number of adjacent cameras that will result in 100% field coverage. The calculated inverse was rounded up, as cameras are discrete elements.

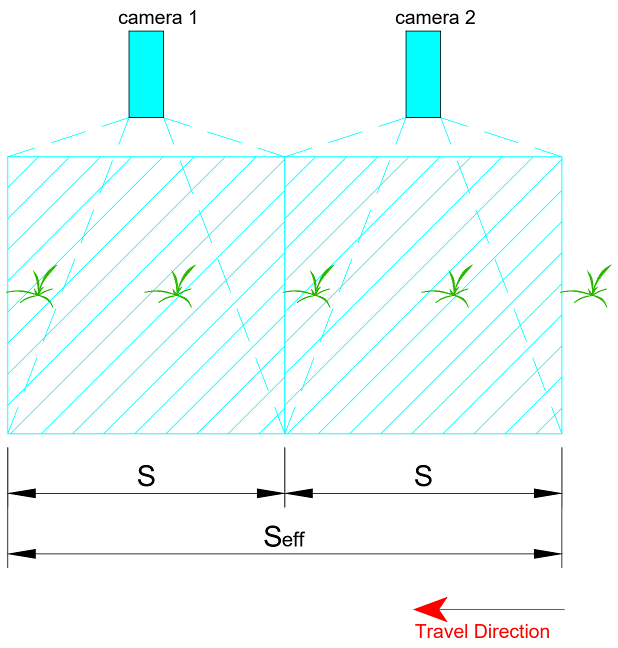
$$n_{vis} = \left\lceil \frac{1}{r_o} \right\rceil \tag{14}$$

The effective actual ground distance (S_{eff}), in meters, captured side-by-side by identical and synchronous vision modules without gaps and overlaps is equal to the product of n_{vis} and S, as shown in Equation 15. This configuration will then allow the use of less powerful devices while operating at the required $\overrightarrow{v}_{travel}$ of an agricultural field operation such as spraying as illustrated in Figure 49.

$$S_{eff} = S \times n_{vis} \tag{15}$$

Figure 49

Illustration of Multiple Adjacent Cameras for Plant Detection at $n_{vis} = 2$ or $S_{eff} = 2S$



5.2.2 *Field Map Modeling*

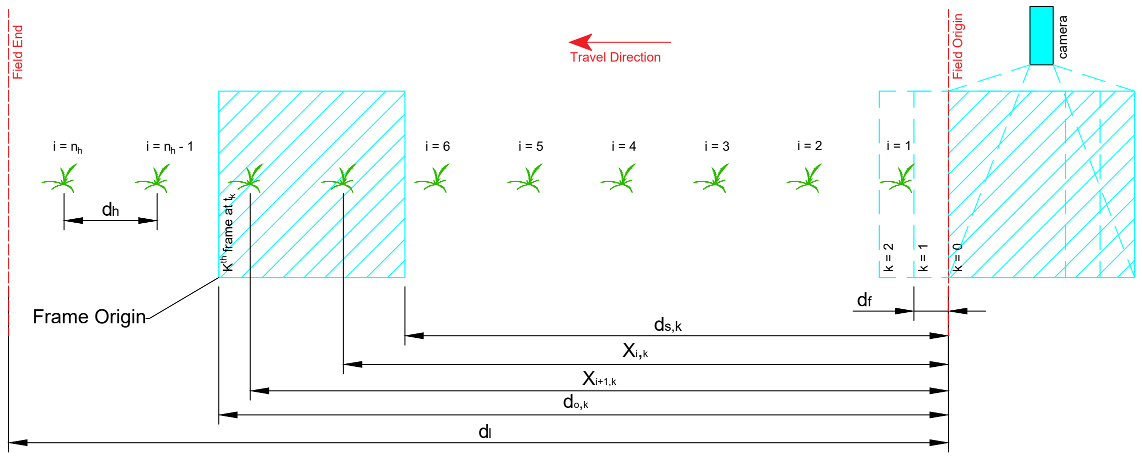
A virtual field was prepared to test the concepts that were presented. A 1,000-m field length (d_l) with crops planted in hills at 0.2-m hill spacings (d_h) was used. The number of hills (n_h) in the entire length of the field and plant hill locations (X_i), in meters, were calculated using equations Equation 16 and Equation 17, respectively. A section of the virtual field is presented in Figure 50. The frame at $k = 0$ is the starting frame just outside the virtual field. The frame at $k = 1$ represents the first frame that entered the virtual field

$$n_{vis} = \left\lfloor \frac{d_l}{d_h} \right\rfloor \tag{16}$$

$$X_i = I \times d_h; i \in 1, 2, \ldots n_h \tag{17}$$

Figure 50

Virtual Field with Map and Motion Modeling Parameters



5.2.3 *Motion Modeling*

The robot was assumed to move from right to left of the field during the simulation, as shown in Figure 50. Therefore, the right border of the virtual area was the assumed field origin. The total number of frames (K) throughout the motion of the vision system then becomes the number of d_f-sized steps to completely traverse d_l, as shown Equation 18.

$$K = \frac{d_l}{d_f} \tag{18}$$

The elapsed time after several frame steps (t_k), in seconds, was calculated by dividing the number of elapsed frames (k) by the inference speed, as shown in Equation 19. t_k was then used to calculate the distance of the left ($d_{o,k}$) and right ($d_{s,k}$) borders of the virtual camera frame concerning the field origin, in meters, using a kinematic equation as shown in Equation 20 and Equation 21, respectively. In Equation 21, S was subtracted from $d_{o,k}$ due to the assumed right-to-left motion of the camera.

$$t_k = k \times \frac{1}{fps} = \frac{k}{fps}; k \in 1, 2, ...K \tag{19}$$

$$d_{o,k} = \overrightarrow{v}_{travel} \times t_k \tag{20}$$

$$d_{s,k} = d_{o,k} - S \tag{21}$$

5.2.4 Detection Algorithm

The simulation was implemented using two Python scripts, which were publicly available on GitHub. The first script, called “settings.py”, was a library that defined the “Settings” object class. This object contained the properties of the virtual field, kinematic motion, and camera parameters for detection. The second script, “vision-module.py”, was a ROS node that published only the horizontal centroid coordinates of the plant hills within the virtual camera frame. The central aspect of ROS was implementing a distributed architecture that allows synchronous or asynchronous communication of nodes [170]. Hence, the ROS software framework was used so that the written simulation scripts for the vision system can be used in simulating the performance and optimizing the code of a precision spot sprayer that was also being developed as part of the future implementation of this study.

When “vision-module.py” was executed, it initially loaded the “Settings” class and fetched the required parameters, including X_i, from “settings.py”. The following algorithm was then implemented for the detection:

1. Create an empty NumPy vector of detected hills.
2. For each k^{th} frame in K total frames:
    (a) t_k, $d_{o,k}$ and $d_{s,k}$ were calculated.
    (b) For each i within the number of hills n_h:
        i. All X_i within the left border, $d_{o,k}$, and the right border, $d_{s,k}$, were plant hills within the camera frame
    (c) Append detected hill indices to list
3. The number of detected hills (n_d) was then equal to the number of unique detected hill indices in the list.

In step 1, an empty vector was needed to store the indices of the detected plant hills. In step 2, each k^{th} frame represents a camera position as the vision system traverses along the field. Step 2a calculated the elapsed time and the left and right border locations of the frame. The specific detection method was performed in Step 2b, which compared the current distance locations of the left and right bounds of the camera frame to the plant hill locations. The plant hill indices that satisfied Step 2b-i were then appended to the NumPy vector. The duplicates were filtered from the NumPy vector in Step 3, and the remaining elements were counted and stored in the integer variable n_d. Finally, the simulated detection rate ($r_{d,sim}$) of the vision system was then the quotient of n_d and n_h, as shown in Equation 22.

$$r_{d,sim} = \frac{n_d}{n_h} \tag{22}$$

5.2.5 Experimental Design

A laptop (Lenovo ThinkPad T15g Gen 1) with Intel Core i7-10750H, 16GB DDR4 RAM, and Nvidia RTX 2080 Super was used in the computer simulation. The script was implemented using Python 2.7 programming language and ROS Melodic Morenia in Ubuntu 18.04 LTS operating system.

The simulation was performed at $S = 0.5m$, based on the camera configuration of Chechliński et al. (2019) [78]. Sensitivity analysis was performed at d_l values of 1, 10, 100, 1000, and 10,000 m. Literature review showed that 20,000 m was used in the study of Villette et al. 2021 [203]. However, the basis for the d_l used in their study was not explained. Hence, sensitivity analysis was performed in this study to establish the sufficient d_l that would not affect $r_{d,th}$ and $r_{d,sim}$. The resulting values of $r_{d,sim}$ were compared to $r_{d,th}$. inference speed of 2.4fps and travel velocity of 2.5 $\frac{m}{s}$ were used for the sensitivity analysis to have an $r_o < 1$ at $S = 0.5m$. If a faster inference speed or slower travel was used, r_o could be equal to or greater than 1. This result will fall into Case 1 or 2 and could not be used for sensitivity analysis.

The model was then simulated at different values of $\overrightarrow{v}_{travel}$ and fps as shown in Table 20 to estimate the detection performance of combinations of CNN model, hardware, and $\overrightarrow{v}_{travel}$. Forward walking speeds using a knapsack sprayer typically ranged from 0.1 to 1.78$\frac{m}{s}$ [80, 81, 82]. On the other hand, the travel velocities of boom sprayers ranged from 0.7 to 2.5 $\frac{m}{s}$ [83, 84, 85]. Solid fertilizer application using a tractor-mounted spreader, on the other hand, operated at 0.89 to 1.68 $\frac{m}{s}$ [29, 204]. Finally, a mechanical weeder with rotating mechanisms worked at 0.28-1.67 $\frac{m}{s}$ [36, 209]. The literature review showed that 0.1 $\frac{m}{s}$ was the slowest [210] and 2.5 $\frac{m}{s}$ was the highest [85] forward velocities found. On the other hand, the mid-point velocity of 1.3 $\frac{m}{s}$ approximates the typical walking speed using knapsack sprayers [80, 81, 82] and forward travel velocities of boom sprayers and fertilizer applicators [83, 84, 85, 210].

Table 20

The Complete Factorial Design for Vision Module Simulation and Theoretical Analyses

Levels	**Parameter**	
	$\overrightarrow{v}_{travel}, \frac{m}{s}$	***fps***
Low (-1)	0.1	2.4
Standard (0)	1.3	12.2
High (+1)	2.5	22

On the other hand, 2.4 and 22 *fps* were the inference speeds of YOLOv3 running on an Nvidia TX2 embedded system and a laptop with Nvidia 1070TI discrete GPU as described in the study of Partel et al. (2019) [79]. Finally, 12.2 *fps* characterized the inferencing time of a custom CNN architecture or SSD MobileNetV1 CNN model optimized in TensorRT and implemented an embedded system [91, 78].

The effect of increasing *S* using multiple camera modules in preventing missed detection was also performed on treatments falling under Case 3.

5.2.6 Vision Module Development

The development of the vision module was divided into three phases: (1) hardware and software development; (2) dataset preparation and training of the CNN model; and (3) simulation and testing.

5.2.6.1 Hardware and Software. Table 21 summarizes the list and function of the hardware components used to develop the vision module. Nvidia Jetson Nano with 4GB RAM was used to perform inferencing on a 1280 x 720 at 30 *fps* video from a USB webcam (Logitech StreamCam Plus). Powering the whole system is a power adapter that outputs 5VDC at 4A.

Table 21

Summary of Vision Module Hardware

Hardware	Model	
	Logitech StreamCam Plus	**Realtime video capture**
Vision Compute Unit	Nvidia Jetson Nano 4GB	Image inferencing
Communication Bus	USB 3.0	Communication with USB devices
Power Adapter	5VDC 4A Power Adapter	Supplies power to the vision compute unit

Table 22 summarizes the software packages used to develop the software framework of the vision module. The software for the vision module was written in Python 2.7. The detectnet object class of the Jetson Inference Application Programming Interface (API) was used to develop the major components of the software framework. Detectnet object facilitated connecting to the webcam using GStreamer, optimizing the PyTorch-based SSD MobileNetV1 model into TensorRT, loading the model, performing inferences on the video stream from the webcam, image processing for drawing bounding boxes onto the processed frame, and displaying the frame. OpenCV is an open-source computer vision library focused on real-time applications. It was used to display the calculated speed of the vision module and convert the detectnet image format from red-green-blue-alpha (RGBA) to blue-green-red (BGR), which was the format needed by ROS for image transmission.

Table 22

Summary of Vision Module Software

Software Package	Function
Nvidia Jetson Inference API	Facilitates camera connection, training of object detection model, converting to TensorRT, loading of object detection model, inferencing, and image processing
Python	General programming language to implement the algorithms
OpenCV	Image processing
Robot Operating System (ROS)	Image data, plant coordinate, and processing time transmission
Ubuntu 18.04 ARM	The operating system for Jetson Nano and hosts the other software packages

To enable modularity, the software framework, as illustrated in Figure 51, was also implemented using ROS version Melodic Morenia, which was the version compatible with Ubuntu 18.04. The vision module node required two inputs: (1) RGB video stream from a video capture device and (2) TensorRT-optimized SSD MobileNetV1 object detection model. It calculates and outputs four parameters, namely: (1) weed coordinates, (2) crop coordinates, (3) processed images, and (4) total delay time. Each parameter was published into its respective topics. Table 23 summarizes the datatype and the function of these outputs.

Figure 51

Software Framework of the Vision Module

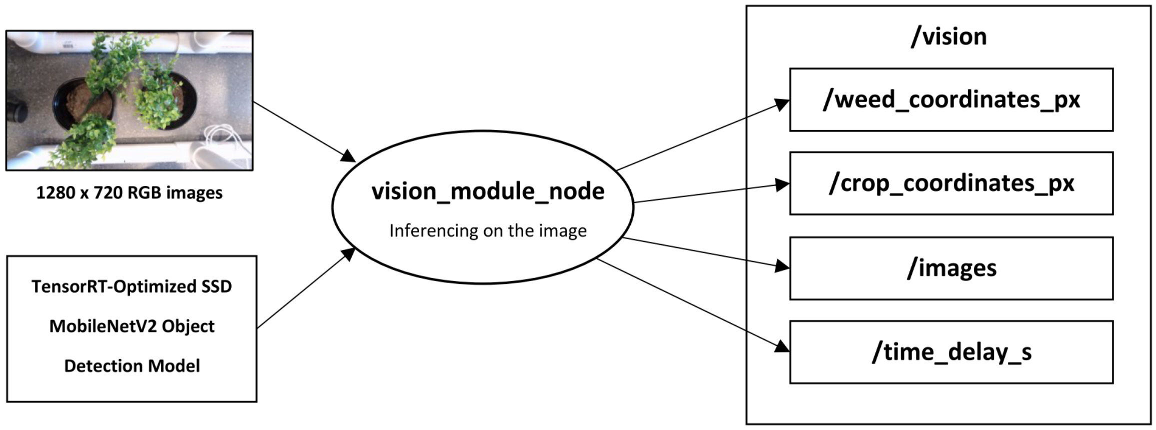


Table 23

Output Parameters of the Vision Module with Their Description

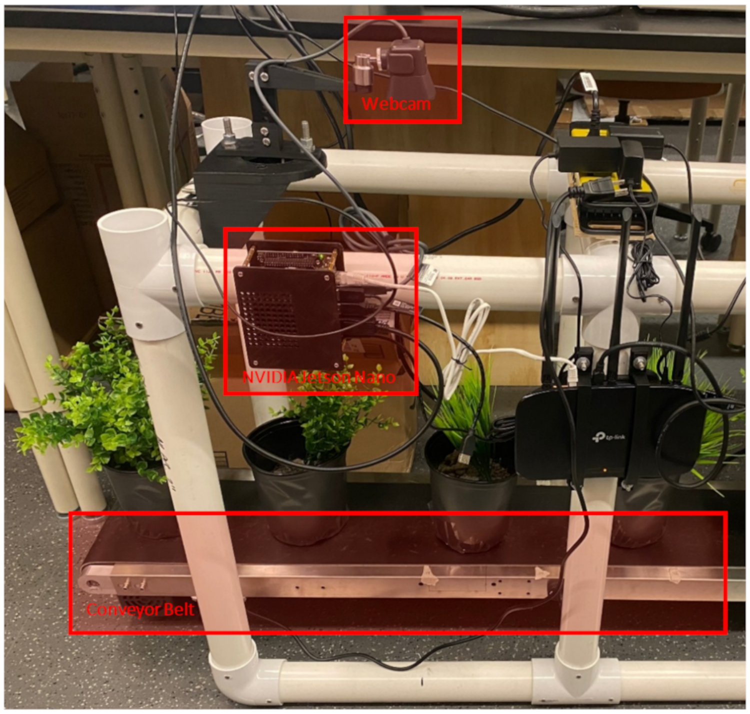
Parameter	**Datatype**	**Description**
Weed coordinates, px	Integer	Array of integers representing the x-coordinate of all detected weed per frame
Crop coordinates, px	Integer	Array of integers representing the x-coordinate of all detected crops per frame
Images	CvBridge	Image data with detections
Time delay, s	Float	Total delay time of the vision module as a result of inferencing, image processing, calculation, and data transmission

5.2.6.2 Dataset Preparation and Training of the CNN Model. Using the Jetson Inference library, a CNN model for plant detection was trained using SSD MobileNetV1 object detection architecture and PyTorch machine learning framework. 2,000 sample images of artificial potted plants at 1280 x 720 composed of 50% weeds and 50% plants were prepared. 80% and 20% of the datasets were used for CNN model training and validation, respectively. A batch size of 4, a base learning rate of 0.001, and a momentum of 0.90 were used to train the model for 100 epochs (5,000 iterations).

5.2.6.3 Testing and Simulation. The performance requirement for the vision module was to avoid missed detections for spraying operations at walking speeds, which was 0.1 $\frac{m}{s}$ at minimum [80, 81, 82]. The Jetson Nano and webcam were mounted at a height where S = 0.79 m (Figure 52). S was determined so that the top projections of the potted plants were within the camera frame. The camera and plants would not collide during motion. A conveyor belt equipped with a variable speed motor was used to reproduce the relative travel velocity of the vision system at 0.1, 0.2, and 0.3 $\frac{m}{s}$. A maximum of 0.3 $\frac{m}{s}$ was used. Beyond this conveyor speed, consistent d_h at 0.2 m was difficult to maintain despite three people performing the manual loading and unloading, as the potted artificial plants were traveling too fast.

Figure 52

Laboratory Test Setup of the CNN-Based Vision Module



A total of 60 potted plants were loaded onto the conveyor for each conveyor speed setting. The detection was done at a minimum conference threshold of 0.5. Detected and correctly classified plants were considered true positives (TP), while detected and incorrectly classified plants were categorized as false positives (FP). Missed detections were classified as False Negatives (FN). The application-level precision (p_d) and recall (r_d) of the vision module were then determined using Equation 1 and Equation 2, respectively.

5.3 Results and Discussion

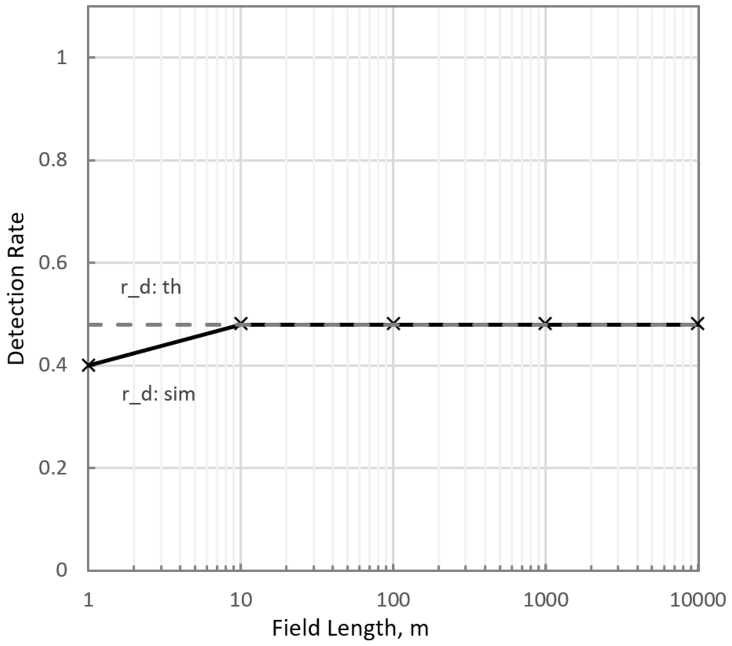
The sensitivity of $r_{d,th}$ and $r_{d,sim}$ to the total traversed distance was first determined to establish the d_l used in the experimental design. The influence of $\vec{v}_{travel}$ and fps at specific S on $r_{d,th}$ and $r_{d,sim}$ were then compared and analyzed. Finally, the results of performance testing the vision module were compared to theoretical and simulation results.

5.3.1 Sensitivity Analysis

As illustrated in Figure 53, the sensitivity analysis results showed that $r_{d,th}$ and $r_{d,sim}$ converged at a 10-m traversed distance. The 20% difference of $r_{d,th}$ from $r_{d,sim}$ can be attributed to the different variables considered to determine each parameter. $r_{d,th}$ used inference speed, travel velocity, and capture width to theoretically calculate the gaps between consecutive processed frames related to the detection rate.

Figure 53

The Theoretical (Solid) and Simulated (Broken-Line) Detection Rates of the Virtual Vision Module



On the other hand, $r_{d,sim}$ determined the detection rate using the number of detected unique plants as influenced by traversed distance, hill spacing, inference speed, travel velocity, and capture width (Sections 2.2 to 2.4). Results showed that simulation better approximated the detection rate than theoretical approaches at less than a 10-m traversed distance. These results infer that at very short distances, $r_{d,sim}$ approximates the detection rate more accurately than $r_{d,sim}$. However, for long traversed distances, the influence of hill spacing on the detection rate was no longer significant, and $r_{d,th}$ can be used to calculate the detection rate.

5.3.2 Effects of Travel Velocity and Inference Speed

Table 24 summarizes the theoretical and simulation. Comparing the $r_{d,th}$ to $r_{d,sim}$ for any combinations of the tested parameters showed that detection rates were equal. Results also showed that there were no missed detections at any $\overrightarrow{v}_{travel}$ when the inference speeds were at 12.2 and 22 *fps* (Case 2), as illustrated in Figure 54 and Figure 55.

Table 24

Theoretical and Simulation Performance at $S = 0.5m$ at Different Travel Velocities and Inference Speeds

Treatment No.	$\overrightarrow{v}_{travel}, \frac{m}{s}$	***fps***	$d_f, \frac{m}{frame}$	r_o	**Case**	r_g	$r_{d,sim}$	$r_{d,th}$
1	0.1	2.4	0.0417	12.00	2	0.00	1.00	1.00
2	0.1	12.2	0.0082	61.00	2	0.00	1.00	1.00
3	0.1	22	0.0045	110.00	2	0.00	1.00	1.00
4	1.3	2.4	0.5417	0.92	3	0.08	0.92	0.92
5	1.3	12.2	0.1066	4.69	2	0.00	1.00	1.00
6	1.3	22	0.0591	8.46	2	0.00	1.00	1.00
7	2.5	2.4	1.0417	0.48	3	0.52	0.48	0.48
8	2.5	12.2	0.2049	2.44	2	0.00	1.00	1.00
9	2.5	22	0.1136	4.40	2	0.00	1.00	1.00

Figure 54

Simulated Plant Hill Detection Rates of the Vision Module Different Velocities

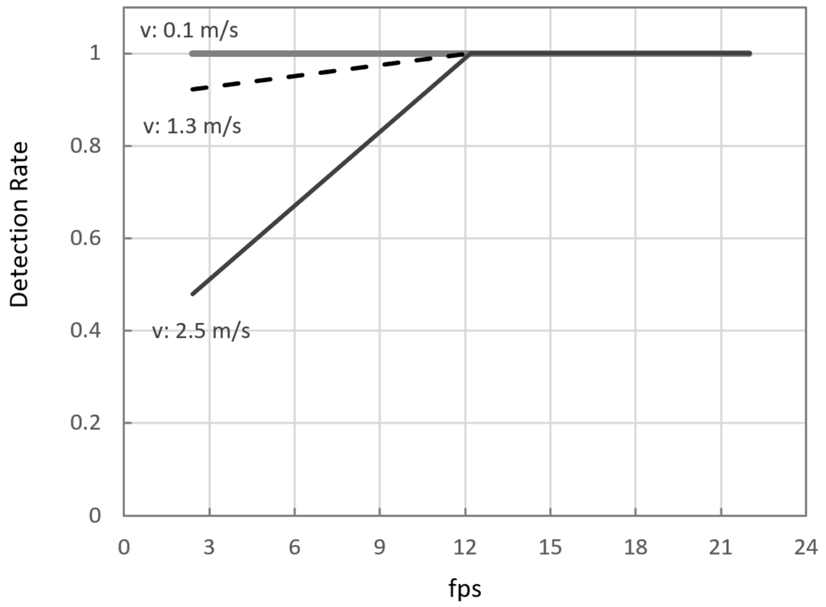
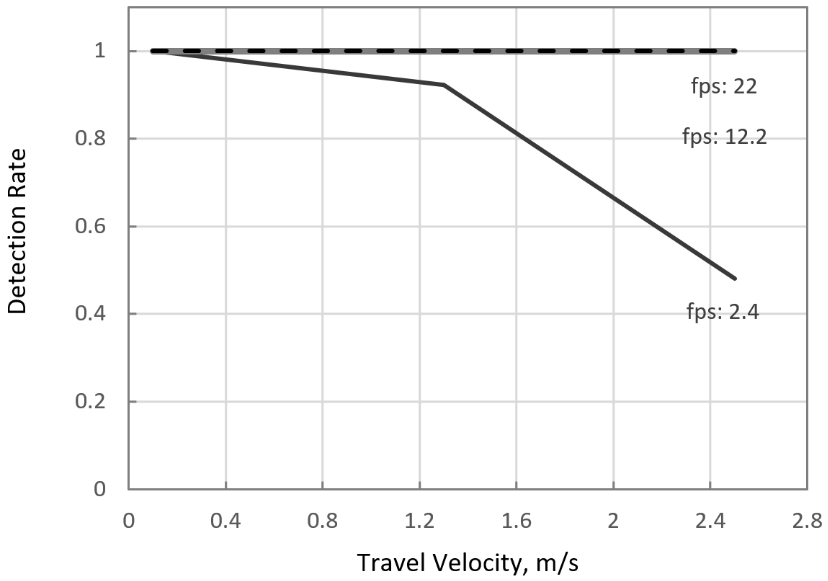


Figure 55

Simulated Plant Hill Detection Rates of the Vision Module Different Inference Speeds



These results infer that one-stage object detection models, such as YOLO and SSD, running on a discrete GPU such as 1070TI, have sufficient inference speed to avoid detection gaps in typical ranges of travel velocities for agricultural field operations. The result was also comparable to the 92% precision of the CNN-based MVS with 22 *fps* inference speed in the study of Partel et al. (2019) [79]. Therefore, these results infer that using a one-stage CNN model such as YOLOv3 on a laptop with at least Nvidia 1070TI can provide sufficient inference speed to avoid gaps in different field operations. However, the study did not report the travel velocity and field of view length of their setup. Thus, only an estimated performance comparison can be made.

The results also agree with other studies with known S, $\overrightarrow{v}_{travel}$, and *fps*. In the study of Chechliński et al. (2019) [78], their CNN-based-vision spraying system had $S = 0.55m$, $\overrightarrow{v}_{travel} = 1.11\frac{m}{s}$, and *fps* = 10.0. Applying these values to equation (2) also yields $r_o > 1$ (Case 2), which correctly predicted their results of full-field coverage. In the study of Esau et al. (2018) [134], their vision-based spraying system had $S = 0.28m$, $\overrightarrow{v}_{travel} = 1.77\frac{m}{s}$, and *fps* = 6.67 and also falls under Case 2. Similarly, the vision-based robotic fertilizer application in the study of Chattha et al. (2018) [29] had a $S = 0.31m$, $\overrightarrow{v}_{travel} = 0.89\frac{m}{s}$, and *fps* = 4.76. Again, calculating r_o yielded Case 2, which also agrees with their results.

At 2.4 *fps*, the simulated MVS failed to detect some plant hills when $\overrightarrow{v}_{travel}$ was 1.3 (Treatment 4) or 2.4 $\frac{m}{s}$ (Treatment 7). In contrast, missed detections were absent at $0.1\frac{m}{s}$ (Treatment 1). As mentioned in Section 5.2.5, treatments 1, 4, and 7 represent typical inference speeds of CNN models, such as YOLOv3 running in an embedded system, such as Nvidia TX2 [79]. From these results, it can be inferred that unless CNN object detection models were optimized, as illustrated in previous studies [78, 211], MVS with embedded systems shall only apply to agricultural field operations at walking speeds.

Figure 56 illustrates the detected hills per camera frame along the first 10-m traversed distance at 2.4 *fps* (treatments 1, 4, and 7). From Figure 56, three pieces of information can be obtained: (1) the absence of vertical gaps between consecutive frames; (2) horizontal overlaps among consecutive frames; and (3) the detection pattern. In Figure 56a, the absence of vertical gaps at $0.1\frac{m}{s}$ infers that all the hills were captured as the vision moved along the field length. The horizontal overlaps among consecutive frames also illustrate that a plant hill was captured by more than one processed frame. Finally, a detection pattern was repeated every 24 consecutive frames or approximately every 1-m length. The pattern length was the product of the total frames to complete a cycle and d_f.

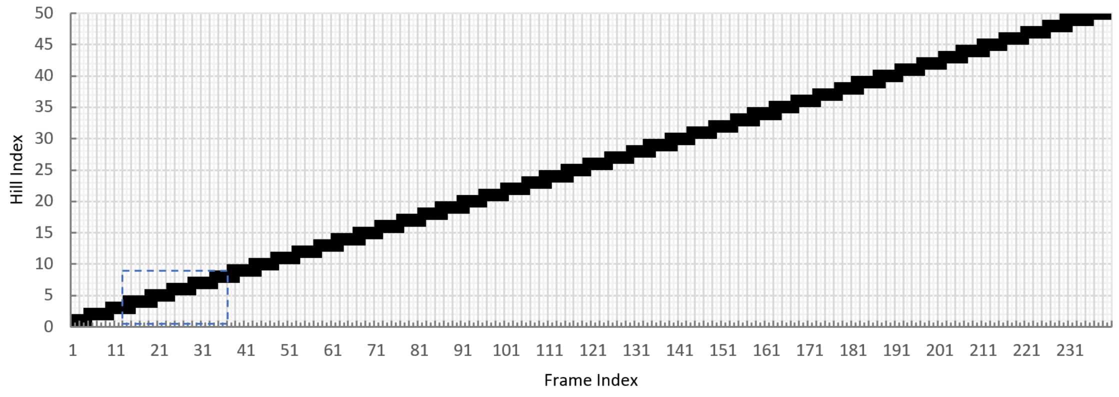
In contrast, the vertical gaps in some consecutive frames at 1.3 $\frac{m}{s}$, shown in Figure 56b, illustrated the missed detections. Horizontal overlaps were also absent. Hence, the detected plant hills were only represented in the frame once. The vision module traveled too fast and processed the captured frame too slowly at the set capture width, as demonstrated by the detection pattern of one missed plant hill every seven consecutive frames or approximately every 3.8-m traversed distance.

Similar results were also observed at 2.5 $\frac{m}{s}$ travel velocity, as shown in Figure 56c. However, due to faster travel speed, the vertical gaps were more extensive than Figure 56b. Observing the detection pattern showed that 14 plant hills were being undetected by the vision system every five frames or approximately every 5.21-m traversed distance. This pattern that forms every 5.21 m further explains the difference in the $r_{d,th}$ and $r_{d,sim}$ in the sensitivity analysis when the traversed distance was only 1 meter. A complete detection pattern was already formed when the distance was more than 10 meters, resulting in better detection rate estimates.

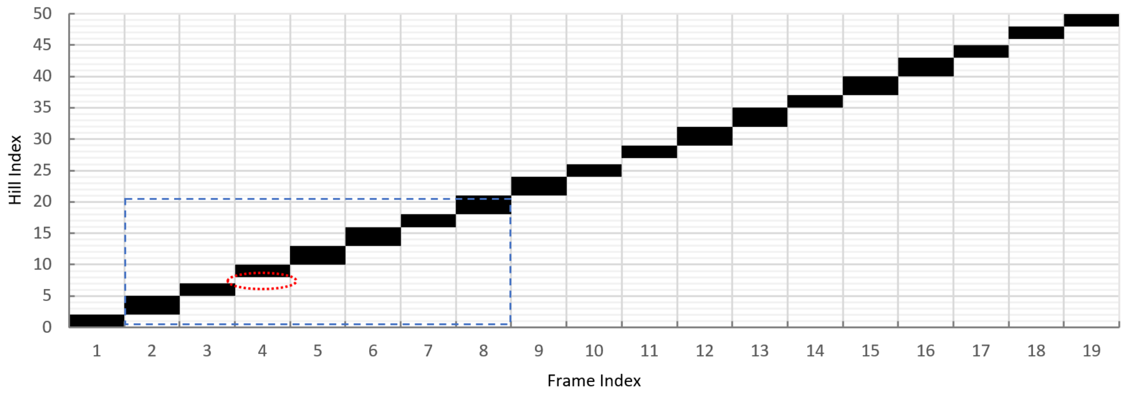
From these results, two vital insights can be drawn. First, at $r_o < 1$, $r_{d,th}$ shall have a margin of error when the length of the detection pattern is less than the traversed distance. Second, object tracking algorithms, such as Euclidean-distance-based tracking [212] that require objects to be present in at least two frames, would not apply when $r_o \leq 1$. Hence, the importance of $r_o > 1$ in MVS designs is further emphasized.

Figure 56

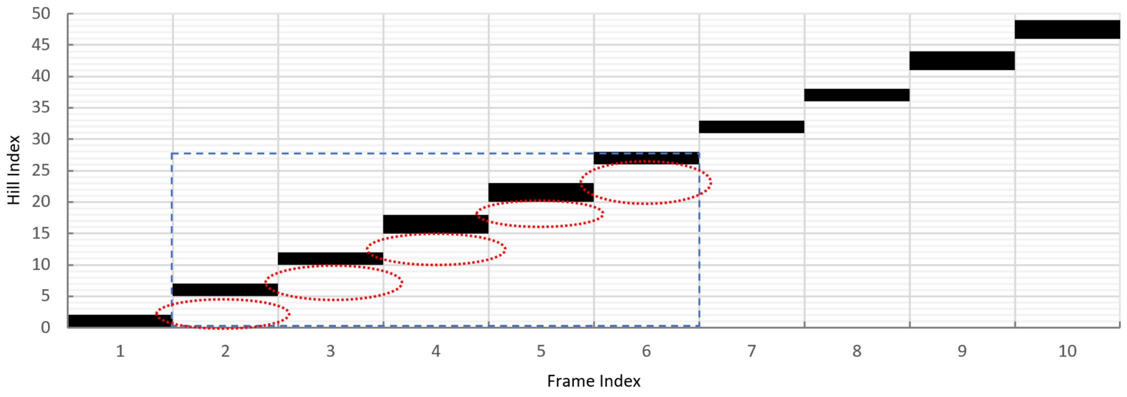
Detected and Undetected (Broken Red Lines) Plant Hills on Each Frame and Detection Pattern (Blue Broken Lines)



(a) 0.1 $\frac{m}{s}$



(b) 1.3 $\frac{m}{s}$



(c) 2.5 $\frac{m}{s}$

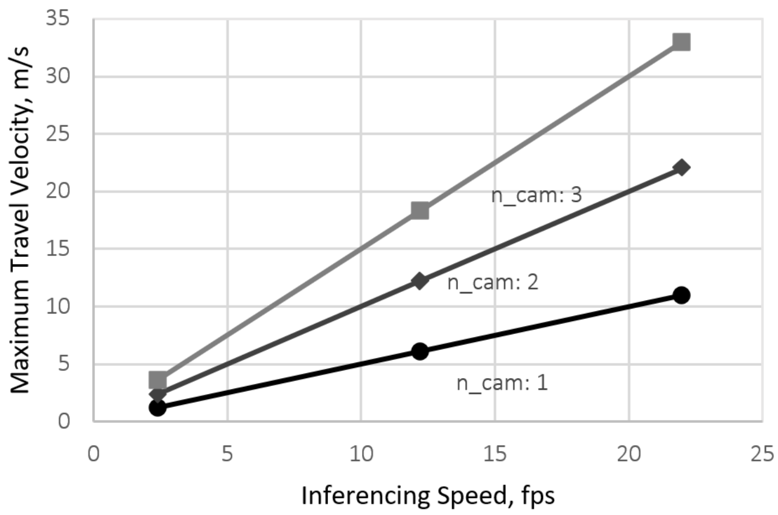
5.3.3 Effect of Increasing S or Multiple Cameras

In cases where $r_o < 1$ (Case 3), a practical solution to increase $\overrightarrow{v}_{travel,max}$ is to raise the camera mounting height, which in effect, shall increase S. However, if raising the camera mounting height is inappropriate, as doing so shall also decrease object details, using multiple cameras can be a viable solution.

Figure 57 illustrates the effect of increasing the effective *S* or using multiple cameras on the calculated values of $\overrightarrow{v}_{travel,max}$ for the three levels of inference speeds (2.4, 12.2, and 22 *fps*) simulated at $S = 0.5m$. The results showed that treatments with missed detections exceeded the allowable $\overrightarrow{v}_{travel,max}$. For treatments 4 and 7, the permissible travel velocity was only $1.2\frac{m}{s}$ using a single camera module, which was less than the simulated $\overrightarrow{v}_{travel}$ of 1.3 and $2.5\frac{m}{s}$, respectively.

Figure 57

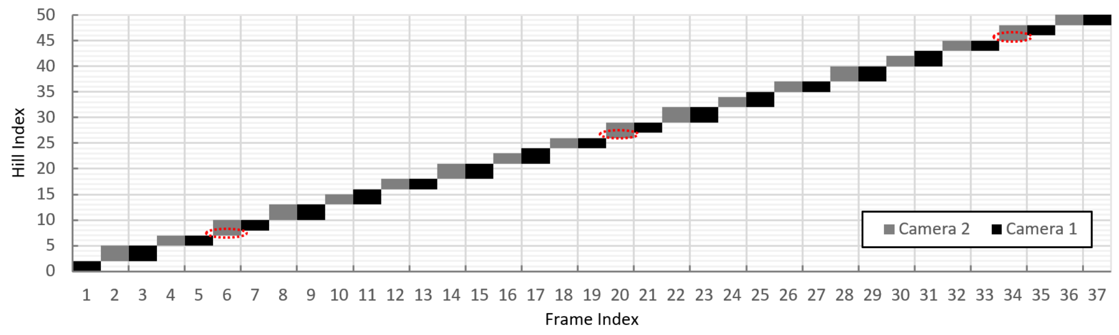
Theoretical Maximum Travel Velocity to Prevent Missed Detections at Different Number of Cameras and Inference Speeds



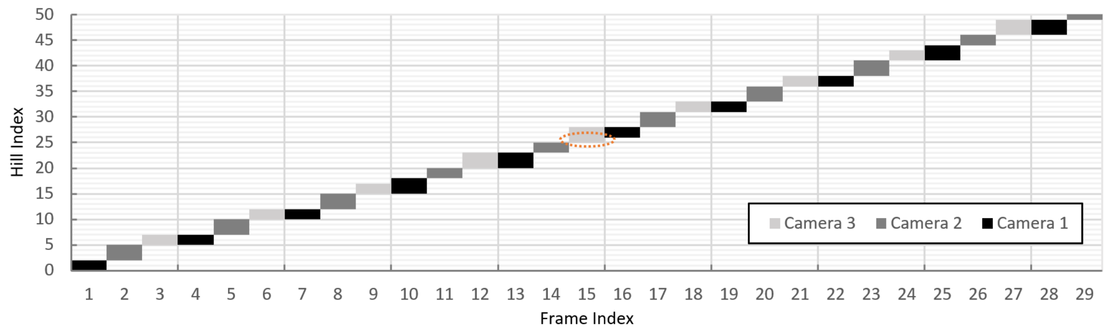
Calculating n_{vis} using Equation 14 for treatments 4 and 7 showed that 2 and 3 vision modules, respectively, were required to prevent missed detections. Thus, using two vision modules for treatment 4 prevented missed detections, as shown in Figure 58. The 6th, 20th, and 34th frames captured by the second camera detected the plants undetected by the first camera.

Figure 58

Detected and Undetected (Broken Red Lines) Plant Hills on Each Frame Along the First 10 m at $fps = 2.4$



(a) 1.3 $\frac{m}{s}$ *at* n_{cam} = *2*



(b) 2.5 $\frac{m}{s}$ *at* n_{cam} = *3*

As predicted, a two-vision-module configuration for treatment 7 was insufficient in preventing missed detections since the simulated $\vec{v}_{travel}$ of 2.5 $\frac{m}{s}$ of the vision system was still higher than the increased $\vec{v}_{travel,max}$. As illustrated in Figure 58a, the two-camera configuration would still result in an undetected hill on the 16th frame without a third camera.

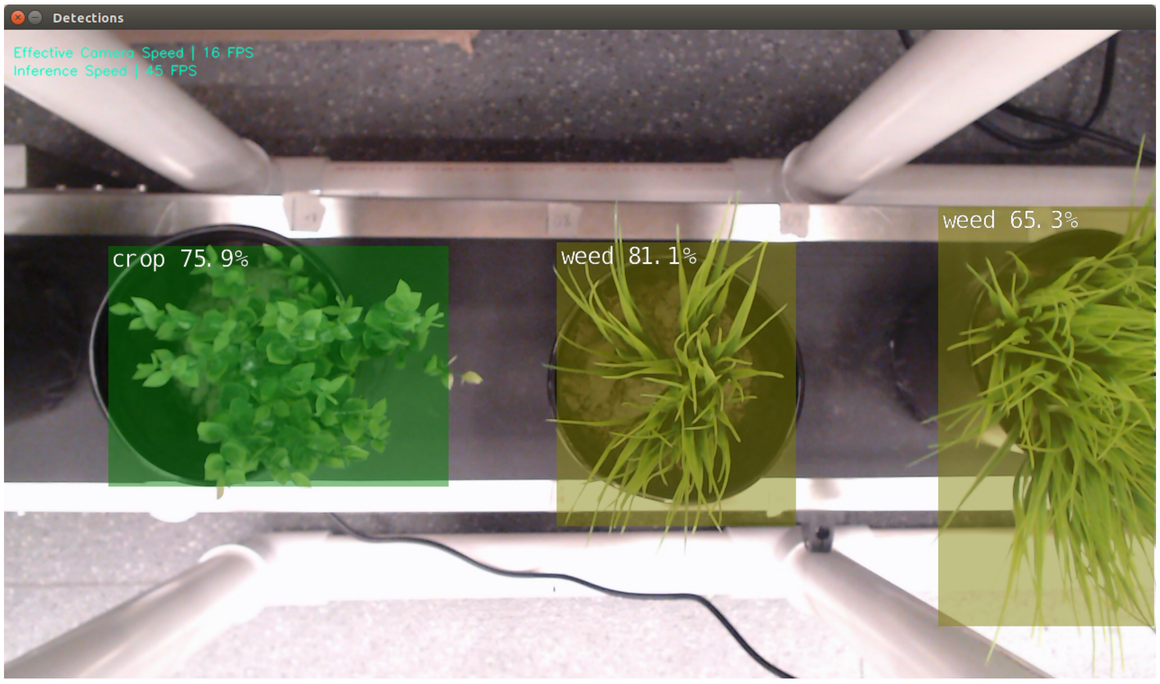
Based on these simulated results, the problem of missed detection due to the slow inference speed of embedded systems could be potentially solved by using multiple, adjacent, non-overlapping, and colinear cameras along the traversed row when raising the height of the camera is unwanted.

5.3.4 Vision Module Simulation and Testing Performance

Figure 59 shows the sample detection of the vision module. Results showed that using a TensorRT-optimized SSD MobileNetV1 to detect plants in 1280×720 images on an Nvidia Jetson Nano 4GB had an average inference speed of 45 *fps*. This average inference speed only represented the elapsed time to inference on an already loaded frame. However, due to calculation overheads caused by additional data processing and transmission, the average effective inference speed of the vision module was only 16 *fps*.

Figure 59

Sample Real-Time Inferencing Using Trained SSD MobileNetV1 Model and Optimized in TensorRT



The results using the theoretical approach and simulation for the vision module are shown in Table 25 Using equation (2), the configuration of the laboratory setup falls under Case 2 since $r_o > 1$. Then, using Equation 12, $r_{d,th}$ was calculated to be equal to 1.00. Applying Equation 13 yields $\overrightarrow{v}_{travel,max} = 12.64\ \frac{m}{s}$, which was highly sufficient for the target 0.3 $\frac{m}{s}$ and inferred that multiple vision modules were not required to prevent missed detections. Theoretical prediction of the performance of the vision module showed that the configuration was sufficient to prevent missed detection. Likewise, the theoretical result was confirmed by the simulation results that showed no missed detections ($r_{d,sim}$ = 1.00) for both crops and weeds among the simulated $\overrightarrow{v}_{travel}$.

Table 25

Theoretical and Simulation Performance of the Developed Vision Module Different Test Velocities

$\overrightarrow{v}_{travel}, \frac{m}{s}$	$d_f, \frac{m}{frame}$	r_o	**Case**	r_g	$r_{d,th}$	$r_{d,sim}$
0.1	0.0063	126.40	2	0.00	1.00	1.00
0.2	0.0125	63.20	2	0.00	1.00	1.00
0.3	0.0188	42.13	2	0.00	1.00	1.00

Table 26 summarizes the precision and recall of the trained CNN model in detecting potted plants at different relative travel velocities of the conveyor. Results showed that the combination of an optimized SSD-MB1-TRT running in a Jetson Nano 4GB has robust detection performance, and incorrect or missed detections were absent despite increasing travel velocity. Furthermore, the detection rates were equal when comparing the value of r_d to $r_{d,th}$ and $r_{d,sim}$. The recall was used for comparison instead of precision since the former is the ratio of the correctly detected plants to the total sample plants. This definition of r_d in Equation 2 is equivalent to the definition of $r_{d,sim}$ in Equation 22. Since the r_d,

$r_{d,th}$ and $r_{d,sim}$ were equal, these results proved the validity of the theoretical concepts and simulation methods presented in this study. Hence, $r_{d,th}$ and $r_{d,sim}$ can be used to theoretically determine the detection rate of a vision system in capturing plant images as a function of $\vec{v}_{travel}$ and *fps* with known *S*.

Table 26

The Detection Performance of the CNN-Based Vision Module

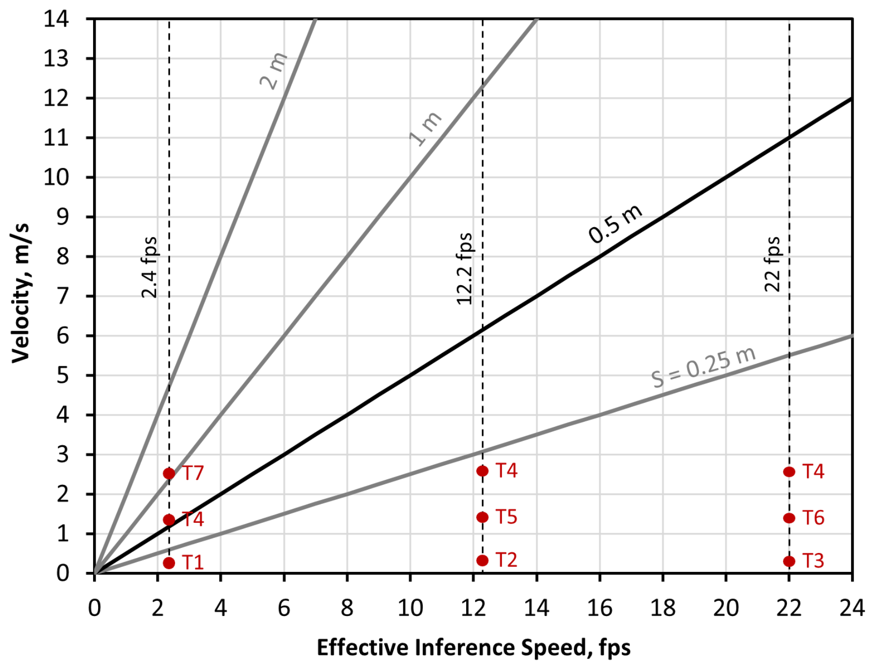
$\vec{v}_{travel}, \frac{m}{s}$	***TP***	***FP***	***FN***	p_d	r_d
0.1	60	0	0	1.00	1.00
0.2	60	0	0	1.00	1.00
0.3	60	0	0	1.00	1.00

5.3.5 Proposed Reference Chart

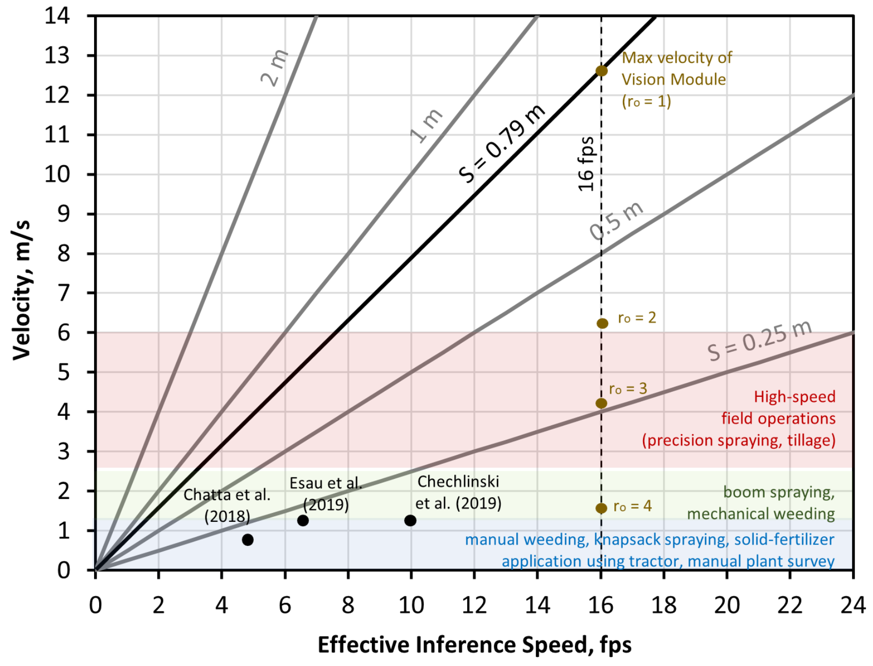
Figure 60 illustrates the proposed reference chart in designing an MVS-CNN for plant detection based on the results of simulations and actual systems. The horizontal and vertical axes represent textitfps of the MVS-CNN and $\vec{v}_{travel}$, respectively. On the other hand, the diagonal lines represent the *S*. Figure 60a shows the plot of the treatments in the simulation run. Identical to the simulation results, the plot indicates that treatments above the *S* = 0.5 m diagonal lines for the selected test *fps* had missed detections.

Figure 60

Proposed Reference Chart in Designing an MVS-CNN for Plant Detection



(a) *Test Simulations*



(b) *Actual Systems*

On the other hand, Figure 60b shows the plot of existing systems and the ranges of $\overrightarrow{v}_{travel}$ depending on the field operation. Based on Figure 60b, the developed vision module with S = 0.79 m and 16 *fps* has adequate performance for any field operation. Nonetheless, a certain degree of overlap is desired ($r_o > 1$) when implementing MVS-CNN for plant detection to increase the number of inferencing attempts. For example, the developed system could examine a scene approximately four times for a spraying operation.

Lastly, the graph could be a reference chart when selecting hardware and CNN architecture combinations for specific field operations. Using a reference $\overrightarrow{v}_{travel}$ of a desired field operation and probable S, the minimum *fps* of MVS-CNN can be determined. For example, for boom spraying at an average of 2 $\frac{m}{s}$ and desired field of view of S = 0.5 m, a minimum of 4 fps is required to prevent gaps between processed frames at r_o = 1. If a higher r_o is desired, the minimum effective inference speed is multiplied by r_o. For example, if an r_o of 2 is desired, the effective inference speed from the previous example is 8 fps.

In the next chapter, we extended the functionality of the developed MVS-CNN to include vision-based velocity estimation and plant targeting. Further, the developed MVS-CNN was connected to a sprayer module, and the overall system was tested for accuracy for real-time precision spraying.

Chapter 6

Development a CNN-Based Precision Sprayer with Vision-Based Velocity Estimation and Valve Control Using Variable Time Delay [1]

6.1 Introduction

Motion estimation plays an important role in the operation of precision agricultural equipment. Traditionally, RTK-GPS or wheel encoders are used to determine the position and velocity of precision sprayers. In this chapter, the plant bounding box detections of the MVS-CNN were tracked, and the Euclidean distances traveled by the plants in frame sequences were used to estimate the relative velocity of a precision sprayer. Traditionally, MVS applications in precision agriculture (PA) focus on object detection [49] and navigation guidance [214]. Very few studies used MVS for motion estimation of PA equipment. [105] estimated the distance traveled using a tractor-drawbar-mounted camera, and the travel distance was calculated from consecutive images using a k-nearest neighbor. Their vision-based system achieved lower errors ($\approx$ 3 mm) than wheel-encoder-based measurements ($\approx$ 7 mm) on soil tests.

Nonetheless, vision-based velocity estimation for precision spraying remains unexplored. We demonstrate that accurate spot spraying can be achieved using vision-based velocity estimation combined with VTD queuing and dynamic filtering.

Vision-based velocity estimation and plant targeting algorithms were added to the previously developed MVS-CNN in Chapter. By adding velocity estimation capability to the MVS-CNN, VTDs can be calculated without relying on auxiliary systems for motion estimation. Further, a sprayer module was connected to the MVS-CNN to utilize VTD queuing and dynamic filtering. The result was a reconfigurable design called a modular agrochemical precision sprayer (MAPS). Potentially, the proposed method can simplify the overall design and reduce the total system cost of vision-based precision sprayers.

[1] Some parts of this chapter were published in [213].

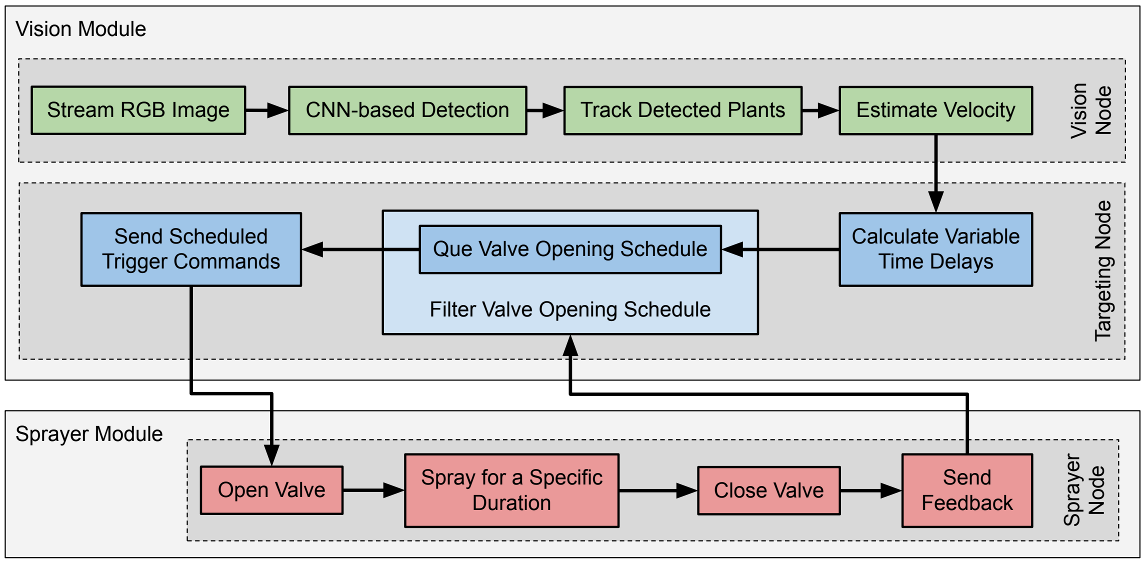
6.2 Materials and Methods

6.2.1 Scalable Unit

A sprayer module was integrated with the developed vision module from Chapter 5 to form a Scalable Unit (SU). The SU represents the fundamental working unit of the precision sprayer. The main functions of the SU were implemented as virtual nodes and communicated using Robot Operating System (ROS). An overview of this workflow is summarized in Figure 61.

Figure 61

Workflow of an SU of the MAPS

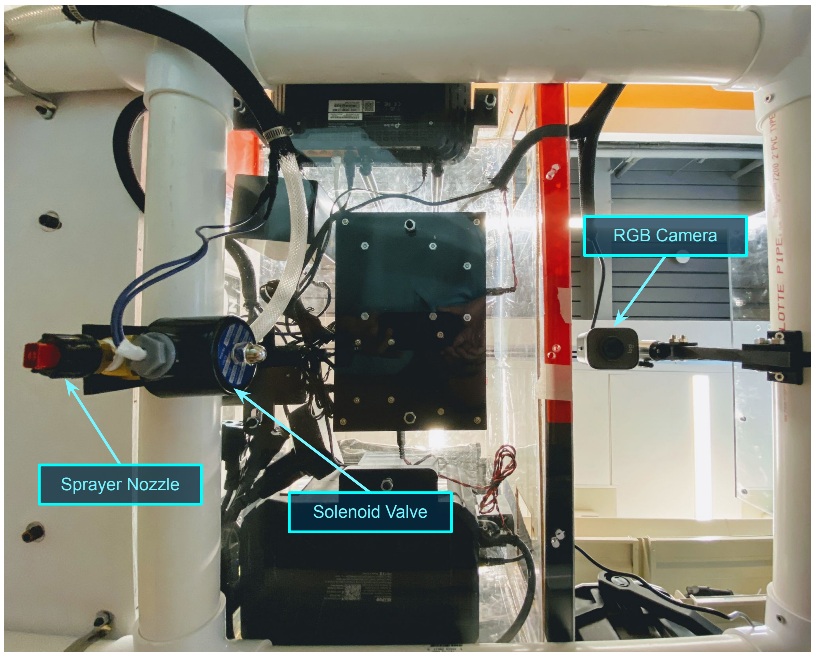


To start the process, an RGB camera (Logitech StreamCam) streams 1280 px × 720 px images at 30 fps to a vision computing unit (Nvidia Jetson Nano 4GB) via a universal serial bus (USB). The vision computing unit hosts two virtual nodes: (1) vision and (2) targeting nodes. Each node has specific functions. The vision node handles image streaming, CNN-based detection, tracking, and velocity estimation. On the other hand, the targeting algorithm calculates VTDs, issues valve trigger commands, and filters valve opening schedules. The details of each step executed by the vision and targeting nodes are discussed in the succeeding subsections.

An Arduino Nano microcontroller (ATmega328P) receives the valve trigger commands from the vision module via USB. It then sends a 5V signal to a relay (Arceli KY-019) that opens the 12VDC solenoid valve (US Solid USS2-00006). A fan-type sprayer nozzle (Solo 4900654-P) then sprays a target plant at a specified fixed spraying duration ($t_{spraying}$) and rate of 1.6 L/min. The nozzle was positioned such that it was horizontally aligned to the center of the camera frame (Figure 62). At the end of each spraying event, the microcontroller sends feedback to the Jetson Nano, which filters valve opening schedules based on the velocity, spraying duration, and filter size factor (FSF).

Figure 62

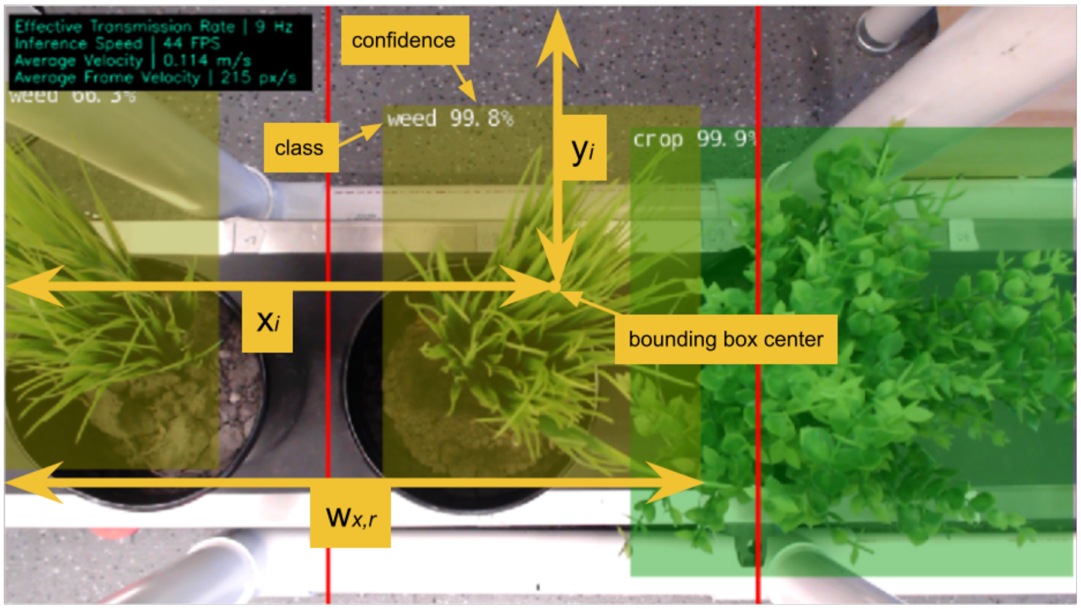
The Bottom View of the Sprayer Showing the SU Components



6.2.1.1 CNN-Based Detection. The vision module implements the same software architecture from Chapter 5 (Figure 51). The TensorRT-optimized SSD-MB1 (SSD-MB1-TRT) generates bounding boxes around detected objects. It also generates parameters such as the object class and frame coordinates in pixels (Figure 63). When the MVS-CNN detects a crop in the current frame, the center coordinates of the generated bounding boxes are stored in a list. On the other hand, when a weed is detected, the right-side ($w_{x,r}$) and center coordinates of the bounding box are recorded in another list. The targeting algorithm $w_{x,r}$ is the reference start position for the opening of the solenoid valve. On the other hand, the centers (x_i, y_i) were used in the tracking algorithm. All coordinates were in pixels, and the top-left corner of the frame served as the frame origin.

Figure 63

Sample Weed and Crop Detections Using SSD-MB1



6.2.1.2 Tracking Algorithm. The vision node implemented Euclidean-distance-based tracking for each plant class. Thus, plants and weeds can be counted separately. Figure 64 super-imposes two consecutive frames, which were labeled for reference. Suppose there were a total of I and J detected plants of a class in the current and previous frames, respectively, the Euclidean distances ($d_{e|i,j}$), in px, between the centers of bounding box detections in the current (x_i, y_i) and (x_j, y_j) previous frames, for each $i \in I$ to all $j \in J$, were calculated using equations Equation 23, Equation 24, and Equation 25.

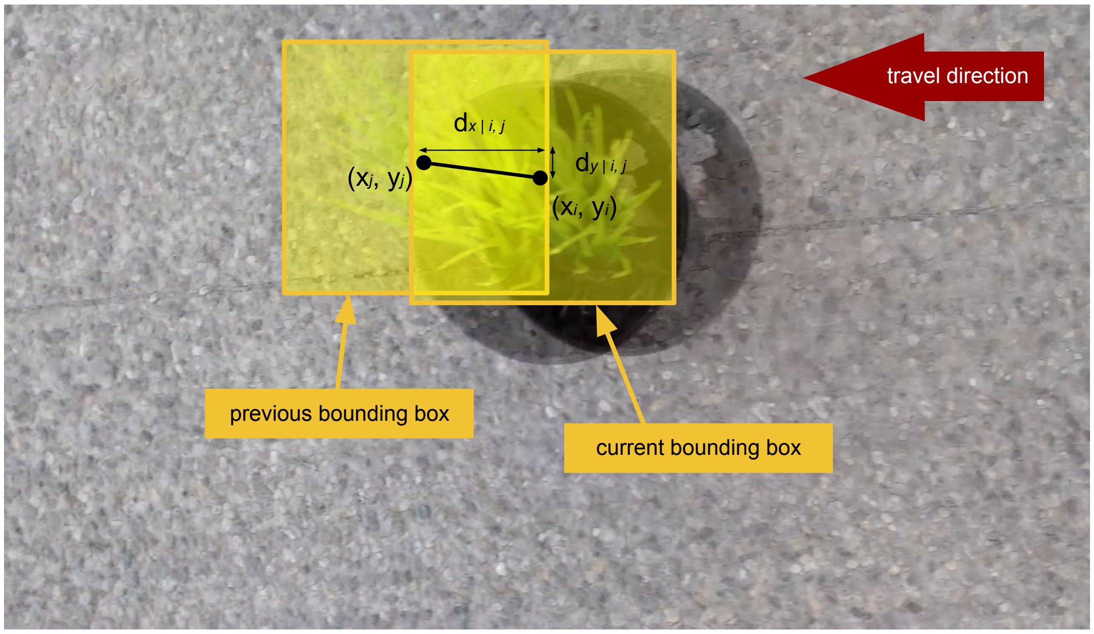
$$d_{x|i,j} = x_i - x_j \tag{23}$$

$$d_{y|i,j} = y_i - y_j \tag{24}$$

$$d_{e|i,j} = \sqrt{d_{x|i,j}^2 + d_{y|i,j}^2} \tag{25}$$

Figure 64

Distances Used in Tracking Plants



If $d_{e|i,j} \leq d_{e|thresh}$, a pre-determined tracking threshold, then we could determine that (x_i, y_i) and (x_j, y_j) represented the same object in the current and previous frames.

Note that $d_{e|thresh}$ was tuned with respect to frame rate and operating velocity. The full tracking algorithm was implemented as follows:

1. Sort i from highest to lowest x-coordinate (x_i) to start at the right-most plant.

2. If the previous frame J is empty, then assign a unique tracking index to each $i \in I$.

3. Else, for each $i \in I$:

   (a) Calculate $d_{e|i,j}$ for the current i for all $j \in J$.

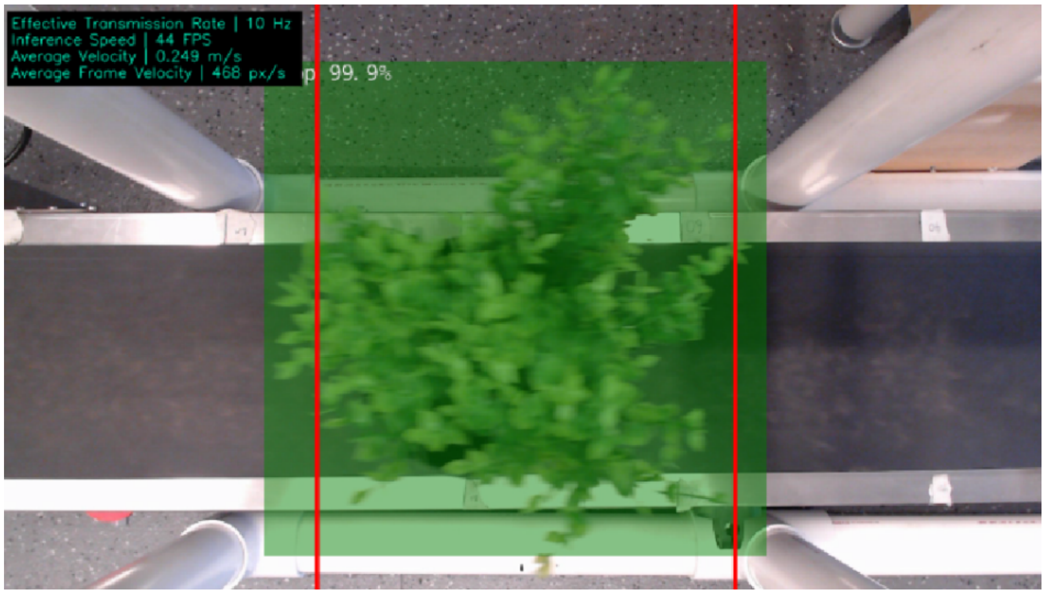
   (b) Assign tracking index of j with $\min\left(d_{e|i,j} \leq d_{e|thresh}\right)$ to i and delete $j \in J$. In detail, this step will transfer the indices of repeated objects from the previous frame to the current one by generating a tracking index lists $(x_j, y_j, Index_j)$ and $(x_i, y_i, Index_i)$. Then, if $\min\left(d_{e|i,j} \leq d_{e|thresh}\right)$, we let $Index_i = Index_j$.

   (c) Assign new tracking index to i if all calculated $d_{e|i,j} > d_{e|thresh}$ as they are new objects in the current frame.

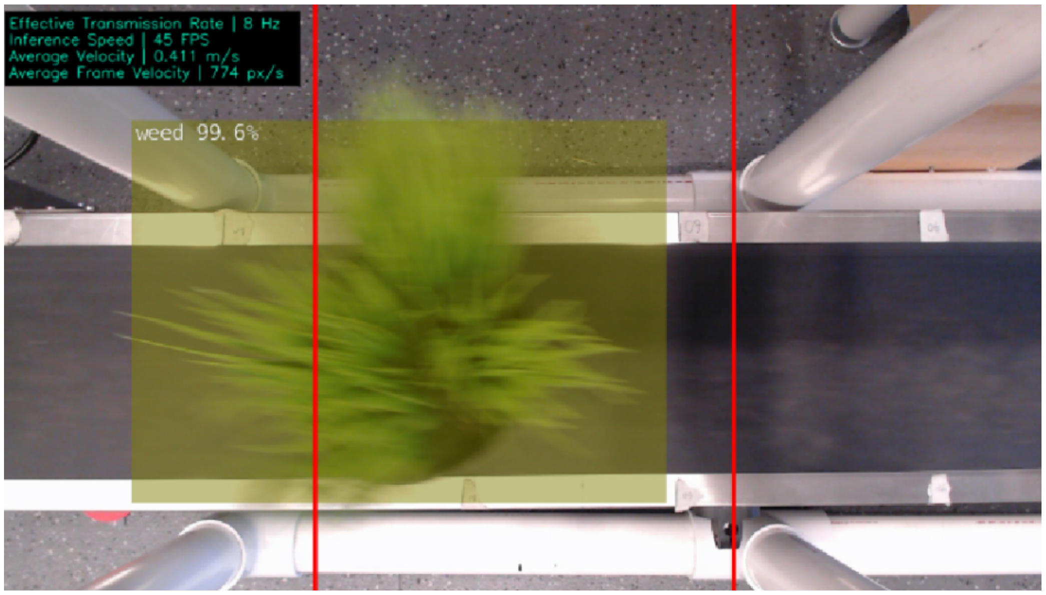
6.2.1.3 Velocity Estimation Algorithm. The vision module estimates velocity at the frame center and assigns buffer regions (s_{buffer}) at the left and right sides of the frame (Figure 65). Each buffer region was arbitrarily set to be 30% of the horizontal camera resolution (s_x) and parallel to the travel direction. This way, we can avoid partially formed bounding boxes when only a portion of the plant was within the camera frame. Further, we can also minimize the effect of lens distortions at the edge.

Figure 65

Sample Image of Crop and Weed with Detections, Velocity Measurements, and Extents of the Location and Velocity Estimation Region (Red Line)


Effective Transmission Rate | 10 Hz
Inference Speed | 44 FPS
Average Velocity | 0.249 m/s
Average Frame Velocity | 468 px/s
p 99. 9%

(a) *Crop*


Effective Transmission Rate | 8 Hz
Inference Speed | 45 FPS
Average Velocity | 0.411 m/s
Average Frame Velocity | 774 px/s
weed 99. 6%

(b) *Weed*

Knowing the horizontal frame displacement ($d_{x|i,j}$) from the tracking algorithm and the frame time difference of consecutive processed frames ($\Delta t_{vision|k}$), the frame travel velocity of each plant ($\overrightarrow{v}_{frame}$), in px/s, was estimated using the following equation:

$$\overrightarrow{v}_{frame} = \frac{d_{x|i,j}}{\Delta t_{vision|k}} = d_{x|i,j} \times fps_{effective} \tag{26}$$

Note that $\Delta t_{vision|k}$ is the total time needed to perform CNN-based inferencing, object tracking, and data transmission. k represents the frame index since the capture starts. The inverse of $\Delta t_{vision|k}$ equals $fps_{effective}$. The travel velocity ($\overrightarrow{v}_{travel}$), in m/s, was then estimated using the following equation:

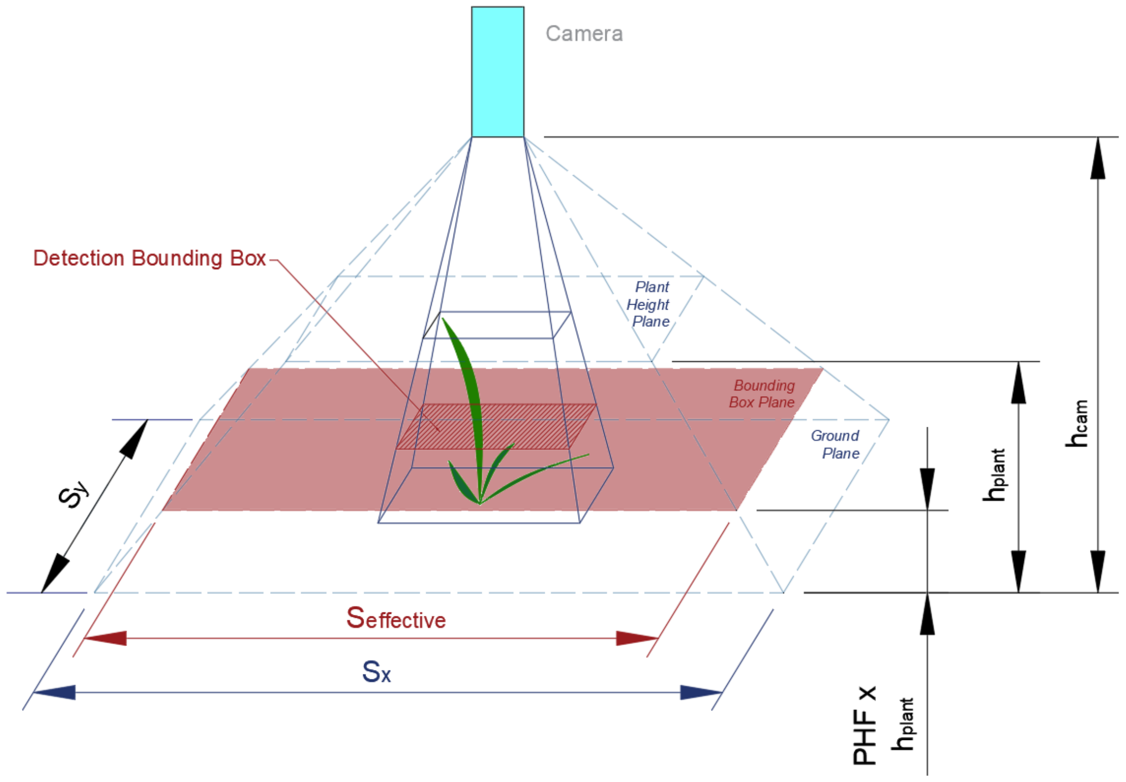
$$\overrightarrow{v}_{travel} = \overrightarrow{v}_{frame} \times LCR \tag{27}$$

The linear capture resolution (LCR), in m/px, is the conversion factor from pixel to meter depending on the capture device and mounting height. A moving average with three sample sizes was then applied to the $\overrightarrow{v}_{travel}$ for smoothing.

6.2.1.3.1 Estimating LCR. Preliminary tests showed that due to the height of the plant, the detection plane was situated at an offset distance from the ground. This offset distance was represented as a fraction of the height of the plant, denoted as plant height factor (PHF), as illustrated in Figure 66.

Figure 66

Distance Nomenclature Illustrating for LCR Calculation



PHF was then used to calculate an adjusted field of view ($S_{effective}$) along the travel direction. $S_{effective}$ is the size of the field of view, in meters, based on the location of the bounding box plane. Using similar triangles, shown in Equation 28, $S_{effective}$ was then calculated as a function of the camera mounting height from the ground (h_{cam}), in meters; plant height (h_{plant}), in meters; actual field of view (S_x) along the travel direction at ground level, in meters; and PHF.

$$S_{effective} = [h_{cam} - (h_{plant} \times PHF)] \times \frac{S_x}{h_{cam}} \tag{28}$$

Finally, LCR can be calculated using Equation 29. s_x was 1280 px in the setup since the long side of the image was parallel to the travel direction.

$$LCR = \frac{S_{effective}}{s_x} \tag{29}$$

6.2.1.3.2 Maximum Measurable Velocity. Since the decision for tracking an object requires $d_{e|i,j} \leq d_{e|thresh}$, the $d_{x|i,j}$ can only have a maximum value equal to $d_{e|thresh}$. This situation can occur when there is no y-component of the frame displacement, as illustrated in Figure 64, while the inference speed and travel velocity perfectly match [198]. Thus, substituting $d_{e|thresh}$ to $d_{x|i,j}$ to equations Equation 26 and Equation 27 yields the following equation:

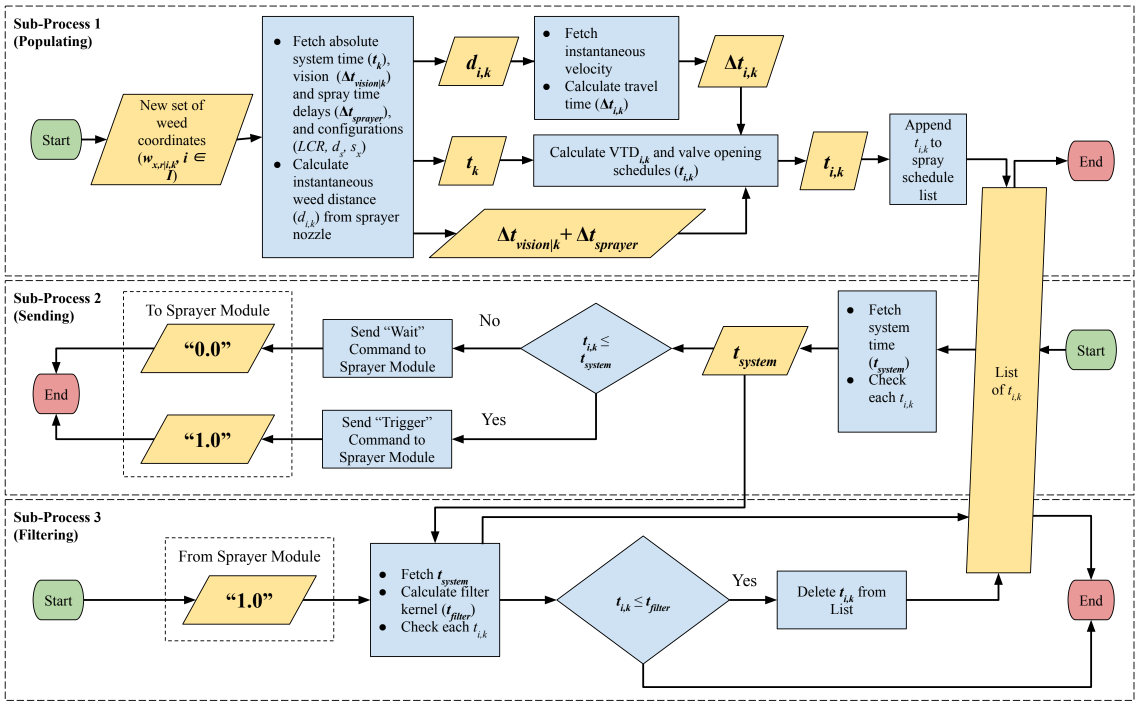
$$\vec{v}_{travel|max} = d_{e|thresh} \times LCR \times fps_{effective} \tag{30}$$

where $\vec{v}_{travel|max}$ is the theoretical maximum measurable travel velocity of the MVS, in m/s. However, imperfect camera mounting, vibrations, and fluctuations in the size of the bounding box may cause the actual $\vec{v}_{travel|max}$ be less than the theoretical value in Equation 30.

6.2.1.4 Targeting Algorithm. The targeting algorithm has three sub-processes that manage the list of valve opening schedules ($t_{i,k}$). The sub-processes include (1) populating the valve opening schedule list, (2) sending trigger commands to the sprayer module, and (3) filtering the list of valve opening schedules for elapsed entries. The detailed steps for each subprocess are shown in Figure 67.

Figure 67

The Workflow of the Targeting Algorithm of the MAPS Showing the Three Sub-Processes

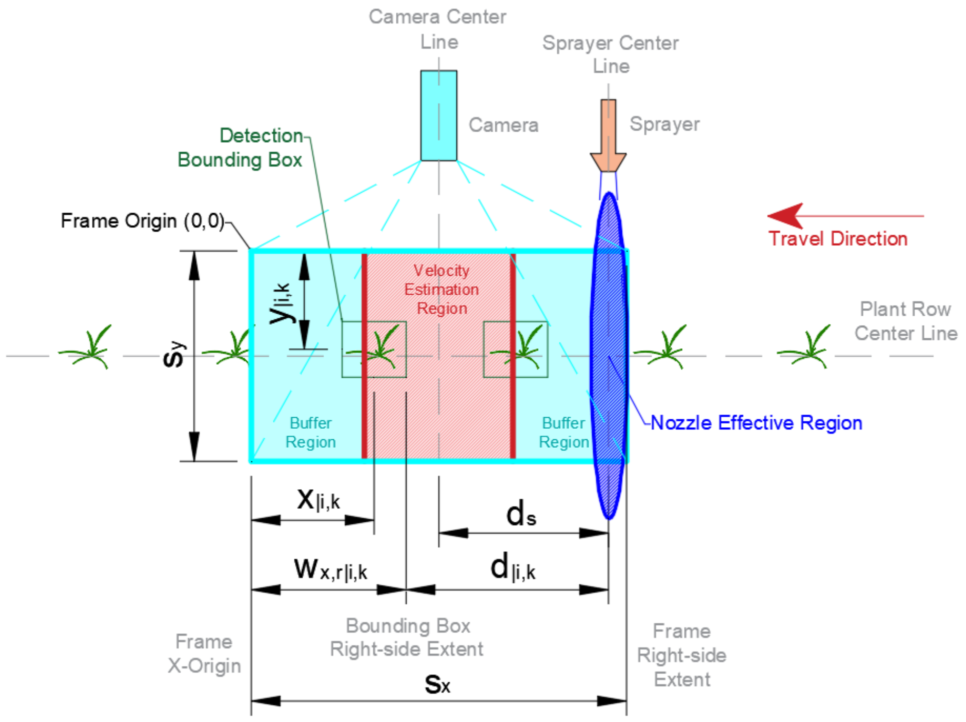

Sub-Process 1 (Populating)
Start
New set of weed coordinates ($w_{x,r|i,k}$, $i \in I$)
Fetch absolute system time (t_k), vision ($\Delta t_{vision|k}$) and spray time delays ($\Delta t_{sprayer}$), and configurations (LCR, d_s, s_x)
Calculate instantaneous weed distance ($d_{i,k}$) from sprayer nozzle
$d_{i,k}$
Fetch instantaneous velocity
Calculate travel time ($\Delta t_{i,k}$)
$\Delta t_{i,k}$
t_k
Calculate VTD$_{i,k}$ and valve opening schedules ($t_{i,k}$)
$t_{i,k}$
Append $t_{i,k}$ to spray schedule list
End
$\Delta t_{vision|k} + \Delta t_{sprayer}$
Sub-Process 2 (Sending)
To Sprayer Module
"0.0"
Send "Wait" Command to Sprayer Module
No
$t_{i,k} \le t_{system}$
t_{system}
Fetch system time (t_{system})
Check each $t_{i,k}$
List of $t_{i,k}$
Start
End
"1.0"
Send "Trigger" Command to Sprayer Module
Yes
Sub-Process 3 (Filtering)
Start
From Sprayer Module
"1.0"
Fetch t_{system}
Calculate filter kernel (t_{filter})
Check each $t_{i,k}$
$t_{i,k} \le t_{filter}$
Yes
Delete $t_{i,k}$ from List
End

6.2.1.4.1 Sub-Process 1: Populating. When the vision node publishes a new set of weed coordinates ($w_{x,r|i,k}$), the right-side coordinate of the bounding boxes of $i \in I$ number of detected weeds, the instantaneous distances from the sprayer ($d_{i,k}$), in meters, of each i detected weed ($i \in I$) at frame k were calculated using Equation 31, as illustrated in Figure 68. Note that the spray nozzle was mounted 0.45 m from the ground and had an elliptical spray pattern 0.15 m and 1.08 m. The shorter axis was parallel to the direction of travel. During this instance, the targeting algorithm stored the absolute time (t_k) when the weed coordinates were received. The distance (d_s) between the nozzle and sprayer was 0.41 m.

$$d_{i,k} = d_s - [(w_{x,r|i,k} - \frac{s_x}{2}) \times LCR] \tag{31}$$

Figure 68

Distance Nomenclature of Targeting Algorithm Showing the i^{th} Detected Plant at the k^{th} Captured Frame



The targeting node then used the most recent published values of $\overrightarrow{v}_{travel}$ to calculate the required travel time ($\Delta t_{i,k}$), in seconds, for the target to reach the effective spray region of the nozzle (Equation 32).

$$\Delta t_{i,k} = \frac{d_{i,k}}{\overrightarrow{v}_{travel}} \tag{32}$$

The valve trigger delay (VTD) for each i detected weed at the current k frame was then calculated using Equation 33. Image processing and transmission time ($\Delta t_{vision|k}$) was then subtracted from $\Delta t_{i,k}$ to represent the actual instance of image capture. Similarly, the sprayer latency time $\Delta t_{sprayer}$ was also deducted from $\Delta t_{i,k}$ to compensate for the unavoidable time delay by the sprayer mechanism. Preliminary tests recorded 9.50 $\pm$2.62 ms latency for the Jetson Nano to send a triggering command to the Arduino Nano. The response time of the solenoid valve was approximately 95 ms based on preliminary tests. Hence a 105ms total spraying latency ($\Delta t_{sprayer}$) was used. These calculated values of $VTD_{i,k}$ were then appended to the list of valve opening schedules at 1 ms resolution. Finally, the trigger schedule ($t_{i,k}$) of each detected weed at frame k was calculated in Equation 34.

$$VTD_{i,k} = \Delta t_{i,k} - (\Delta t_{vision|k} + \Delta t_{sprayer}) \tag{33}$$

$$t_{i,k} = t_k + VTD_{i,k} \tag{34}$$

6.2.1.4.2 Sub-Process 2: Sending. Since $t_{i,k}$ has 1 ms resolution, sub-process 2 monitors the valve opening schedule list at 100 Hz. If a time value in the spray schedule elapsed, i.e. a $t_{i,k}$ is less than the system time (t_{system}), the targeting algorithm publishes a value of 1.0 to the trigger command topic; otherwise, a value of 0.0 is published. We reserve the values between 0.0 and 1.0 for future improvements where the nozzle can be partially opened.

6.2.1.4.3 Sub-Process 3: Filtering. When feedback is received from the sprayer module, sub-process 3 checks each $t_{i,k}$ in the list, and any $t_{i,k}$ within the effective spray region were deleted to prevent unnecessary multiple sprays. This step is performed by comparing $t_{i,k}$ to a reference filter time (t_{filter}) that was calculated using Equation 35. t_{system} was the previous spraying reference time measured by sub-process 2. Any $t_{i,k} \leq t_{filter}$ are deleted.

$$t_{filter} = t_{system} + FSF * t_{spraying} \tag{35}$$

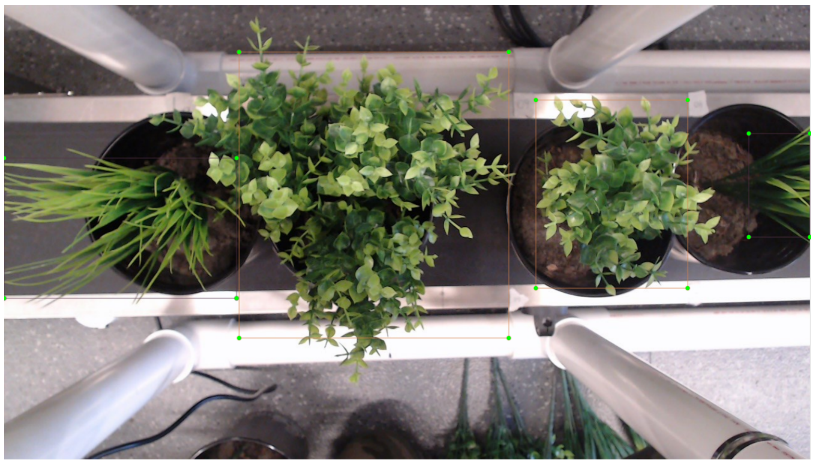
The filter size factor (FSF) dictates the intensity of filtering. It scales the spraying duration ($t_{spraying}$) and defines the allowable overlap between consecutive sprays.

6.2.2 CNN Model Development

1,643 sample images at 1280 $px \times 720$ px with 1,694 potted artificial crops and 1,967 potted artificial weeds were captured using Logitech StreamCam in laboratory and field settings. The images were annotated using LabelImg [182]. Figure 69 shows samples of annotated images. The dataset was then randomly stratified into 80% and 20% training and validation sets, respectively.

Figure 69

Sample Annotated Images of Artificial Weeds and Crops in (a) Laboratory and (b) Outdoor Settings



(a) *Indoor*

(b) *Outdoor*

The SSD-MB1 model was trained using NVidia Jetson Nano 4GB and Pytorch-SSD [195]. A batch size of 4, a base learning rate of 0.001, and a momentum of 0.90 were used to train the model. First, the training was performed for 115 epochs using a pre-trained model from a previous study of the authors [198]. Second, the trained model was converted to ONNX format using the conversion script in Pytorch-SSD [195] and then optimized using TensorRT [185].

6.2.3 Testing of Vision-Based Velocity Estimation

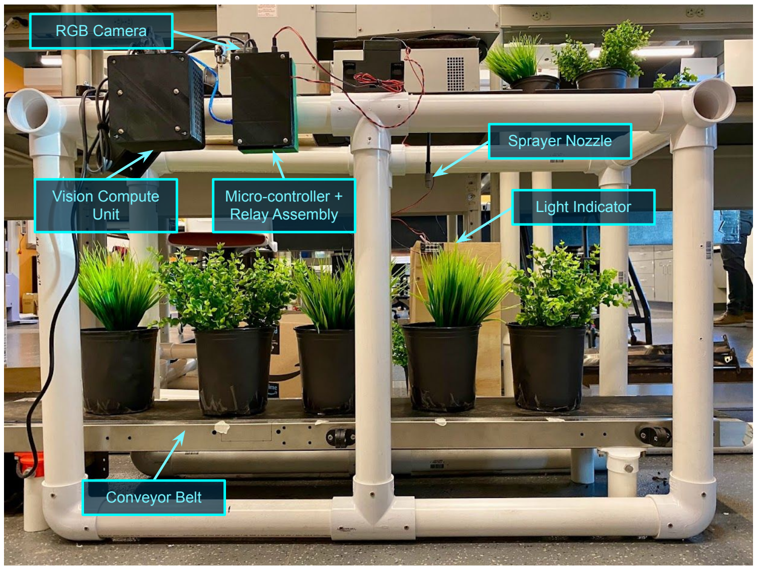
The tests were performed by comparing the estimated $\overrightarrow{v}_{travel}$ using the proposed vision-based approach against the actual measured travel velocity ($\overrightarrow{v}_{actual}$) of sample crop and weed. The mean absolute error or MAE (Equation 36) was used for evaluating the accuracy of measurements. The significant differences among MAEs were compared using ANOVA and Tukey's HSD at 95% confidence level test (N = 30 for each treatment).

$$MeanAbsoluteError = \frac{1}{N}\sum_{n=1}^{N} |\overrightarrow{v}_{travel|n} - \overrightarrow{v}_{actual|n}| \tag{36}$$

The test was implemented using a laboratory setup in Figure 70. The setup was composed of a single SU attached to a fixed frame and had a vision module that transmitted processed images at 1280 $px \times 720$ px and 9 fps. A potted artificial crop and weed with 0.33- and 0.34-m heights, respectively, were used as test samples. The average height of the two samples was used.

Figure 70

Laboratory Setup for Velocity Calibration and Preliminary Spraying Performance Testing of the SU



A $d_{e|thresh}$ equal to 100 px was initially used to determine the effect of different PHFs (0.1 to 1.0 at 0.1 intervals) on the accuracy of the proposed approach. Then, the system was tested at different tracking thresholds (50, 100, and 150 px) using the optimum PHF to determine the effect of $d_{e|thresh}$ on the accuracy of the system. The tests were performed at ten-speed settings (10% to 100% at 10% intervals) of the electric conveyor motor and three trials. The $\overrightarrow{v}_{actual}$ at the different conveyor motor power settings ranging from 0.04 to 0.53 m/s (0.14 to 1.91 $\frac{km}{h}$). The MVS was tested up to $d_{e|thresh}$ of 150 px in the laboratory since a significant change in the measured error was no longer observed despite increasing $d_{e|thresh}$.

6.2.4 Sprayer Performance Testing

This section outlines the performance parameters, hardware setup, and field experiments to optimize and characterize the performance of the sprayer. The metrics used are presented first and followed by the experiments conducted.

6.2.4.1 Performance Metrics.

A set of metrics, similar to the studies of [79] and [73], were used to characterize the performance. Table 27 summarizes the detection and spraying parameters observed during the field experiments.

Table 27

Parameters in the Performance Testing of the Precision Sprayer

Parameters		**Description**
Symbol	**Name**	
TP	True Positive	Number of weeds correctly identified as weeds by the vision system
FP	True Positive	Number of crops incorrectly identified as weeds by the vision system
TN	True Negative	Number of crops correctly identified as crops by the vision system
FN	False Negative	Number of weeds incorrectly identified as crops by the vision system
A_1	Single Spray, Full Coverage	Number of weeds completely sprayed with a single triggering of the solenoid valve
A_2	Multiple Spray, Full Coverage	Number of weeds completely sprayed with multiple triggering of the solenoid valve
B	Partial Coverage	Number of weeds partially sprayed with a single triggering of the solenoid valve
C	Miss Spray	Number of weeds missed sprayed due to early or delayed opening of the solenoid valve
D	Wrong Spray	Number of crops fully sprayed due to early opening or delayed closing of the solenoid valve
E	No Spray	Number of weeds not sprayed

The performance metrics are then summarized in Table 28. The application-level detection precision (p_d) measures the fraction of the positive detections correctly identified as weeds. On the other hand, the application-level detection recall (r_d) was the fraction of correctly detected weed samples over the total number of weed samples.

Table 28

Calculated Performance Metrics of the Precision Sprayer

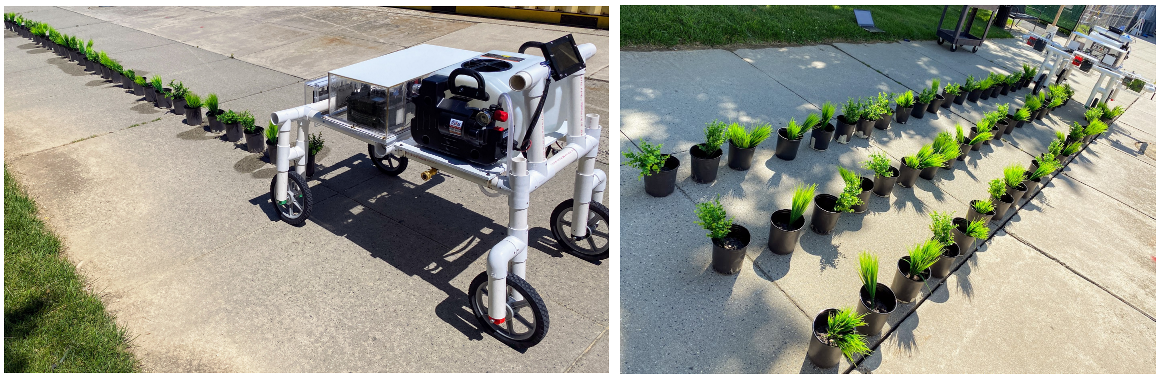
Symbol	**Description**	**Formula**
p_d	Detection Precision	$p_d = TP/(TP+FP)$
r_d	Detection Recall	$r_d = TP/(TP+FN)$
TT	Total Targets	$TT = A+B+C+E$
TS	Total Sprayed	$TS = A+B+C+D$
FS_1	Single Full Spray Rate	$FS_1 = A_1/TT$
FS_2	Multiple Full Spray Rate	$FS_2 = A_2/TT$
FS	Full Spray Rate	$FS = FS_1 + FS_2$
PS	Partial Spray Rate	$PS = B/TT$
MS	Miss Spray Rate	$MS = C/TT$
WS	Wrong Spray Rate	$WS = D/TS$
NS	No Spray Rate	$NS = E/TT$
p_s	Spraying Precision	$p_s = 1 - WS$
r_s	Spraying Recall	$r_s = 1 - (MS+NS)$

The total targets (TT) were the number of weed samples during the test. On the other hand, the total sprayed (TS) was the total number of crops and weeds that were sprayed. The rate of each spray type was then classified as full (FS), partial (PS), miss (MS), or no sprays (NS). On the other hand, the wrong spray (WS) described the fraction of the sprays which incorrectly targeted a crop. The rates of correct spray attempts that targeted the weeds were then expressed as the spraying precision (p_s). Finally, the spraying recall (r_s) was the fraction of the targets that were either fully or partially sprayed.

6.2.4.2 Field Experiments. Two experiments were performed to optimize and characterize the overall performance of the sprayer (Figure 71). All experiments were performed at walking speeds. The videos of each run showing the nozzle spray and the plants were recorded to evaluate and characterize the spraying instances (Figure 72). The vision modules were set to transmit processed images at a reduced size of 569 $px \times 320$ px, resulting in an average detection speed of 19 fps. Each camera was mounted at 0.71 m from the ground and had a field of view of 0.956 m (S_x) parallel to the travel direction. This setting resulted to a 0.6854-m $S_{effective}$ and 0.5354 $\times 10^{-3}$-m/px LCR.

Figure 71

The Two Field Experimental Setups



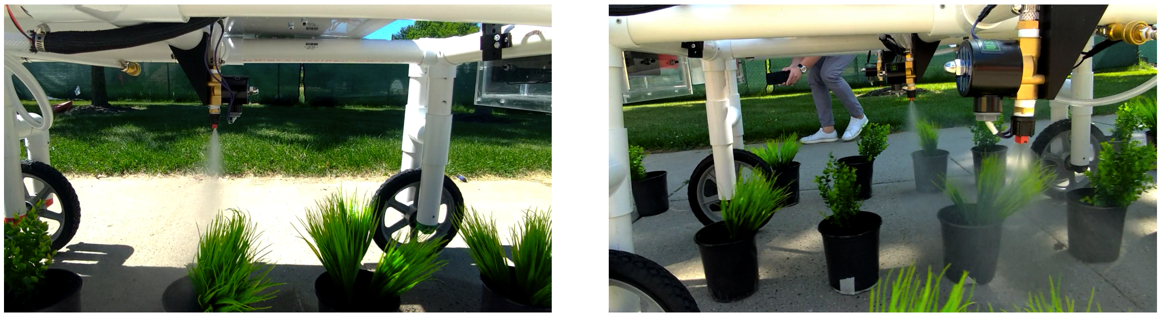
(a) *Field Experiment 1* **(b)** *Field Experiment 2*

The first experiment (Figure 71a) tested a single SU at two levels of $t_{i,k}$ filtering (low and high) and three levels of $t_{spraying}$ (200, 350, and 500 ms). A low level of $t_{i,k}$

filtering would only delete valve opening schedules that were within half spraying duration (FSF = 0.5). In contrast, a high level of $t_{i,k}$ filtering would delete all $t_{i,k}$ that were within the full spraying duration (FSF = 1.0). 30 potted crops and 30 weeds were randomly positioned along a 0.50-m wide plant row at 0.30-m hill spacing. For the second experiment (Figure 71b), the sprayer with three SUs was tested using the derived optimum FSF and $t_{spraying}$ from the first experiment. 10 potted crops and 10 weeds were randomly positioned along three plant rows. Each run was performed three times, and the average of the recorded parameters, as described in Table 27 were used to calculate detection and spraying performance.

Figure 72

Sample Recorded Spray Instances of the Precision Sprayer



(a) *Single Scalable Unit* **(b)** *Three Scalable Units*

6.3 Results and Discussion

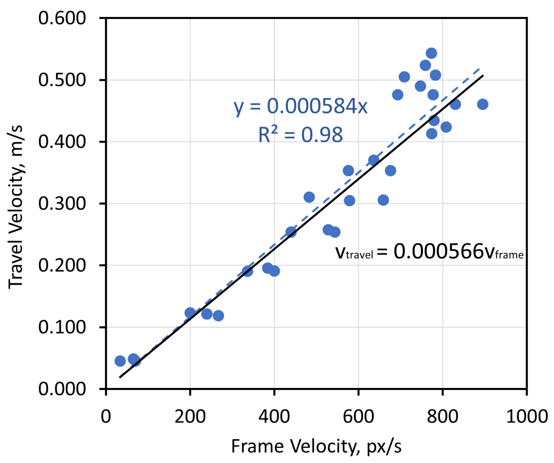
6.3.1 Velocity Measurement Calibration

Figure 73 summarizes the measured $\overrightarrow{v}_{actual}$ against measured $\overrightarrow{v}_{frame}$ of the MVS-CNN. The broken and solid lines represent the prediction using linear regression and proposed analytical models, respectively. The slope of the lines represents the calculated *LCR* of each model. In general, the data showed a strong linear relationship between $\overrightarrow{v}_{actual}$

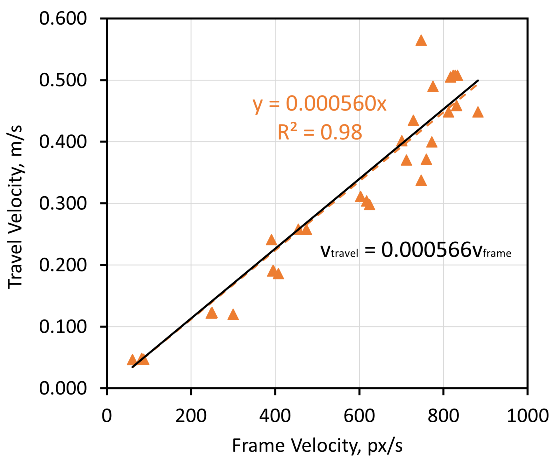
against measured $\overrightarrow{v}_{frame}$ ($R^2 > 0.98$).

Figure 73

The Measured Frame Velocity of the MVS-CNN Against Different Ranges of Test Velocities



(a) *Crop*

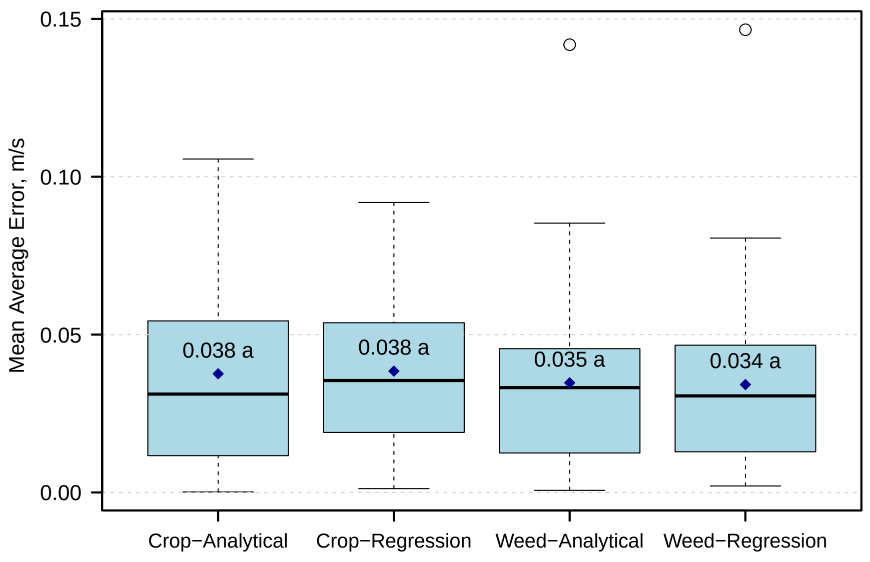


(b) *Weed*

The test results also showed variations of $\overrightarrow{v}_{frame}$ measurements within test velocities, which can be attributed to the random size fluctuations of the bounding box generated by the CNN among consecutive frames. Comparing the converted values of $\overrightarrow{v}_{frame}$ (px/s) to $\overrightarrow{v}_{travel}$ (m/s) showed that the analytical approach tends to have more variation in the mean average errors of the velocity estimates than using the regression equation (Figure 74). However, in general, there are no significant difference in the mean average errors using ANOVA and Tukey's HSD test at 95% confidence level.

Figure 74

Comparison of the MAEs of Velocity Estimates Using Regression and Analytical Models

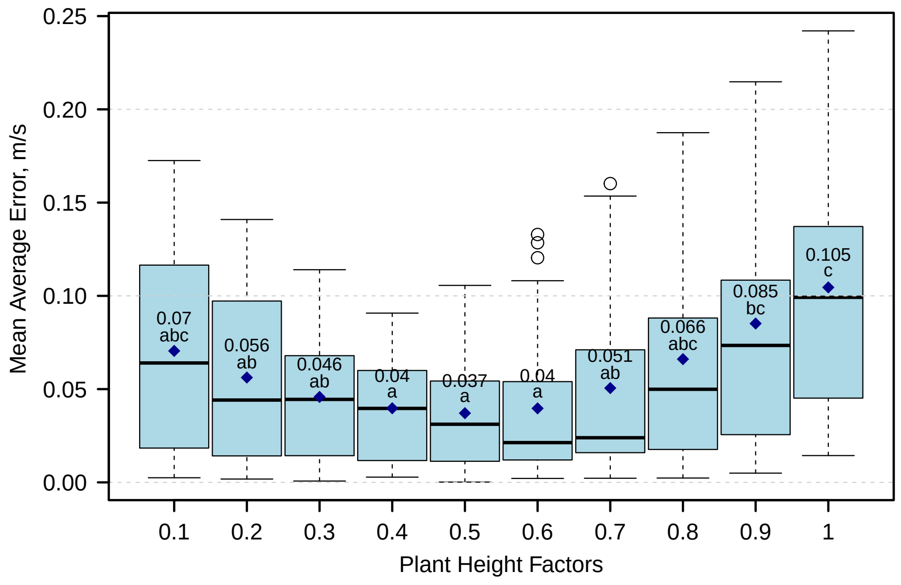


6.3.2 Effect of Plant Height Factor

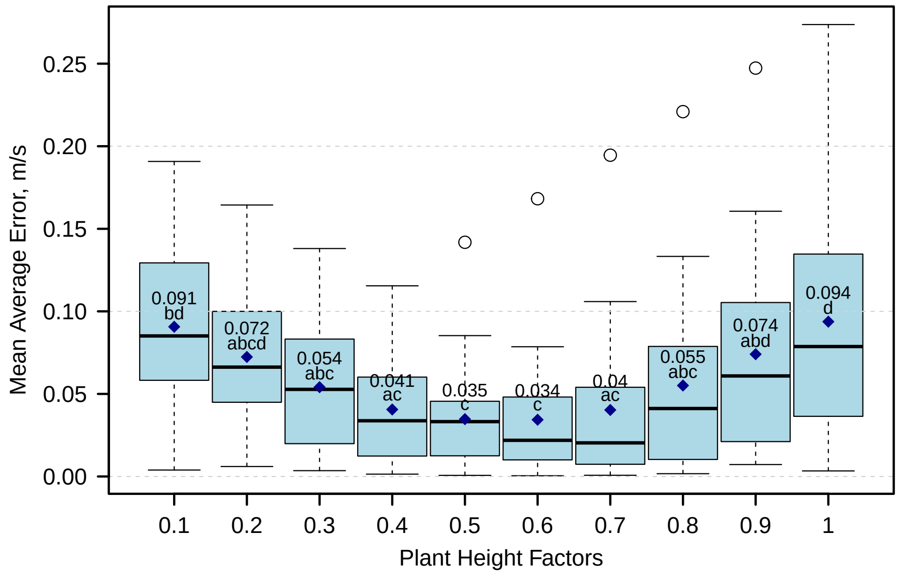
Figure 75 compares the mean absolute errors across the test velocities when the PHF of the analytical model was adjusted. The results showed that, on average, the minimum mean absolute error was attained at 0.6 PHF for the crop (Figure 75a) and 0.5 PHF for the weed (Figure 75b). Comparing the mean absolute errors showed no significant difference at 0.4 to 0.6 PHF (95% confidence level using ANOVA and Tukey's HSD test 30 samples for each treatment). The bounding box plane tends to be at about half the plant height. In general, as the assumed bounding box plane location moves away from half of the plant height, the means and variations of the absolute errors increase. These results infer that in future applications, measuring the height of reference plants and applying 0.5 PHF can yield good velocity estimates using the proposed analytical model. This way, the calibration method can be simplified by minimizing the amount of data that needs to be collected.

Figure 75

The MAEs of Velocity Estimates at Different PHF Using the Analytical Approach



(a) *Crop*



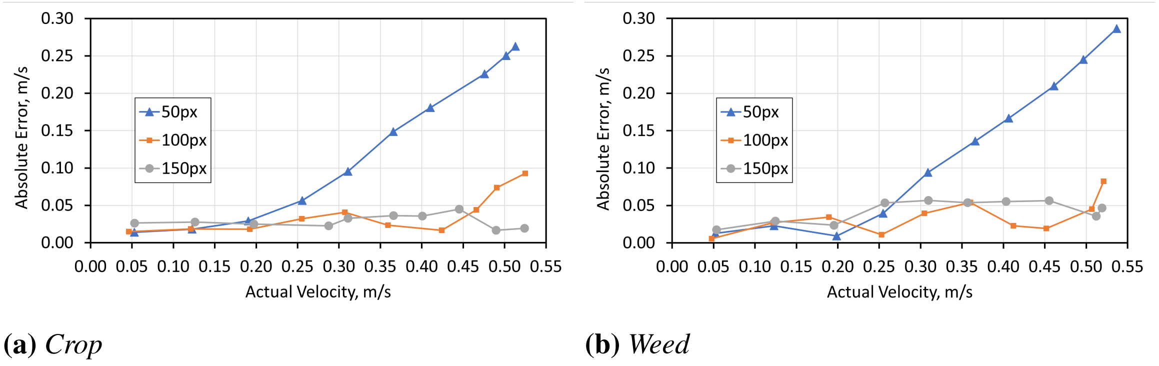
(b) *Weed*

6.3.3 Effect of Tracking Distance Threshold

Figure 76 details the absolute errors at 0.5 PHF and different tracking distance thresholds. Results showed a sudden rise in absolute errors when a certain velocity was reached at 50 and 100 px $d_{e|thresh}$. These results verify the behavior of the system as predicted by Equation 30. At a detection speed of $9fps$ and $d_{e|thresh}$ equal to 50 px, the theoretical $\vec{v}_{travel|max}$ was only 0.252 m/s. Thus, proper object tracking could not be established beyond $\vec{v}_{travel|max}$ during the test, resulting in inaccurate measurements. Increasing $d_{e|thresh}$ to 100 px increased $\vec{v}_{travel|max}$ to 0.504 m/s. Similarly, the errors started to increase at around the calculated maximum velocity. Finally, since the test velocities were lower than the theoretical $\vec{v}_{travel|max}$ of 0.756 m/s at the 150-px threshold, a sudden rise in absolute errors was not observed.

Figure 76

The MAEs Using Analytical Approach at Different Tracking Distance Thresholds



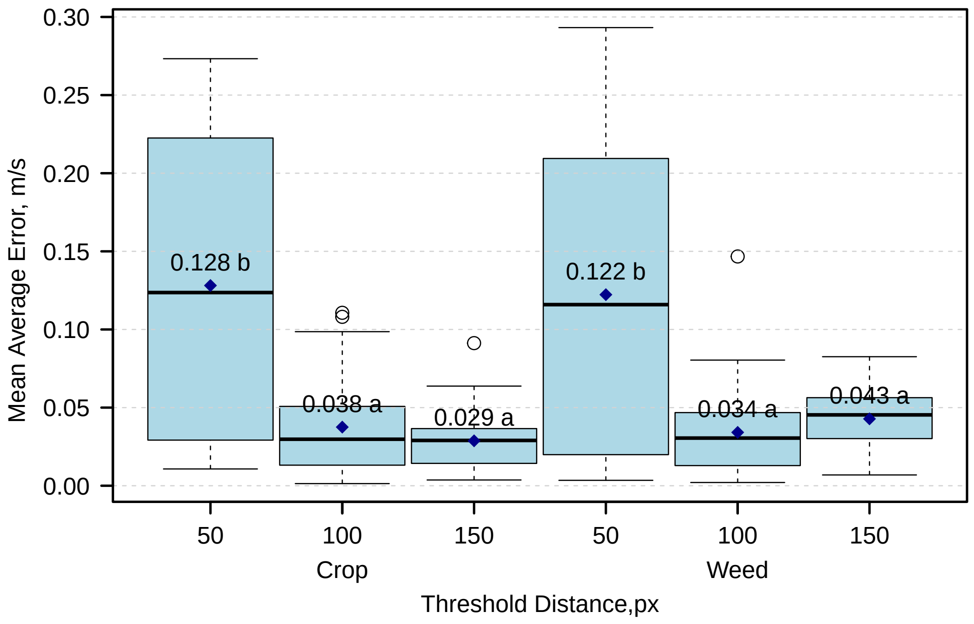
(a) *Crop* **(b)** *Weed*

Further, Figure 76 also illustrates that the mean average error tends to be lower at low than high test velocities for all tested values of $d_{e|thresh}$. This might have been caused by the blurring of the objects at higher velocities, as shown previously in the sample test image (Figure 65).

Finally, comparing the mean average errors using ANOVA and Tukey HSD showed that the errors using 50-px were significantly different than at 100 and 150 px thresholds (Figure 77). On average, the proposed vision-based approach to estimate the velocity had a mean absolute error of 0.036 m/s in measurements within the theoretical $\overrightarrow{v}_{travel|max}$. Note that the theoretical $\overrightarrow{v}_{travel|max}$ can still be increased by using a larger $d_{e|thresh}$ and faster $fps_{effective}$. However, using a $fps_{effective}$ is potentially better than a larger $d_{e|thresh}$. When the distance between reference plants is small, using a large $d_{e|thresh}$ can potentially result in incorrect tracking. Nonetheless, despite the mentioned limitations, the tests demonstrated the potential of the proposed method in measuring velocity using CNN bounding boxes.

Figure 77

Comparison of the MAEs from 0.04 to 0.53 m/s at Different Tracking Distance Thresholds

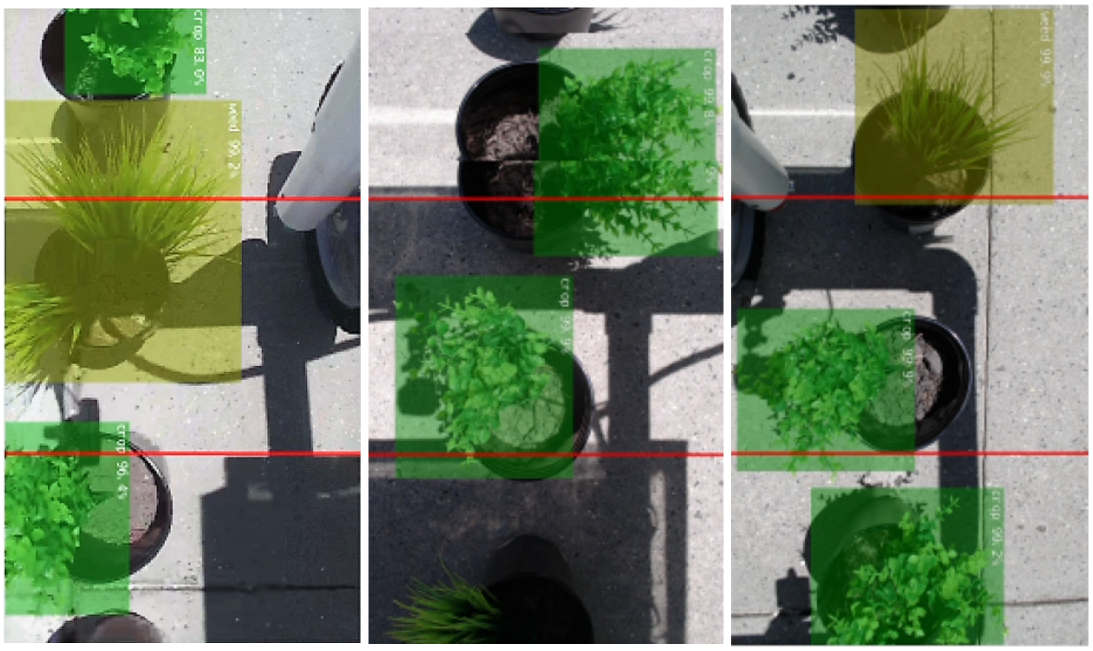


6.3.4 Spraying Performance

Figure 78 shows sample detections of each of the SU that were streamed at 320 px to the GUI of the central module. Despite being captured in motion, significant blurring in the images was not observed, and the CNN model had very high confidence in the detections.

Figure 78

Sample Detections of the (a) Left, (b) Middle, and (c) Right SUs of the MAPS



(a) *Left* **(b)** *Middle* **(c)** *Right*

The results showed that the system was very accurate in detecting both targets and non-targets, as summarized in Table 29. Incorrect ($P_d = 1.0$) or missed ($R_d = 1.0$) detections were absent at the tested walking speeds of 1.03 ± 0.16 m/s (3.71 ± 0.57 km/h) in all of the test runs. These results agree with our predictions that the inference speed (19 fps) of the MVS-CNN was sufficient to prevent gaps between process frames at the tested velocities and camera field of view [198].

Table 29

The Detection Performance of the MAPS at Different FSF (low and high), spraying duration (200, 350, and 500 ms), and $1.03\ \pm 0.16\ \frac{m}{s}$ *(*$3.71\ \pm 0.57\ \frac{km}{h}$*)*

Spray Schedule Filtering	**Spray Duration,** *ms*	TP	FP	TN	FN	p_d	r_d
Low	200	30	0	30	0	1.00	1.00
	350	30	0	30	0	1.00	1.00
	500	30	0	30	0	1.00	1.00
High	200	30	0	30	0	1.00	1.00
	350	30	0	30	0	1.00	1.00
	500	30	0	30	0	1.00	1.00

The results of targeting performance are then summarized in Table 30. The system was initially tested with a high level of spray schedule filtering (FSF = 1.0), which resulted in only 1% wrong spray at 200 ms spray duration. The rate of the wrong spray then increased to 4% (350 ms) and 13% (500 ms), caused by the longer area that was covered by each spray when the spray duration was increased. At the tested $\vec{v}_{travel}$, 200, 350, and 500 ms covered a distance of 0.21, 0.36, and 0.52 m, respectively. Since hill spacing was only 0.30 m, the 350 and 500 ms spraying duration resulted in the adjacent plants being either partially or fully sprayed. The minimum partial sprays were achieved at 500 ms (3%), which was expected as the long spray duration was sufficient to spray a plant fully.

Table 30

The Targeting Performance of the MAPS at Different FSF (low and high), spraying duration (200, 350, and 500 ms), and $1.03 \pm 0.16\ \frac{m}{s}$ *(*$3.71 \pm 0.57\ \frac{km}{h}$*)*

FSF	$t_{spraying}$, **ms**	TT	TS	FS_1	FS_2	FS	PS	MS	WS	NS	p_s	r_s
	200	30.00	32.33	0.92	0.07	0.99	0.01	0.00	0.07	0.00	0.93	1.00
0.5	350	30.00	43.00	1.00	0.00	1.00	0.00	0.00	0.30	0.00	0.70	1.00
	500	30.00	48.67	1.00	0.00	1.00	0.00	0.00	0.38	0.00	0.62	1.00
	200	30.00	29.33	0.91	0.01	0.92	0.04	0.00	0.01	0.03	0.99	0.97
1.0	350	30.00	29.00	0.81	0.00	0.81	0.08	0.00	0.04	0.11	0.96	0.89
	500	30.00	31.00	0.87	0.00	0.87	0.03	0.00	0.13	0.10	0.87	0.90

Nonetheless, the high spray schedule filtering also deleted time schedules of the adjacent weed that were within the previous spray duration, resulting in partial and no spray. The lowest no-spray rate was observed at 200 ms (3%), where a short spray duration did not cause the succeeding valve opening schedule to be deleted. Overall, the initial results fall within expectations. Longer spraying duration results in more full sprays than shorter duration. However, it also causes an unintentional spray of adjacent plants. Shorter spray duration results in a more precise spray than a longer duration. However, early nozzle opening with short spray duration can also result in partial or missed sprays.

The results using the full spraying duration to filter the valve opening schedule motivated the researchers to test the system at 0.5 FSF. This approach would theoretically prevent the succeeding valve opening schedule from being deleted. The results showed that the approach was effective in preventing no sprays on the tested spray duration. However, the 0.50 FSF also increased the rate of multiple sprays (FS_2), from 1% to 7%, at 200 ms. Note that multiple sprays were only observed at 200 ms spray duration for both tested values of FSF and also caused the wrong sprays. Most likely, the small size of the filter did not delete succeeding valve opening schedules for the same target weed. To some extent, this observation can also be beneficial as multiple sprays often occur on large target plant samples. Depending on the degree of random fluctuations of weed coordinates and velocity estimates, some valve opening schedules, for the same weed can be outside the filter, resulting in multiple sprays.

On the other hand, the 0.5 FSF at 350 and 500 ms spraying duration did not prevent the adjacent non-targets from being covered by the previous spraying. Further, it also resulted in the second spray, when two succeeding weeds were present, being triggered late. These two scenarios resulted in a high rate of the wrong spray for the 350 ms (30%) and 500 ms (38%). Overall, the performance testing showed that the combination of the developed targeting algorithm and method for estimating $\vec{v}_{travel}$ using CNN-based MVS resulted in a very high spraying precision (93%) and recall (100%) at the optimum settings (low-level spray schedule filtering and 200 ms spray duration).

The performance testing with three SUs was then conducted at $0.87 \pm 0.14 \frac{m}{s}$ ($3.14 \pm 0.49 \frac{km}{h}$), which was slower than the first experiment due to the additional effort to keep each SU aligned to each designated row. Table 31 then summarizes the detection performance of each of the SUs in the second experiment and shows no difference with the first experiment despite the difference in $\overrightarrow{v}_{travel}$ since the experiment was within the identified allowable operating condition.

Table 31

The Detection Performance of the MAPS with Three SUs at Low-Level Spray Schedule Filtering, 200 Spraying Duration, and $0.87 \pm 0.14 \frac{m}{s}$ *(*$3.14 \pm 0.49 \frac{km}{h}$*)*

Scalable Unit	TP	FP	TN	FN	p_d	r_d
Left	12	0	8	0	1.00	1.00
Middle	10	0	10	0	1.00	1.00
Right	8	0	12	0	1.00	1.00

The targeting performance with three scalable units and optimum values of $t_{spraying}$ and FSF are then summarized in Table 32. The obtained performance was similar to the first experiment at the same $t_{spraying}$ and FSF. However, due to a slower $\overrightarrow{v}_{travel}$ than in the first experiment, there were higher rates of multiple sprays (16%) caused by a smaller filter and partial (7%) sprays caused by a shorter effective spray distance. Similarly, the slower $\overrightarrow{v}_{travel}$ also slightly reduced the rate of wrong sprays (2%) due to the smaller spray distance than the first experiment.

Table 32

The Targeting Performance of the MAPS with Three SUs at Low-Level Spray Schedule Filtering, 200 Spraying Duration, and $0.87 \pm 0.14\ \frac{m}{s}$ *(*$3.14 \pm 0.49\ \frac{km}{h}$*)*

Scalable Unit	TT	TS	FS_1	FS_2	FS	PS	MS	WS	NS	p_s	r_s
Left	12.00	12.33	0.92	0.06	0.97	0.03	0.00	0.03	0.00	0.97	1.00
Middle	10.00	10.00	0.67	0.27	0.93	0.07	0.00	0.00	0.00	1.00	1.00
Right	8.00	8.33	0.71	0.17	0.88	0.13	0.00	0.04	0.00	0.96	1.00
All	30.00	30.67	0.76	0.16	0.93	0.07	0.00	0.02	0.00	0.98	1.00

In general, the performance using a single or multiple SUs at the optimum condition had a minimal difference, and variations were mainly caused by the difference in $\overrightarrow{v}_{travel}$. However, the system can still be improved by developing an algorithm that will calculate a variable instead of a fixed spray duration due to the limitations of the latter method. Using a lower $\overrightarrow{v}_{travel}$ than what was used in the tests, as demonstrated in the experiment with three SUs, would result in higher rates of partial and multiple sprays. In contrast, using a higher $\overrightarrow{v}_{travel}$, smaller weed samples, or closer hill spacing than what was utilized experiment would result in higher wrong sprays or over-application.

6.3.5 Summary of Observations

This chapter demonstrated that vision-based velocity estimation combined with VTD queuing and dynamic filtering could achieve accurate spot spraying without using auxiliary velocity measurement systems. In the next chapter, the spraying performance of the developed precision sprayer is tested in a broadcast-seeded crop. Precision sprayer designs can be potentially simplified with the results, and total system costs can consequentially be reduced.

Chapter 7

Field Evaluation of a CNN-Based Modular Precision Sprayer [1]

7.1 Introduction

This chapter presents the field test methods and results of the MAPS. The targeting performance and spray volume reduction (SVR) of the MAPS in broadcast-seeded soybean (*Glycine max* L.) were evaluated.

Farmers commonly plant soybeans in rows [216, 217, 218]. However, in recent years, growers have also been adopting conservation practices and broadcast seeding to reduce labor at the cost of minimal yield loss [219, 220]. Still, the absence of distinct row-spacing in broadcast-seeded fields and risks of crop damage by mechanical weed control compel farmers to spray herbicide [59].

Various attempts have already been made in the past to recognize weeds in soybean fields using MVS-CNN for the purpose of precision spraying. For example, past studies showed that CNN-based weed detection could achieve 65% to 99.5% precision [128, 221]. Nonetheless, the developed CNN-based precision sprayer for soybean had low targeting accuracy, ranging from 20% to 78% [136]. In addition, the precision sprayer operated only at $0.5\frac{m}{s}$ [136], compared to 5.6 $\frac{m}{s}$ using spectrometric sensors [137] and 1.17 $\frac{m}{s}$ using MVS with traditional image processing [57].

The high computational requirement of CNN-based detection using image sensors, despite having high detection performance and robustness compared to spectrometric, optical, and distance sensors, hinders its wide adoption for real-time precision spraying [55]. To solve the high computational cost of CNN, various CNN architecture designs and optimizations were proposed and showed promising results. Some of these techniques include using single-stage architectures such as Single Shot MultiBox Detector (SSD) [63] and You Only Look Once (YOLO) [65], implementing depth-wise separable convolutions [64,

[1]Some parts of this chapter were published in [215].

222], reducing feature reuse within convolution [223] and optimizing trained CNN model using TensorRT [90].

In this chapter, the previously developed modular agrochemical precision sprayer (MAPS), which localized the computational load of TensorRT-optimized SSD-MB1 (SSD-MB1-TRT) among multiple low-power and low-cost hardware for precision spraying, was tested for actual field performance. The modular design approach, in effect, would increase the operating travel velocity of CNN-based precision sprayers without the need for powerful desktop- or server-grade systems by having dedicated computational hardware for each capture device.

7.2 Materials and Methods

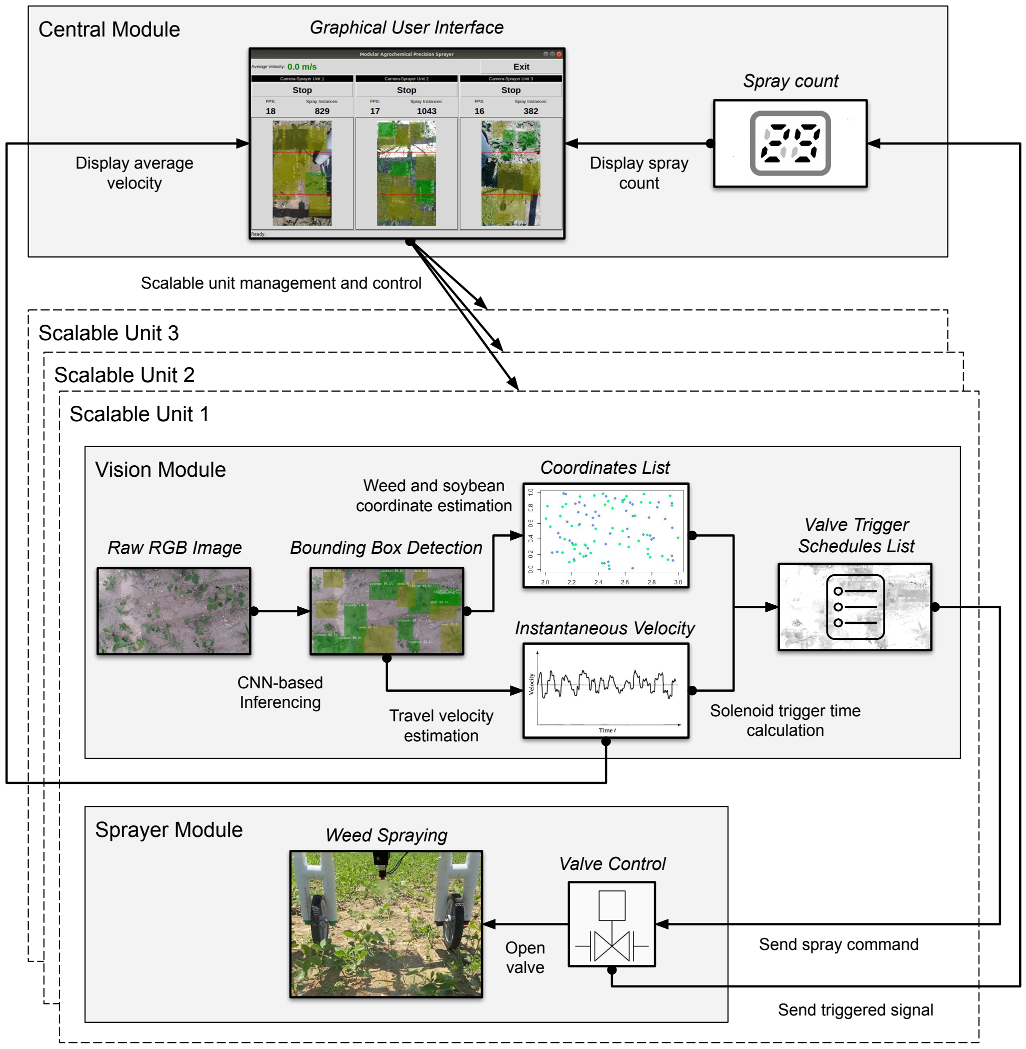
7.2.1 Overview of the MAPS

Figure 79 summarizes the general workflow of each scalable unit (SU). The process starts by capturing a top view of the soybean plot using an RGB camera (Logitech StreamCam). Then, the vision module (Nvidia Jetson Nano 4GB) performs CNN-based inferencing. This process detects and generates bounding boxes of soybeans and weeds in the image. The vision module then stores the frame coordinates of soybeans and weeds in separate lists. The detection algorithm then analyzes the list to track and estimate the distance traveled by plants between two consecutive frames.

The velocity estimation algorithm then estimates the relative velocity of the sprayer using the elapsed time and the distance traveled by tracked plants. Finally, knowing the instantaneous distance of the weeds from the sprayer nozzle and travel velocity, the targeting algorithm calculates the variable time delays to schedule the time to trigger the sprayer. These time schedules were then stored in a spray schedule list. Note that the vision module appends spray schedules whenever a weed is detected in a frame. Thus, the vision module clears elapsed spray schedules within the effective spray region of the previous spraying whenever it receives a successful valve opening and closing feedback from the sprayer module. This step filters the spray schedules in the list to prevent multiple spraying on already sprayed weeds.

Figure 79

Workflow and Communication of the SU of the MAPS



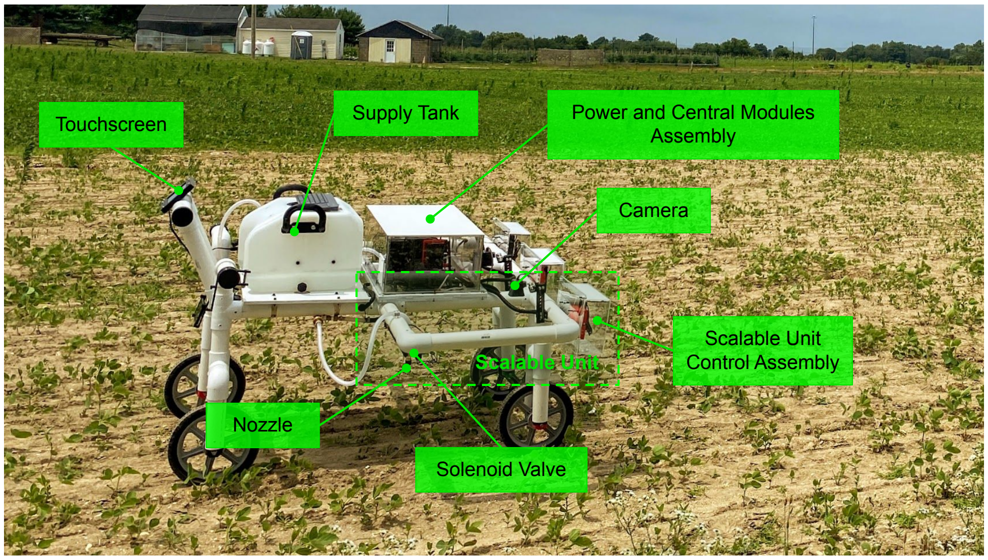
The spraying algorithm runs as a parallel process in the vision module to regularly check the list for elapsed spray schedules. The process then publishes a trigger signal to a ROS topic if a time value in the spray schedule has been reached. A USB-connected microcontroller (Arduino Nano), subscribed to the trigger signal topic, closes the relay switch (Arceli KY-019) when a trigger signal is published. This event causes the solenoid valve (US Solid USS2-00006) to open for 0.2 s and delivers liquid to a fan-type nozzle (Solo 4900654-P). The microcontroller then publishes a signal to a feedback ROS topic to indicate that the valve was triggered. The central and vision modules then finally read this message to update the spray count and clear any spray schedules that were within the previous spraying time.

7.2.2 Experimental Field

The image dataset collection and field testing were performed in the agricultural field in South Jersey Technological Park of Rowan University, Glassboro, New Jersey, USA. Figure 80 illustrates the test field on June 30, 2022, with the MAPS. The farm, located at 39°43’08.1"N and 75°08’52.5"W, was broadcast-seeded with Pioneer-brand soybean variety during the first week of June 2022 at approximately 395,000 seeds per hectare.

Figure 80

The Developed MAPS with Labeled Components in the Experimental Field

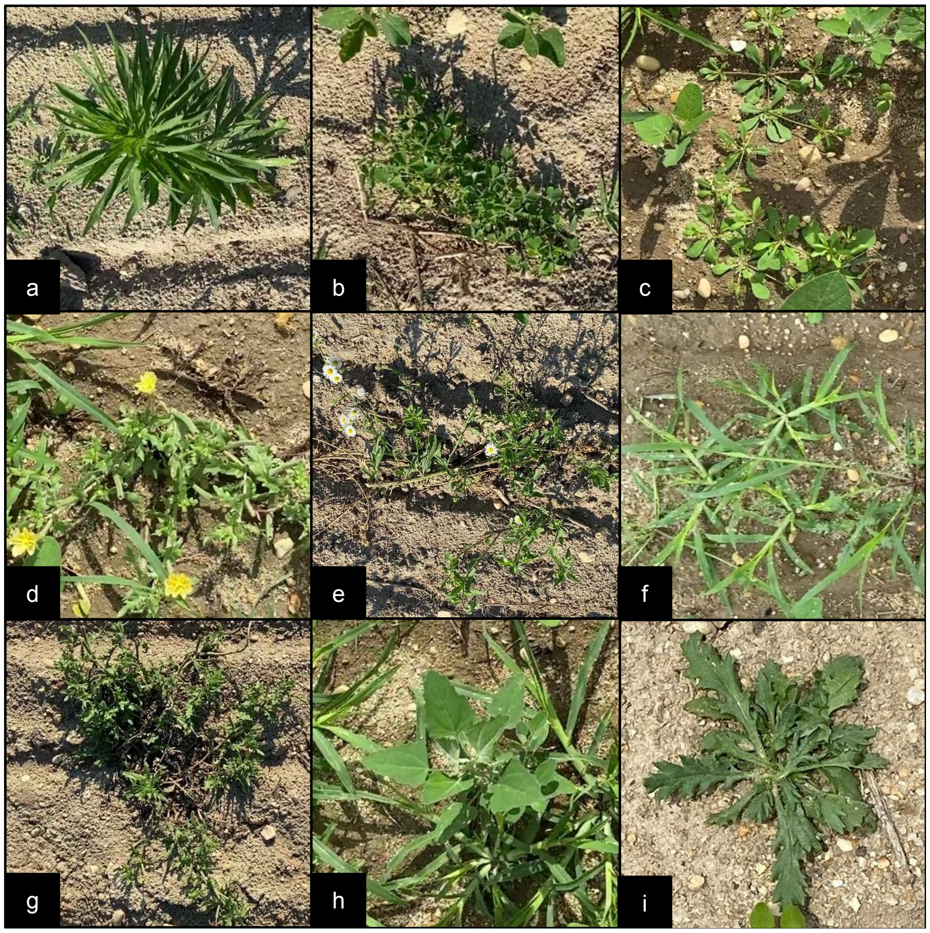

Touchscreen
Supply Tank
Power and Central Modules Assembly
Camera
Scalable Unit
Scalable Unit Control Assembly
Nozzle
Solenoid Valve

7.2.3 SSD-MB1 Training and Validation

Several videos were recorded from June 30 to July 8, 2022, at approximately 4-day intervals, using two video capture devices (Apple iPhone 11 and Logitech StreamCam). 877 RGB images were extracted from the captured videos and then annotated as "soybean" and "weed" using LabelImg [182]. Figure 81 illustrates sample weeds in the test field. The final dataset contained 5,080 and 6,934 instances of soybeans and weeds, respectively, divided randomly into 80% training and 20% validation.

Figure 81

Sample Images of Target Weeds: (a) Horseweed, (b) Purslane, (c) Carpet Weed, (d) Cut-Leaved Evening Primrose, (e) Hairy Fleabane, (f) Goosegrass, (g) Ragweed, (h) Lambquarter, and (i) Thistle



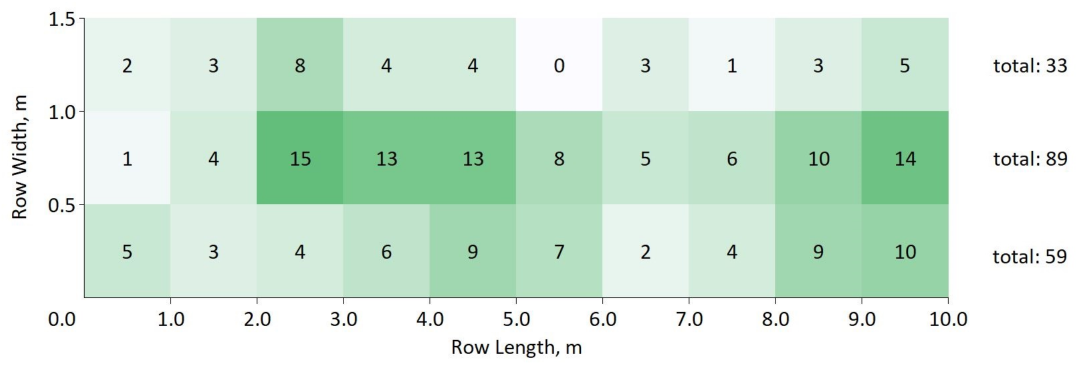
The SSD-MB1 model was trained using an Nvidia Jetson NX Xavier for about 4,000 epochs at an initial learning rate of 0.005, base learning rate of 0.0005, momentum of 0.9, weight decay of 0.00005, and batch size of 24. The performance of the trained CNN model was then evaluated using PASCAL VOC 2007 metrics, including precision (P), recall (R), average precision per class (AP), and mean average precision (mAP) at 0.5 intersection over union (IOU) threshold. It was then optimized using TensorRT on the Nvidia Jetson Nano.

7.2.4 Field Testing

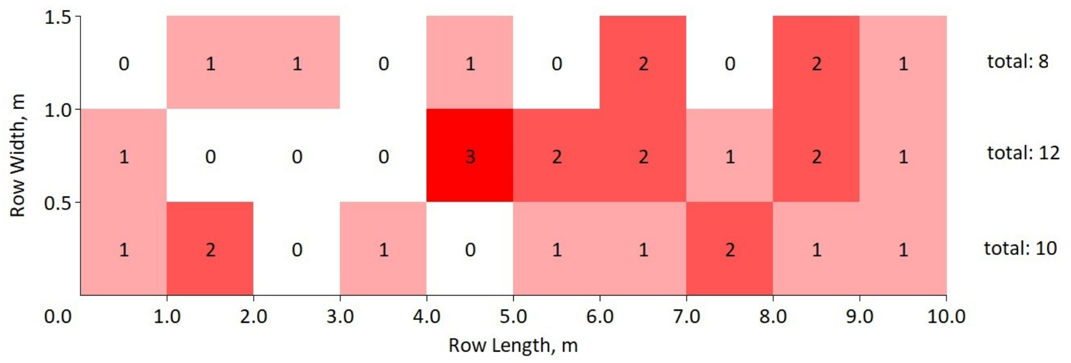
Fields tests were performed on July 11, 2022 (28.9 °C, 11.1 $\frac{km}{h}$ wind speed, 42-44% relative humidity) on three adjacent 0.5 m $\times$ 10 m rows of broadcast-seeded soybeans with randomly growing weeds. A video of the test plots was recorded and analyzed per frame to construct the 1-meter resolution maps for the distribution and location of the weeds and samples in each test row. Figure 82a illustrates the weed distribution for each row at approximately 5.00, 13.48, and 8.94 $\frac{weeds}{m^2}$ on the left, middle, and right test rows, respectively. 30 target weeds (N_w) and 30 non-target soybeans (N_s) were randomly selected among the test rows, as shown in Figures Figure 82b and Figure 82c, respectively. 10 soybeans had no adjacent weeds (Figure 82c), also referred to as soybean without weeds ($N_{s|wow}$). The performance testing was then performed in three trial runs.

Figure 82

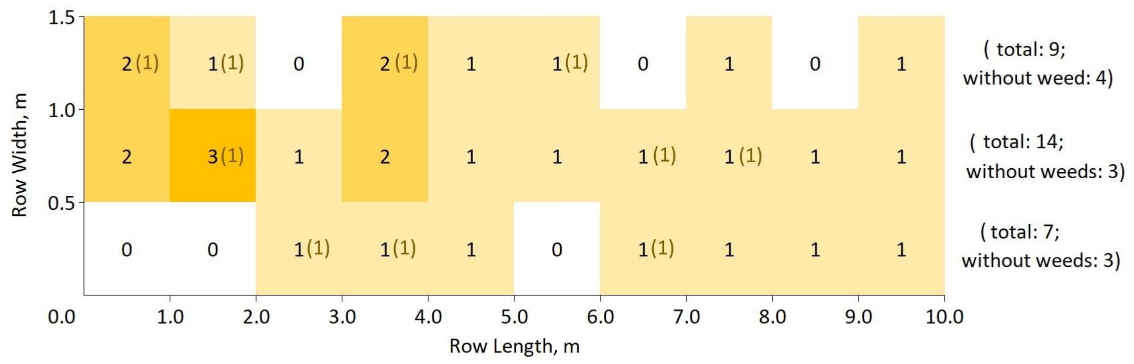
(Weed and Soybean Distributions in the Three Test Rows with Values Enclosed in Parenthesis Represent Soybean Plants Without Adjacent Weeds



(a) *Weed Population*



(b) *Weed Sample Distribution*



(c) *Soybean Sample Distribution*

7.2.4.1 Targeting Performance. Target weeds and non-target soybeans were labeled for visual reference during the evaluation. In weed spraying, correctly sprayed weeds were considered True Positives (TP). Incorrectly sprayed soybeans were considered False Positives (FP). Unsprayed soybeans were True Negatives (TN), and unsprayed weeds were False Negatives (FN). Spraying precision (p_s) and recall (r_s) were then calculated using equations Equation 37 and Equation 38, respectively. The wrong spraying rate (WS) was calculated using Equation 39. Lastly, due to the proximity of soybean samples to weeds, the non-targeting rate (NT) or the fraction of unsprayed soybeans without adjacent weeds was also determined using Equation 40.

$$p_s = \frac{TP}{TP+FP} \tag{37}$$

$$r_s = \frac{TP}{TP+FN} \tag{38}$$

$$WS = \frac{FP}{TN+FP} \tag{39}$$

$$NT = \frac{TN}{N_{s|wow}} \tag{40}$$

7.2.4.2 Spray Volume Reduction. The average variable spray volume in each row ($Q_{row|variable}$) was estimated by calculating the product of the average number of actuation of the solenoid valve ($N_{valve|row}$), nozzle flowrate (Q_{nozzle}), in $\frac{L}{min}$, and nozzle opening time ($t_{spraying}$), in seconds, as shown in equation Equation 41.

$$Q_{row|variable} = N_{valve|row} \times Q_{nozzle} \times t_{spraying} \tag{41}$$

The sprayer was pre-calibrated and tested the flowrate of each nozzle using ASAE

EP367.2 MAR1991 (R2017) standard [121], yielding Q_{nozzle} of 1.6 $\frac{L}{min}$. The average uniform spray volume for each row ($Q_{row|uniform}$) was then the product of the Q_{nozzle} and spraying trial time (t_{trial}), as shown in equation Equation 42. Finally, based on the pre-calibrated flow rate of the nozzle, the spray volume reduction (SVR) was calculated using equation Equation 43.

$$Q_{row|uniform} = Q_{nozzle} \times t_{trial} \tag{42}$$

$$SVR = \frac{Q_{row|uniform} - Q_{row|variable}}{Q_{row|uniform}} \tag{43}$$

7.3 Results and Discussion

7.3.1 CNN Model Performance

Results showed that the model detected soybean better than weeds (Figure 83). For instance, at 68% recall and 0.5 IOU threshold, the trained model had 71% and 90% precisions in detecting weeds and soybeans, respectively. Our results were slightly higher than the 65% precision, at the same recall and IOU threshold, obtained by [128] using SSD to detect weed patches in row-planted soybeans. Figure 84 shows that most small soybeans and weeds were undetected at a 0.5 confidence threshold. However, this situation was not consequential in actual spraying scenarios, as small weeds would have difficulty competing with large soybeans for resources.

Figure 83

The Precision-Recall Curve of the SSD-MB1 Model on Detecting Soybean and Weeds at 0.5 IOU Threshold

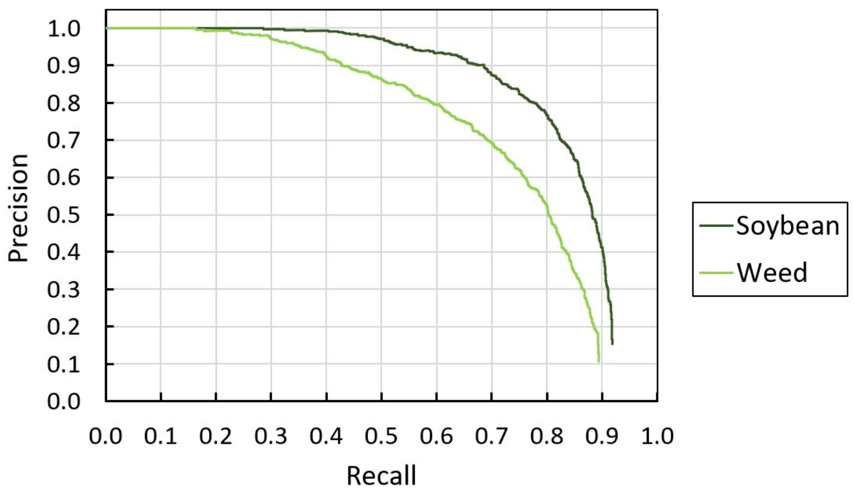


Figure 84

Soybean and Weed Detections of SSD-MB1 at 0.5 Confidence Threshold

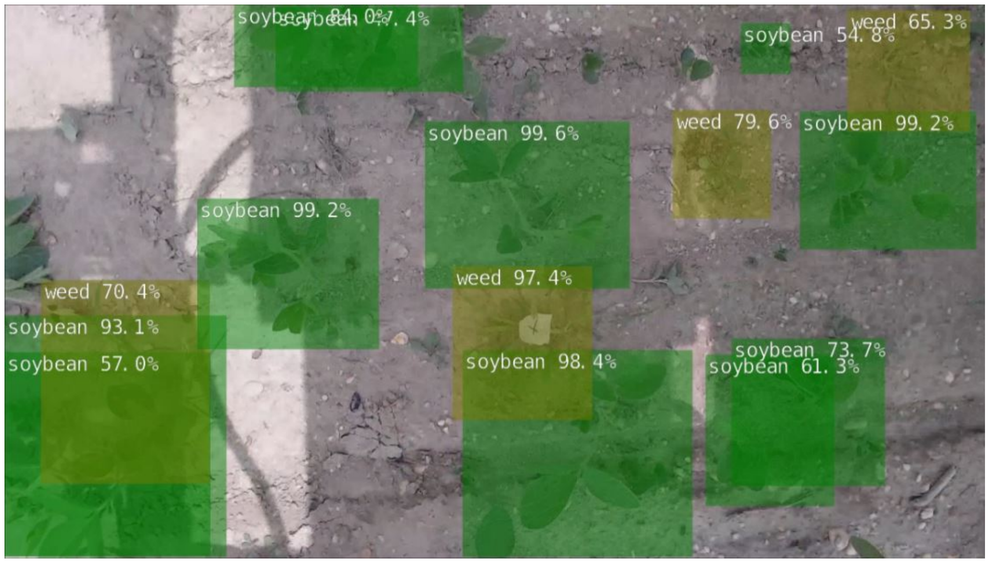


Table 33 then summarizes the detection performance of the trained model. Compared to a sprayer with YOLOv3-tiny running on an Nvidia GTX 1050 mini PC [86], our design (76.0% $mAP^{0.5}$ and 19 fps) at approximately 30% the hardware cost had similar detection performance (76.4% $mAP^{0.5}$) but 40% slower (31.5 fps).

Table 33

Soybean and Weed Detection Performance at 0.5 Threshold IOU of SSD-MB1

Class	$AP^{0.5}$**, %**	$mAP^{0.5}$**, %**	**Inference Speed, fps**
Soybean	81.4	76.0	19.0
Weed	70.6		

7.3.2 Targeting Performance

We observed that the MAPS successfully sprayed all sampled target weeds, as summarized in Table 34, and can be mainly attributed to queuing multiple spray schedules and operating within the maximum velocity. Despite a $mAP^{0.5}$ of 76%, traversing at 0.69 $\frac{m}{s}$ and 19 fps enabled inferencing on multiple frames. Our weed targeting algorithm only required a single frame with correct weed detection and velocity estimate to calculate an accurate spray schedule. If a weed was incorrectly detected in a frame, the weed can still be possibly detected in succeeding frames.

Table 34

Targeting Performance of the MAPS at 0.69 $\pm 0.13\ \frac{m}{s}$

Trial	TP	TN	FP	FN	$N_{s\|wow}$	p_s, %	r_s, %	WS, %	NT, %
1	30	7	23	0	10	56.66	100.00	76.67	70.00
2	30	7	23	0	10	56.66	100.00	76.67	70.00
3	30	9	21	0	10	58.82	100.00	70.00	90.00
Average	30	8	22	0	10	57.32	100.00	74.44	76.67

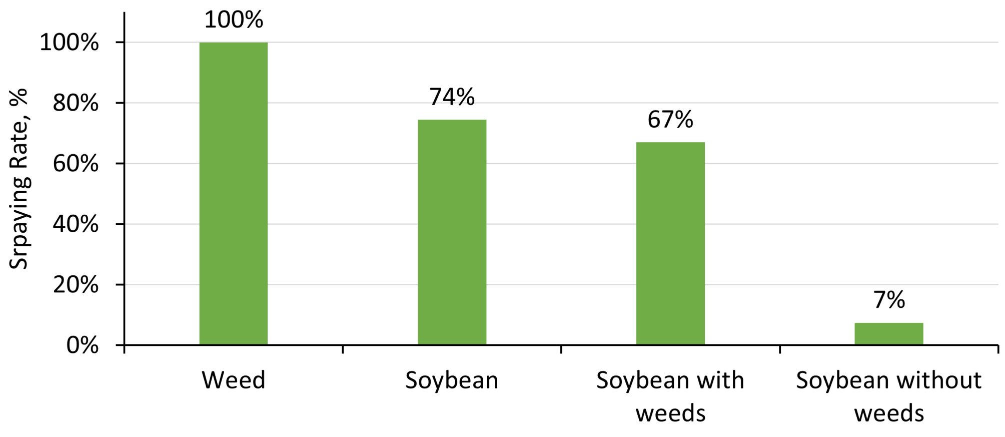
Comparing the results with other studies, our system had a higher weed spraying recall than a precision sprayer for soybeans that targeted 78% of weeds using Mask R-CNN [136]. Similarly, the high spraying recall despite low detection performance was also observed in [75]. Despite having 57% precision and 84% recall, their system still achieved 96% average spraying recall by needing only a single correct detection among the processed frames. However, the 4 fps effective inference speed of their system was approximately four times slower than that of our configuration. This situation resulted in more frames available in our system for inferencing and most likely contributed to the higher spraying recall than [75].

On the other hand, the average spraying precision (57.32%) of the MAPS was relatively low compared to the results of [79] at 78% and [224] at 96.67%. However, their precision sprayers were tested on plants arranged in distinct rows with only 30 weeds, compared to our test site with a broadcast-seeded layout and six times the weed population. Evaluating the results, the wide coverage of the nozzle, unintended sprays, and the natural proximity of targets and non-targets caused the low spraying precision.

As described in Figure 82c, only 10 soybeans, or 33% of the non-targets, had no adjacent weeds. This high rate of FP of the sprayer caused by coarse nozzle resolution and proximity of non-targets to targets was also observed in [75]. Their system sprayed 50% of non-targets on a row-seeded field, on average, due to proximity to targets [75]. On the other hand, our precision sprayer has approximately sprayed 74% of non-targets on a broadcast-seeded layout, compared to the theoretical 67% wrong spray rate. This theoretical wrong spray rate represented the percentage of soybean with adjacent weeds during the test as summarized in Figure 85. Nonetheless, despite low precision due to coarse nozzle resolution, only a 7% increase against the theoretical wrong spray rate was observed.

Figure 85

Spray Rate of Weeds and Soybeans

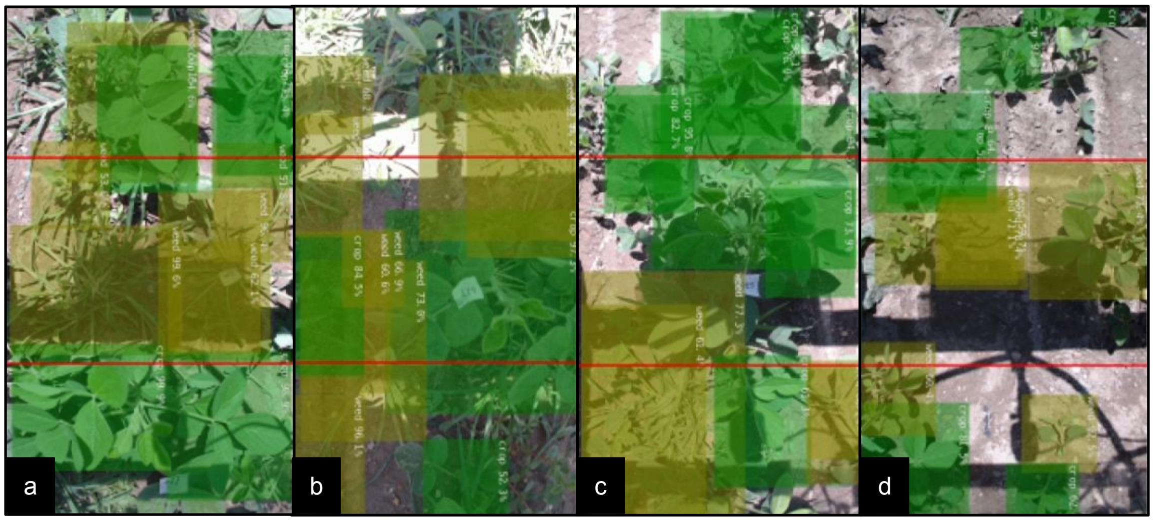


The sprayer successfully avoided spraying 76.67% of soybeans without adjacent weeds, on average. Still, due to inaccuracy and extreme fluctuations in weed location and travel velocity estimates, the multiple spread-out spraying schedules for a weed and incorrect detections also caused unintentional spraying of 10% to 30% of soybeans without adjacent weeds. If soybean is incorrectly detected as weed, this situation results in incorrect targeting. Moreover, the algorithm measures the distance traveled by a detected plant between two consecutive frames. If a wrong detection occurs in the succeeding frame, the error contributes to inaccurate velocity estimates and may cause the valve to open when soybeans are directly below the nozzle.

Figure 86 illustrates sample detection scenarios showing the labeled target and non-targets during the experiment. An ideal scenario is shown in Figure 86a, where the weed at the center of the frame only had another weed in proximity along the horizontal axis of the frame. Furthermore, the soybeans at the bottom of the frame also had no adjacent weeds, resulting in them not being sprayed. This case was also observed in Figure 86c, where the soybeans at the top of the frame were not sprayed. Figure 86b, on the other hand, shows a non-target soybean in the middle of the frame with weeds growing on its left side. This non-target soybean was unintentionally sprayed. Lastly, Figure 86d shows a non-target soybean detected as a weed causing it to be sprayed.

Figure 86

Sample Detection Scenarios During the Experiment with Labeled Targets Weeds (Yellow) and Non-Target Soybeans (Green)



Finally, the MAPS at 0.69 $\frac{m}{s}$ (2.5 $\frac{km}{h}$), on average, traveled up to three times that of similar CNN-based sprayers with known targeting performance. The lowest and highest recorded velocities during the test were 0.53 $\frac{m}{s}$ (1.9 $\frac{km}{h}$) and 0.83 $\frac{m}{s}$ (3.0 $\frac{km}{h}$), respectively, and varied due to the rough terrain of the field. Theoretically, each SU could spray at

3.54 $\frac{m}{s}$ (12.7 $\frac{km}{h}$). However, the current push-type configuration limited the test velocity to walking speeds. Compared to other systems, [224] reported 0.28 $\frac{m}{s}$ (1.0 $\frac{km}{h}$). [136] tested their sprayer at 0.5 $\frac{m}{s}$ (1.8 $\frac{km}{h}$) but had 20% to 78% targeting accuracy. On the other hand, [86] reported 1.38 $\frac{m}{s}$ (5.0 $\frac{km}{h}$) on the field test of their developed sprayer, but the field targeting performance was not quantified.

7.3.3 Spray Volume Reduction

Table 35 showed that the average count of the solenoid valve opening tends to increase with the weed population, an indicator of the variable spraying capability. The left row (row 3) had the lowest weed population (33) and the lowest spray instances (38). Similarly, the middle row (row 1) with the highest weed population (89) had the highest spray count (57). Considering the middle row (row 2), the average number of spray instances was less than double the left row despite the former having tripled the weed population of the latter. The spray instances tend to increase at a diminishing rate as the weed population increases (Figure 87a). In addition, the SVR decreases as the weed population increases (Figure 87b). Theoretically, the 0.2 s spray duration and 0.69 $\frac{m}{s}$ velocity would result in a maximum of 73 spray instances for a 10-m row fully covered with weeds. However, the area where weeds could grow was limited, causing additional weeds to be within an effective region of a single nozzle spray after a certain weed population level.

Table 35

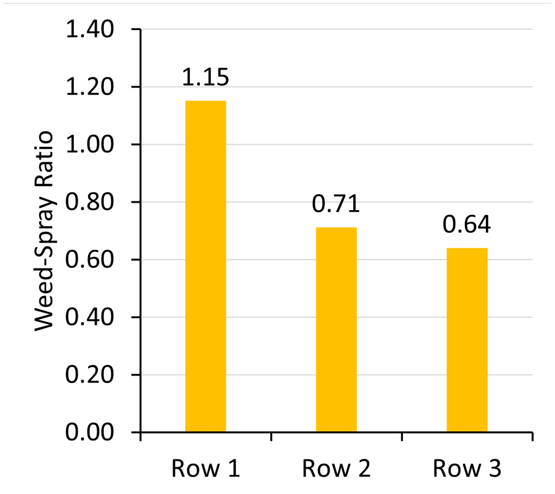
Spray Volume Reduction of the MAPS at 0.69 $\pm$ 0.13 $\frac{m}{s}$ and 15-Second Average Traverse Time

Row	$N_{w\|total}$	$N_{valve\|row}$	$\frac{N_{valve\|row}}{N_{w\|total}}$	$Q_{row\|variable}{}^{a}$, **L**	SVR^{b}, %
Left	33	38.33 $\pm$ 7.72	1.16	0.204 $\pm$ 0.012	48.89
Middle	89	57.33 $\pm$ 11.09	0.64	0.306 $\pm$ 0.0.059	23.56
Right	59	41.67 $\pm$ 10.21	0.71	0.222 $\pm$ 0.051	44.44
All	181	137.33 $\pm$ 22.54	0.76	0.732 $\pm$ 0.120	38.96

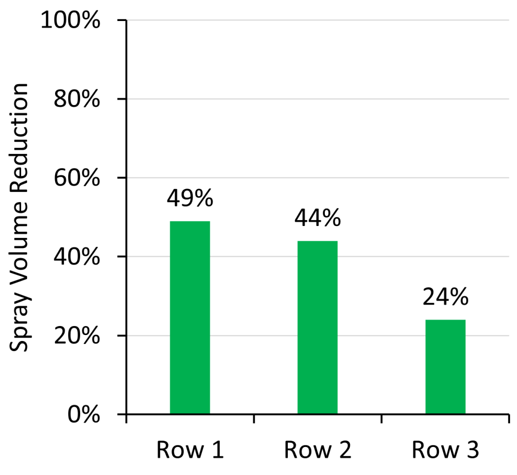
[a] Calculated at 0.2 s spray duration.; [b] Calculated at 1.2 $\frac{L}{s}$ continuous nozzle delivery rate.

Figure 87

The Weed Population, Spray Instances Ratio, and Spray Volume Reductions



(a) *Weed-Spray-Ratio*



(b) *Spray Volume Reduction*

The results also demonstrated that the sprayer had an average spray volume reduction of 38.96%, most likely representing the bare soil and soybean-only regions in the experimental field. The highest spray reduction was observed in the left row (48.89%), where the weed population was also the lowest. Consequently, the middle row had the lowest volume reduction (23.56%), having the lowest weed population. However, direct comparisons with developed precision sprayers in other studies were difficult as the weed population of their test area was not reported. In [137], their spectral-sensor-based sprayer recorded 24.21% to 63.15% SVR on the post-emergence spray. On the other hand, [86] calculated 50.8% to 52.5% SVR using their CNN-based precision sprayer.

7.3.4 Performance Summary

In this study, we demonstrated that a modular precision sprayer with multiple low-cost and low-power devices could increase the operating velocity of CNN-based precision sprayers without needing powerful and expensive computational hardware. Our method of queuing multiple spray schedules by analyzing multiple frames enabled accurate targeting of weeds during our tests. The performance testing also demonstrated the variable spraying functionality of our design when the number of spray instances of each SU increases at a diminishing rate as the weed population in the test rows increases.

Chapter 8

Conclusion and Future Works

8.1 Conclusion

The dissertation demonstrated that the modular precision sprayer with vision-based velocity estimation and VTDs could potentially reduce the cost, increase adaptability, and increase the operating velocity in agricultural field environments. SSD-MB1 optimized using TensorRT and running on an NVIDIA Jetson Nano was identified as a low-cost and energy-efficient configuration for running real-time weed detection CNN models. During the study, the total cost of the developed vision module was USD 261.73.

Further, unlike past designs with landscape camera orientations and fast GPUs, the developed models showed that a portrait camera orientation and a Jetson Nano with SSD-MB1-TRT could provide sufficient performance for precision spraying at typical $\overrightarrow{v}_{travel}$ of spraying operations. A reconfigurable and scalable precision sprayer with CNN-based MVS was fabricated based on the designed modular hardware and software architectures. The prediction of the model was verified by demonstrating in the tests the absence of missed detections at tested velocities up to 1.19 $\frac{m}{s}$ (4.28 $\frac{km}{h}$), which was above average walking velocities of hand-held spraying and near the average velocities of boom spraying. The tests also showed that the developed precision sprayer could target plants accurately at 2% and 7% incorrect sprays of non-target plants during the laboratory and field tests, respectively.

8.1.1 MVS-CNN Benchmarking

The performance in different hardware of one-stage CNN object detection models for weed detection was compared. In general, YOLO-based models were faster to train than SSD-based models. Further, the YOLOv5s model exhibited the highest *mAP*, particularly at 0.5 to 0.95 *IoU* thresholds, among the tested CNN models. Despite high processing speeds in the laptop with RTX2080, YOLOv5s was limited by its slow loading and inference speeds in the Jetson Nano. On the other hand, the optimized SSD-MB1-TRT model on the Jetson Nano had statistically similar $mAP^{0.5}$ and $mAP^{0.5:0.95}$ to YOLOv5s while providing sufficient inference speed at the highest cost efficiency among tested configuration. These characteristics of SSD-MB1-TRT on a Jetson Nano directly addressed the current concerns of PA and CNN-based MVS technologies, which are: (1) the high cost of PA technologies and (2) the low-productivity of CNN-based MVS for field robotics. Therefore, SSD-MB1-TRT on a Jetson Nano was the most compelling configuration for our application.

The study also demonstrated that data augmentation and a balanced dataset would not always improve the detection performance of the one-stage CNN models, as only SSD-MB1 and Scaled-YOLOv4-CSP demonstrated statistically significant improvement. Thus, we recommend that data augmentation is unnecessary when training YOLOv5s and SSD-MB2. In addition, using the generated image dataset of mulched onion, we verified the results of past studies that one-stage CNN object detection architectures could be utilized to identify and locate weeds in a dense agricultural environment. We demonstrated that the processing speeds of tested CNN architectures in a laptop with high-performance discrete graphics were sufficient. However, model optimization is recommended to utilize low-cost and low-power hardware, such as Jetson Nano, for real-time CNN-based weed detection.

8.1.2 MVS-CNN Modeling

Theoretical and simulation methods in determining the effect of fps, $\overrightarrow{v}_{travel}$, and S on missed plant detections of an MVS-CNN were also developed. We introduced the dimensionless parameter r_o as a theoretical predictor of the application-level detection recall or r_d. The reliability of r_o in predicting the r_d of a vision system as a function of inference speed and travel velocity was successfully demonstrated by having no margin of error compared to simulated and actual MVS at a sufficient traversed distance ($\geq$ 10 m). In addition, a set of Python scripts for simulating the performance of a vision system for plant detection was also developed. Computer simulations at different *fps* and $\vec{v}_{travel}$ showed no margin of error compared to the r_d of actual MVS. This set of scripts was made publicly available to verify the results of this study and provide a practical tool for developers in optimizing design configurations of a vision-based plant detection system.

The mechanism of missed detection was also successfully illustrated by evaluating each simulated frame in detail. Using the concept of r_o, simulation, and detailed assessment of each processed frame, the mechanism to prevent missed plant hills by increasing the effective S through synchronous multi-camera vision systems in low-frame processing rate hardware was also successfully presented.

Furthermore, a CNN-based vision module was also successfully developed and tested. Performance testing showed that the $r_{d,th}$ and $r_{d,sim}$ accurately predicted the r_d of the vision module with no margin of error. The script for the vision module was also made available in a public repository where future improvements shall be uploaded.

A reference chart was also developed to aid in selecting the minimum inference speed depending on the camera field of view and operating velocity of the desired field operation. Based on simulations and the performance of actual systems, the proposed reference chart was shown to be a reliable tool in the development of MVS-CNN for plant detection in field operations.

8.1.3 MAPS Development

A modular software framework and hardware architectures for a precision sprayer with MVS-CNN were also designed. Based on these frameworks, a prototype of a precision sprayer with MVS-CNN for detection and velocity estimation was successfully fabricated and tested. The application of Euclidean-based tracking, buffer regions in the captured frame, and derived analytical methods to calculate LCR resulted in a reliable method of velocity estimation.

Performance testing showed that the proposed velocity estimation and targeting algorithms, based on queuing and dynamic filtering of VTDs, had high accuracy within the identified and designed operating conditions. The results showed that the velocity estimates agreed with actual measurements with a mean absolute error of 0.036 $\frac{m}{s}$ (0.13 $\frac{km}{h}$). Further, testing the targeting algorithm on rows of artificial crops and weeds at different levels of spraying duration and filter size factor (FSF) showed that short spraying duration and small FSF increase overall spraying accuracy. Finally, testing the MAPS using the optimum settings at up to 1.19 $\frac{m}{s}$ (4.28 $\frac{km}{h}$) successfully sprayed all targets. Further, only 2% to 7% of non-targets were sprayed at the low and high test velocities, respectively. With these results, this study suggests that vision-based velocity estimation combined with VTD queuing and dynamic filtering can be an accurate and low-cost solution for targeted spraying without using auxiliary velocity measurement systems.

8.1.4 Field Testing

Finally, the developed precision sprayer with CNN-based MVS and modular architecture demonstrated its capability to target weeds and reduce spray volume in a broadcast-seeded soybean field. By using multiple low-power devices to run the CNN model, the vision system achieved 19 fps and 76% $mAP^{0.5}$, at similar targeting accuracy, spraying performance, and faster average velocity of up to 0.82 $\frac{m}{s}$ (3.0 $\frac{km}{h}$) than other field CNN-based precision sprayers. Furthermore, the field test also verified the variable spraying capability of the modular design, reducing spray volume by up to 48.89% in the experiments. Nonetheless, direct comparisons with existing CNN-based precision sprayers were difficult due to differences in the experimental setups and the unavailability of weed distribution. As demonstrated in our results, high weed density lowers spraying precision and spray volume reduction. When weed density is high, the likelihood that weeds are next to a crop is also very high, causing non-target crops to be unintentionally sprayed. Similarly, as the weed population increases, the number of spray instances of a precision sprayer also increases at a diminishing rate as it approaches the maximum spray instances equivalent to uniform spraying.

Nonetheless, the broadcast-seeded layout and high weed density presented a challenging scenario for precision spraying. Despite these conditions, our CNN-based plant detection and vision-based velocity estimation proved to be doing well during operation regarding weed spraying recall and spray volume reductions. The spraying errors became secondary compared to the indirect spray caused by the wide effective spray region of the nozzle used in the test. Up to 90% soybean samples without adjacent weeds were not sprayed during our trials. In contrast, all soybean samples with adjacent weeds were unintentionally sprayed. The former errors were caused by inaccurate plant detection and velocity estimation, while the latter could be attributed to the coarse resolution of the nozzle.

8.2 Future Works

Despite accomplishing the set objectives in this research, the dissertation encountered limitations that shall be improved in the future. The coarse nozzle resolution resulted in low overall spraying precision due to the proximity of targets with non-targets in a broadcast-seeded layout. Velocity estimation inaccuracy and incorrect detections also contributed to unintended sprays on non-targets. Further, the MVS-CNN can also be potentially extended for mechanical weed control and fertilizer application. For these reasons, this study recommends pursuing the following future directions:

1. Use the crop as the only reference for travel velocity estimation of the MVS-CNN due to the higher detection reliability and more regular distribution of crops than weeds.

2. Explore more effective signal-filtering techniques to minimize drastic fluctuations of readings and integrate deep-learning-based tracking for velocity estimation.

3. Test the system for liquid fertilizer application.

4. Increase the number of nozzles per SU, as most incorrect sprays were due to wide nozzle coverage.
5. Implement a variable spraying duration to accommodate the varying size of the plant or the travel velocity of the sprayer.
6. Integrate the developed MVS-CNN on a robotic frame with autonomous navigation for fully automated spraying.
7. Test the system with the aforementioned improvements at standard spraying velocity as a tractor implement or smart attachment for an agricultural robot with autonomous navigation.
8. Integrate the developed MVS-CNN for mechanical weed control.

Appendix A

List of Acronyms, Abbreviations, Units, and Symbols

Table A1

List of Units and Symbols

α	Width Hyperparameter
A_1	Number of Fully-Sprayed Target Plant (Single Spray)
A_2	Number of Fully-Sprayed Target Plant (Multiple Sprays)
B	Number of Partially-Sprayed Target Plant
C	Number of Miss-Sprayed Target Plant
D	Number of Wrong Sprays on Non-Target Plant
$d_{e\|i,j}$	Bounding Box Centroid Euclidean Travel Distances in Consecutive Frames
$d_{e\|thresh}$	Threshold Travel Distance
$d_{i,k}$	Instantaneous Distance Between Detected Plant and Nozzle
$d_{o,k}$	Bounding Box Left Border and Nozzle Center Distance
$d_{o,k}$	Bounding Box Right Border and Nozzle Center Distance
d_f	Frame Displacement
d_h	Plant Hill Spacing
D_k	Width of Filter
d_l	Field Length
d_s	Camera-Nozzle Distance
E	Number of Not-Sprayed Target Plant
FN	False Negative
FP	False Positive
fps	Frames per Second

$fps_{effective}$	Application-Level or Effective Inference Speed
FS	Total Rate of Fully-Sprayed Target Plant
FS_1	Rate of Fully-Sprayed Target Plant (Single Spray)
FS_2	Rate of Fully-Sprayed Target Plant (Multiple Sprays)
h	hour
h_{cam}	Camera Height
I	Total Number of Sampling Points
i	Sampling Point or Plant Instance in Current Frame
IoU	Intersection Over Union
J	Total Detected Plants in Previous Frame
j	An Instance of a Detected Plant in Previous Frame
K	Total Number of Frames
k	Frame Instance
$\frac{km}{h}$	Kilometer per Hour
L	Liter
M	Depth of Filter
m	Meter
$\frac{m}{s}$	Meter per Second
mAP	Mean Average Precision
$mAP^{0.5:95}$	Mean Average Precision at 0.5 to 0.95 Intersection Over Union
$mAP^{0.5}$	Mean Average Precision at 0.5 Intersection Over Union
min	Minute
MS	Rate of Miss-Sprayed Target Plant
n_{vis}	Number of Vision Modules
n_d	Number of Detected Plant Hills
N	Depth of Input Network

NS	Rate of Not-Sprayed Target Plant
NT	Non-Targeting Rate
p	Image-Level Detection Precision
p_d	Application-Level Detection Precision
p_s	Spraying Precision
PS	Rate of Partially-Sprayed Target Plant
px	Pixel
Q_{nozzle}	Nozzle Flow Rate
$Q_{row\|uniform}$	Uniform Sprayed Volume per Row
$Q_{row\|variable}$	Variable Sprayed Volume per Row
r	Image-Level Detection Recall
R^2	Goodness of Fit
$r_{d,sim}$	Simulated Application-Level Detection Rate
$r_{d,th}$	Theoretical Application-Level Detection Rate
r_d	Application-Level Detection Recall/Rate
r_g	Gap Rate
r_o	Overlapping Rate
r_s	Spraying Recall
ρ	Resolution Hyperparameter
S, S_x	Field of View Parallel to Travel Direction
s_{buffer}	Horizontal Camera Width of the Buffer Region
S_{eff}	Effective Field of View Parallel to Travel Direction
s_x	Horizontal Camera Resolution
S_y	Field of View perpendicular to Travel Direction
s_y	Vertical Camera Resolution
t	Time

t_{filter}	Filter Time
t_k	Elapsed Time at Frame k
Δt	Time Difference
$\Delta t_{i,k}$	Instantaneous Plant Travel Time to Reach Nozzle
$\Delta t_{inference}$	Image-Level Inference Time
$\Delta t_{sprayer}$	Solenoid Valve Actuation Delay
$\Delta t_{vision\|k}$	Processing Time of the Vision Module
TN	True Negative
TP	True Positive
$\overrightarrow{v}_{frame}$	Plant Travel Velocity in Pixel per Second
$\overrightarrow{v}_{travel,max}$	Maximum Travel Velocity
$\overrightarrow{v}_{travel}$	Travel Velocity
W	Watt
W	Distance Between Nozzles
WS	Rate of Wrong Sprays on Non-Target Plant
X_i	Plant Hill Field Horizontal Coordinate
x_i	X-coordinate of a Detected Plant in the Current Frame
x_j	X-coordinate of a Detected Plant in the Previous Frame
y_i	Y-coordinate of a Detected Plant in the Current Frame
y_j	Y-coordinate of a Detected Plant in the Previous Frame

Table A2

List of Acronyms and Abbreviations

ANN	Artificial Neural Network
ANOVA	Analysis of Variance
ANSI	American National Standards Institute
ASABE	American Society of Agricultural and Biological Engineers
CA	Conservation Agriculture
CAN	Control Area Network
CCD	Charge-Coupled Device
CIGR	International Commission of Agricultural Engineering
CNN	Convolutional Neural Network
CSP	Scaling Cross Stage Partial Network
DC	Direct Current
DL	Deep-Learning
DSLR	Digital Single-Lens Reflex
FPN	Feature Pyramid Network
FR	Functional Requirements
FSF	Filter Size Factor
FTD	Fixed-Time Delay
GB	Gigabyte
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
I/O	Input-Output
IEEE	Institute of Electrical and Electronic Engineers
IMU	Inertial Measurement Unit

IR	InfraRed
ISO	International Organization for Standardization
LCR	Linear Capture Resolution
LiPO	Lithium Polymer
MAE	Mean Average Error
MAPS	Modular Agrochemical Precision Sprayer
MB1	MobileNet Version 1
MB2	MobileNet Version 2
MVS	Machine Vision System
NDVI	Normalized Difference Vegetative Index
NIR	Near-InfraRed
NO	Normally Open
PA	Precision Agriculture
PC	personal Computer
PWM	Pulse-Width Modulation
RC	Radio Communication
RCNN	Region-based Convolutional Neural Network
ReLu	Rectified Linear Unit
ResNet-50	Residual Network with 50 Convolutional Layers
RF	Radio Frequency
RGB	Red-Green-Blue
ROI	Region of Interest
ROS	Robot Operating System
RS232	Recommended Standard 232
RSF	Robot Software Framework
RSxxx	Recommended Standard xxx

RTK	Real-Time Kinematics
SAE	Society of Automotive Engineers
SIFT	Scale Invariant Feature Transform
SSD	Single-Shot Multibox Detector
SU	Scalable Unit
SURF	Speed-Up Robust Feature
SVM	Support Vector Machine
THSD	Tukey's Honestly Significant Difference
TRT	TensorRT
UART	Universal Asynchronous Receiver-Transmitter
UAV	Unmanned Aerial Vehicles
USB	Universal Serial Bus
USD	United States Dollar
VDC	Direct-Current Voltage
VGG-16	Visual Geometry Group with 16 Convolutional Layers
VTD	Variable-Time Delay
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
YOLO	You Only Look Once
YOLOv3	You Only Look Once Version 3
YOLOv4	You Only Look Once Version 4
YOLOv5	You Only Look Once Version 5

Appendix B

Vision Module Velocity Estimation Raw Calibration Data

Table B1

Hardware Configurations of the Different Calibration Setup of the Vision Module for Velocity Estimation

Parameter	h_{cam}, **m**		
	0.680	**0.705**	**0.730**
S_x , m	0.920	0.940	0.981
h_{crop}, m	0.330	0.330	0.330
h_{weed}, m	0.340	0.340	0.340
h_{plant}	0.335	0.335	0.335
PHF	0.500	0.500	0.500
$S_{effective}$, m	0.693	0.717	0.756
s_x, px	1280	1280	1280
LCR, m/px	0.000542	0.00056	0.000591

Table B2

Raw Calibration Data of the Vision Module for Velocity Estimation at h_{cam} = 0.690 m and $d_{e|thresh}$ = 100 px

Motor Setting	Trial	Crop		Weed	
		Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$	Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$
10	1	0.049	147.00	0.050	102.00
	2	0.049	145.00	0.050	108.00
	3	0.050	133.00	0.050	100.00
20	1	0.115	253.00	0.117	299.00
	2	0.116	281.00	0.115	273.00
	3	0.117	228.00	0.115	235.00
30	1	0.198	293.00	0.198	365.00
	2	0.198	419.00	0.204	347.00
	3	0.203	398.00	0.200	377.00
40	1	0.259	470.00	0.287	490.00
	2	0.254	516.00	0.254	492.00
	3	0.258	565.00	0.258	436.00
50	1	0.312	531.00	0.305	525.00
	2	0.317	597.00	0.316	486.00
	3	0.311	653.00	0.310	557.00
60	1	0.362	501.00	0.362	623.00
	2	0.317	673.00	0.362	474.00
	3	0.372	745.00	0.370	423.00
70	1	0.435	661.00	0.400	779.00
	2	0.422	679.00	0.424	708.00
	3	0.400	632.00	0.412	607.00
80	1	0.435	699.00	0.437	793.00
	2	0.433	608.00	0.435	611.00
	3	0.446	765.00	0.424	725.00
90	1	0.524	694.00	0.524	699.00
	2	0.505	720.00	0.505	769.00
	3	0.521	726.00	0.505	746.00
100	1	0.524	792.00	0.508	729.00
	2	0.524	687.00	0.543	705.00
	3	0.541	696.00	0.505	706.00

Table B3

Raw Calibration Data of the Vision Module for Velocity Estimation at h_{cam} = 0.705 m and $d_{e|thresh}$ = 100 px

Motor Setting	Trial	Crop		Weed	
		Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$	Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$
10	1	0.045	34.00	0.047	61.00
	2	0.045	70.00	0.047	88.00
	3	0.048	65.00	0.049	83.00
20	1	0.118	267.00	0.122	248.00
	2	0.123	200.00	0.120	300.00
	3	0.121	240.00	0.123	250.00
30	1	0.191	400.00	0.186	408.00
	2	0.190	336.00	0.190	394.00
	3	0.196	384.00	0.191	396.00
40	1	0.254	440.00	0.258	474.00
	2	0.258	528.00	0.258	455.00
	3	0.254	544.00	0.242	391.00
50	1	0.311	483.00	0.312	603.00
	2	0.305	579.00	0.304	617.00
	3	0.306	659.00	0.299	624.00
60	1	0.370	636.00	0.338	747.00
	2	0.353	676.00	0.372	759.00
	3	0.353	576.00	0.370	712.00
70	1	0.413	774.00	0.400	772.00
	2	0.424	808.00	0.435	728.00
	3	0.435	779.00	0.402	701.00
80	1	0.476	693.00	0.448	882.00
	2	0.461	830.00	0.459	831.00
	3	0.461	895.00	0.448	812.00
90	1	0.490	747.00	0.508	823.00
	2	0.505	709.00	0.505	817.00
	3	0.476	777.00	0.508	833.00
100	1	0.508	783.00	0.508	828.00
	2	0.524	759.00	0.490	775.00
	3	0.543	773.00	0.565	747.00

Table B4

Raw Calibration Data of the Vision Module for Velocity Estimation at h_{cam} = 0.730 m and $d_{e|thresh}$ = 100 px

Motor Setting	Trial	Crop		Weed	
		Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$	Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$
10	1	0.044	141.00	0.043	86.00
	2	0.043	85.00	0.043	98.00
	3	0.043	103.00	0.043	100.00
20	1	0.114	244.00	0.116	212.00
	2	0.116	272.00	0.115	246.00
	3	0.114	241.00	0.116	232.00
30	1	0.195	362.00	0.198	361.00
	2	0.191	410.00	0.193	364.00
	3	0.200	408.00	0.198	302.00
40	1	0.258	441.00	0.258	504.00
	2	0.263	500.00	0.272	256.00
	3	0.254	528.00	0.263	521.00
50	1	0.312	539.00	0.311	582.00
	2	0.305	593.00	0.324	592.00
	3	0.309	630.00	0.312	586.00
60	1	0.353	601.00	0.362	698.00
	2	0.362	694.00	0.353	765.00
	3	0.353	692.00	0.355	645.00
70	1	0.435	649.00	0.400	784.00
	2	0.422	757.00	0.391	782.00
	3	0.412	759.00	0.422	733.00
80	1	0.476	747.00	0.461	855.00
	2	0.448	804.00	0.433	812.00
	3	0.463	843.00	0.446	730.00
90	1	0.488	613.00	0.490	787.00
	2	0.508	676.00	0.474	734.00
	3	0.490	789.00	0.521	726.00
100	1	0.505	735.00	0.508	836.00
	2	0.524	826.00	0.543	813.00
	3	0.524	846.00	0.524	808.00

Table B5

Raw Calibration Data of the Vision Module for Velocity Estimation at h_{cam} = 0.705 m and $d_{e|thresh}$ = 50 px

Motor Setting	Trial	Crop		Weed	
		Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$	Actual Velocity, $\frac{m}{s}$	MVS Frame Velocity, $\frac{px}{s}$
10	1	0.053	0.053	0.072	129.00
	2	0.052	0.052	0.055	99.00
	3	0.053	0.053	0.067	120.00
20	1	0.120	0.120	0.143	255.00
	2	0.124	0.124	0.133	237.00
	3	0.124	0.124	0.162	290.00
30	1	0.191	0.191	0.208	372.00
	2	0.188	0.188	0.207	370.00
	3	0.192	0.192	0.188	336.00
40	1	0.246	0.246	0.203	362.00
	2	0.258	0.258	0.221	395.00
	3	0.262	0.262	0.222	396.00
50	1	0.311	0.311	0.205	366.00
	2	0.312	0.312	0.215	384.00
	3	0.311	0.311	0.224	400.00
60	1	0.353	0.353	0.228	408.00
	2	0.372	0.372	0.228	408.00
	3	0.372	0.372	0.233	416.00
70	1	0.410	0.410	0.239	427.00
	2	0.410	0.410	0.240	429.00
	3	0.412	0.412	0.240	429.00
80	1	0.461	0.461	0.251	449.00
	2	0.490	0.490	0.251	449.00
	3	0.476	0.476	0.251	449.00
90	1	0.524	0.524	0.251	449.00
	2	0.508	0.508	0.251	449.00
	3	0.474	0.474	0.251	449.00
100	1	0.524	0.524	0.250	447.00
	2	0.508	0.508	0.250	447.00
	3	0.508	0.508	0.250	447.00

Table B6

Raw Calibration Data of the Vision Module for Velocity Estimation at h_{cam} = 0.705 m and $d_{e|thresh}$ = 150 px

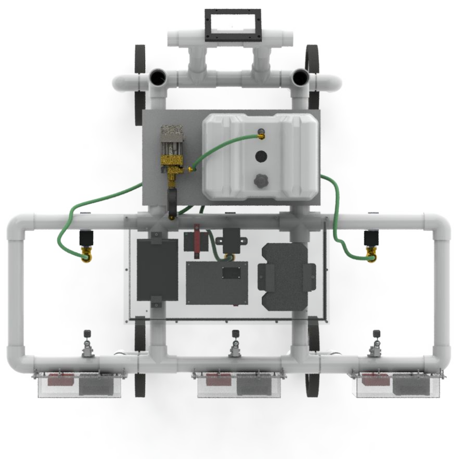
Motor Setting	**Trial**	**Crop**		**Weed**	
		Actual Velocity, $\frac{m}{s}$	**MVS Frame Velocity, $\frac{px}{s}$**	**Actual Velocity, $\frac{m}{s}$**	**MVS Frame Velocity, $\frac{px}{s}$**
10	1	0.054	148.00	0.047	84.00
	2	0.054	461.00	0.067	119.00
	3	0.052	131.00	0.086	153.00
20	1	0.123	272.00	0.163	291.00
	2	0.128	284.00	0.156	278.00
	3	0.127	268.00	0.142	254.00
30	1	0.198	419.00	0.216	386.00
	2	0.195	357.00	0.234	418.00
	3	0.198	414.00	0.207	369.00
40	1	0.262	476.00	0.231	413.00
	2	0.297	462.00	0.341	609.00
	3	0.304	498.00	0.314	560.00
50	1	0.312	582.00	0.357	638.00
	2	0.311	648.00	0.391	698.00
	3	0.311	611.00	0.409	730.00
60	1	0.380	793.00	0.424	758.00
	2	0.362	676.00	0.406	725.00
	3	0.355	685.00	0.401	717.00
70	1	0.391	774.00	0.464	828.00
	2	0.412	751.00	0.467	834.00
	3	0.400	813.00	0.352	628.00
80	1	0.463	840.00	0.103	184.00
	2	0.463	892.00	0.331	591.00
	3	0.410	895.00	0.503	898.00
90	1	0.505	863.00	0.534	953.00
	2	0.474	872.00	0.553	987.00
	3	0.490	559.00	0.557	995.00
100	1	0.541	972.00	0.554	989.00
	2	0.508	983.00	0.559	999.00
	3	0.524	956.00	0.584	1043.00

Appendix C

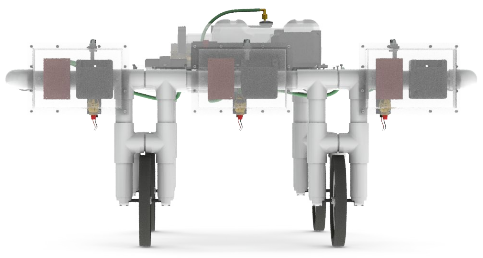
Precision Sprayer Components

Figure C1

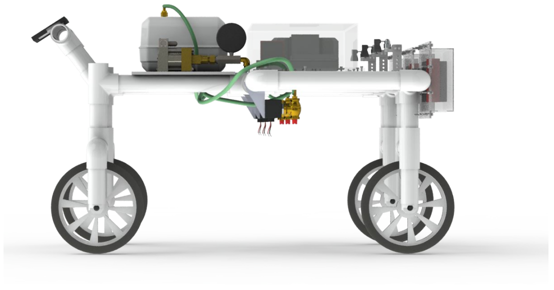
Multiview Projections of the MAPS Assembly



(a) *Top View*



(b) *Front View*



(c) *Right-Side View*

Figure C2

Vision Compute Unit and Valve Control Assembly

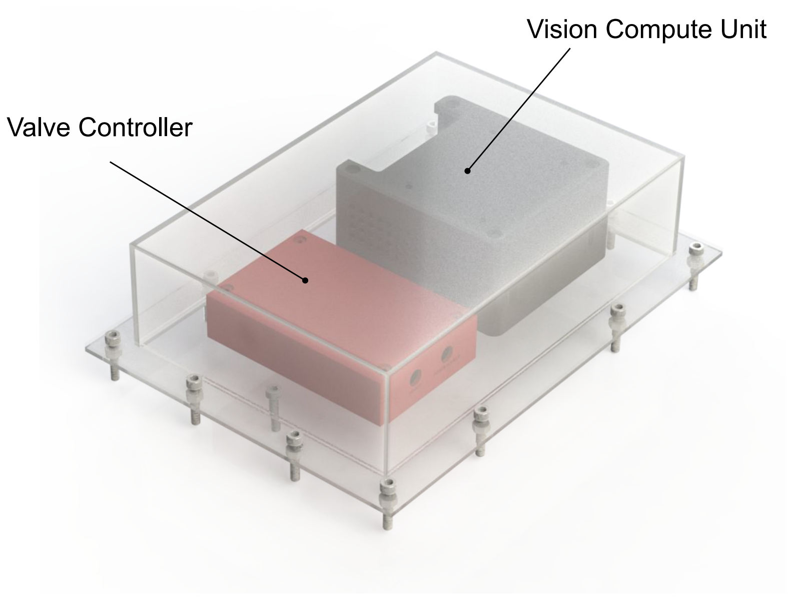


Figure C3

Central and Power Modules Assembly

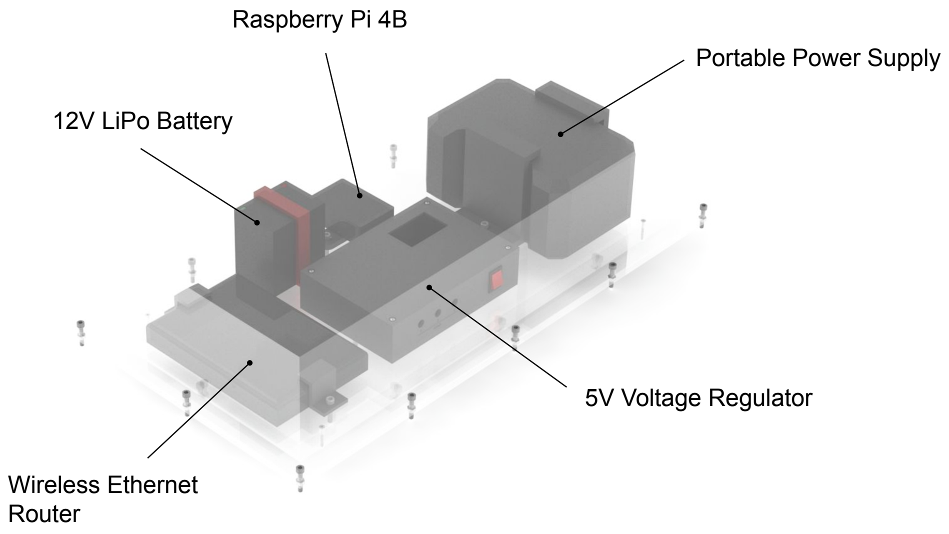


Figure C4

Solenoid Valve and Spray Nozzle Assembly

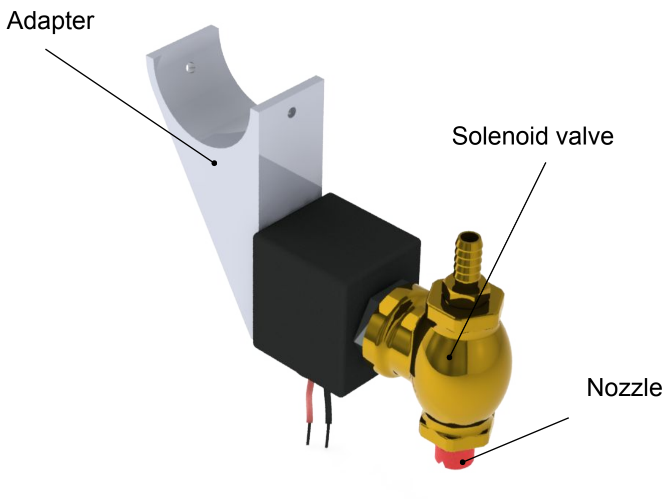


Figure C5

Camera Assembly

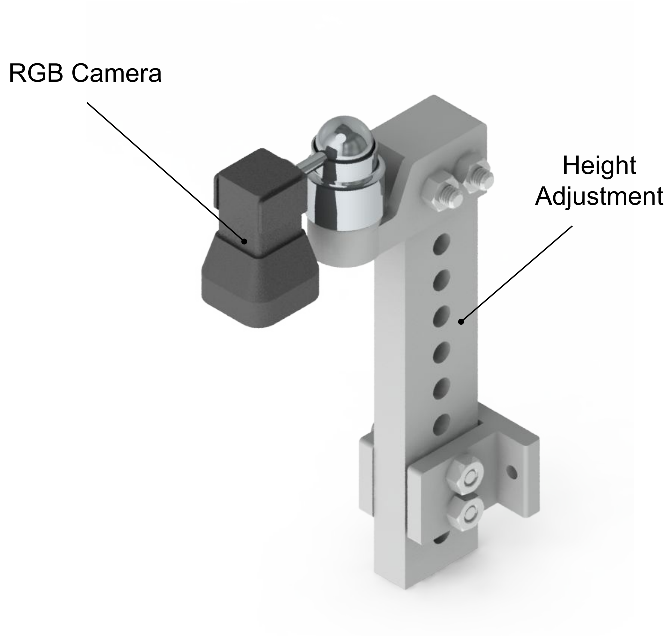


Figure C6

Frame Module

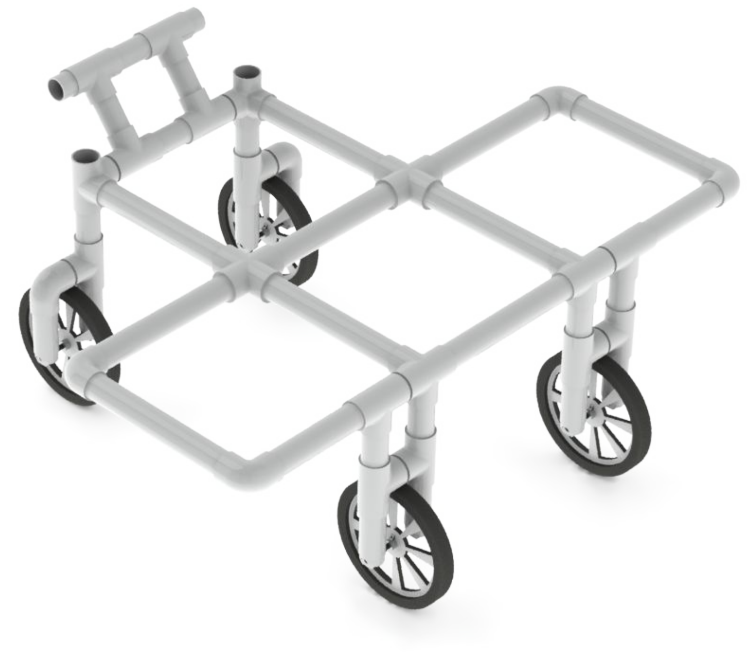


Table C1

Cost of Each Module of the Modular Precision Sprayer

Module	Cost per Module	Unit	Cost	% Cost
3 Vision Modules	261.73	3	$785.19	37.57%
3 Sprayer Modules	103.54	3	$310.61	14.86%
Power Module	339.18	1	$339.18	16.23%
Central Module	421.11	1	$421.11	20.15%
Other Components (tank, pump, fasteners, paint)	233.77	1	$233.77	11.19%
Total Cost			**$2,089.86**	**100.00%**

Table C2

Detailed Component and Part Lists of the Modular Agrochemical Precision Sprayer

Module	Part	Description	Quantity	Unit	Price, USD
Vision Module	Logitech StreamCam	RGB camera for Image Capture	1	pc.	169.99
	Nvidia Jetson Nano Developer Kit	Deep learning computation hardware	1	pc.	84.99
	5V DC 40 mm x 40 mm PWM Fan	Cooling	1	pc.	6.75
Sprayer Module	Arduino Nano (Atmega328P)	Microcontroller	1	pc.	10.99
	Arduino Nano Screw Terminal Shield	Connecting wires	1	pc.	2.93
	5VDC Arcelli Relay Module	Actuating solenoid valve	1	pc.	1.60
	12VDC 3/8" Solenoid Valve (US Solid USS2-00006)	Controlling spray application	1	pc.	31.49
	Fan-type Nozzle (Solo 4900654-P)	Delivery of chemical to plant	1	pc.	15.99
	3/8" Barb x 3/8" NPT Male Pipe Brass	Connecting nozzle to hoze	1	pc.	5.00
	3/8" Heavy Duty Reinforced Vinyl Hose	Connecting nozzle to pump	2.5	m	7.00
	3/8"-5/8" Hose Clamp	Fastening the hoses	2	pc.	0.80

Module	Part	Description	Quantity	Unit	Price, USD
	3/8" NPT Brass Quick Disconnect Hydraulic Coupler Set	Quick release of sprayer module	1	pc.	22.75
	3 ways Connector 3/8" Male x 3/8" Female x 3/8" Male NPT	Dividing liquid flow	1	pc.	5.00
Power Module	Wire Terminals	Wire connection	1	box	38.99
	1 to 3 Power splitter	Additional outlet	1	pc.	15.99
	12V 8Ah Deep Cycle LiFePO4 Battery	Spot Sprayer Power Supply	1	pc.	39.99
	296Wh Portable Power Supply	Main power source	1	pc.	189.99
	1/2" Wire Organizer	Organizing wires	3	m	7.63
	6-way 12V Blade Fuse Block Set	Circuit Protection	1	set	15.99
	Variable Buck Module	Voltage regulation	1	pc.	13.99
	16 AWG Stranded Copper Wire	Wiring	1	roll	16.61

Module	Part	Description	Quantity	Unit	Price, USD
Central Module	Raspberry Pi 4GB Kit	Central compute module	1	set	119.99
	450 Mb WiFi Router (TP-Link)	Router	1	pc.	24.99
	Wireless Keyboard	Input Device	1	pc.	21.19
	10 ft Ethernet Cable	Communication	1	roll	8.79
Frame	12 x 175 Lawn Mower Wheel	Soil and frame contact	4	pcs.	89.04
	1/2" Collar	Support for wheel and shaft	8	pcs.	26.50
	2" 4-way PVC Fitting	pipe connection	6	pcs.	65.94
	#7 1-5/8" Double Lock Thread Screw	Fastener	1	box	13.68
	15" x 30" x 1/8" Acrylic Sheet	Casing	2	Sheets	56.56
	Acrylic Plastic Cement	Fastening acrylic	1	can	10.99
	1/2" x 12" Aluminum Rod	Wheel Shaft	4	pcs.	21.56
	4’x8’ x 1/2" Plywood	Support for tank	1	Sheet	39.98
	1-1/2" x 8’ PVC Pipe Sch. 40	Frame	1	pc.	6.56
	2" x 8’ PVC Pipe Sch. 40	Frame	4	pcs.	35.36
	2" Tee PVC Pipe Fitting Sch. 40	Frame	4	pcs.	9.80
	2" 90° Elbow PVC Pipe Fitting Sch. 40	Frame	8	pcs.	19.44

Module	Part	Description	Quantity	Unit	Price, USD
	2" Cross PVC Pipe Fitting Sch. 40	Frame	2	pcs.	25.70
Others	NorthStar ATV Spot Sprayer - 10-Gallon Capacity, 1.1 GPM, 12 Volt	Chemical storage and pump	1	pc.	124.99
	1.75mm PLA Filament	Casing and electronics	3	rolls	56.97
	M2 Screws and Standoff Set	Fastening	1	Box	11.99
	M3 Screws and Standoff Set	Fastening	1	pc.	12.88
	White Spray Paint	Painting	6	cans	26.94