

Real time application of neural networks and hardware-accelerated image
processing pipeline for precise autonomous agricultural systems.

by

Jose Mateus Raitz Persch

B.S., Kansas State University, 2022

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Mike Wieggers Department of Electrical and Computer Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Co-Major Professor
Don M. Gruenbacher

Approved by:

Co-Major Professor
Ajay Sharda

Copyright

© Jose Mateus Raitz Persch 2024.

Abstract

The agricultural industry is undergoing a significant transformation as it increasingly adopts automation and precision technology to optimize crop management practices¹. In this context, this research focuses on developing an autonomous pesticide spraying rover that leverages advanced technologies to revolutionize precision agriculture. The primary objective of this project is to utilize a neural network for real-time aphid detection in sorghum crops, enabling the targeted application of pesticides only to infested plants. To achieve this goal, cutting-edge technologies and software frameworks, including ROS 2 (Robot Operating System)², have been integrated to create a sophisticated software system for the rover. One of the major hurdles in deploying neural networks for real-time feedback in the field is the limitations of wireless communication. Transmitting large amounts of high-resolution images can be unreliable and slow, making it impractical to deploy the neural network on a remote server. To ensure real-time processing and feedback for the sprayer, the neural network is deployed directly on the rover's computing system. For this purpose, Nvidia's Jetson AGX Orin platform has been chosen, which combines an ARM processor with a powerful GPU. This setup ensures exceptional model inference performance compared to CPU-based solutions, enabling real-time processing and feedback for the sprayer. In addition to deploying the neural network on the rover, this research also focuses on efficiently handling image frames captured by the cameras. The GPU in the Jetson platform is utilized to accelerate the image pipeline and leverage the hardware encoder to compress image frames to H.264 format, resulting in streamlined data recording. This comprehensive approach enhances the rover's capabilities while ensuring energy efficiency, a critical factor for field operations. The entire system is seamlessly integrated into ROS 2, and benefits from Nvidia's Isaac ROS packages, which provide GPU-accelerated ROS nodes. The use of hardware acceleration is a pivotal component of this research, as it enables substantial computational power while

maintaining efficiency. Furthermore, a method has been implemented to estimate the depth from the camera to the aphid from a 2D aphid detection. This depth estimation is subsequently used to determine the 3D coordinates of the aphid in the world, further enhancing the precision of the pesticide application. By offloading intensive processes from the CPU to the GPU and other accelerators, it is ensured that the autonomous pesticide spraying rover can operate effectively and deliver precise results, making it a valuable asset in the quest for optimized and sustainable agriculture. This thesis will provide an in-depth look at this innovative approach and showcase the practical implications of real-time neural network deployment in precision agriculture.

Table of Contents

List of Figures	viii
List of Tables	x
Acknowledgements	xi
Dedication	xii
1 Chapter 1 - Introduction	1
1.1 Introduction	1
1.2 Contributions	3
2 Chapter 2 - Robotics System Overview	5
2.1 Robot Description	5
2.2 Computing Hardware Selection	8
2.3 Camera Selection	10
3 Chapter 3 - Aphid Detection and Sprayer Manager System	12
3.1 Software Architecture	12
3.2 Robot Description	14
3.3 Coordinate Frames	16
3.4 Robot Localization	18
3.5 Intrinsic Camera Calibration	21
3.5.1 Intrinsic Parameters	21
3.5.2 Importance in Projections	21

3.5.3	Calibration Process	22
3.6	Hardware Acceleration	23
3.6.1	Definition	23
3.6.2	Advantages	24
3.6.3	Examples	25
3.6.4	Application in Robotics and Aphid Detection	26
3.7	Efficient Hardware Acceleration	27
3.8	Neural Network Architecture and Deployment	29
3.9	2D Detection to 3D Coordinates Estimation	32
3.10	Sprayer Manager Node	40
3.11	Hardware Accelerated Image Processing and Neural Network Inference pipeline	43
3.11.1	Pipeline Overview	43
3.11.2	Hardware Accelerated Nodes	44
3.11.3	Post-Processing and Control	45
3.11.4	Micro-ROS Integration	45
3.11.5	Hardware Accelerated Pipeline Overview	46
3.12	Aphid Depth Estimation from 2D Detection Validation Experiment Setup . .	46
3.12.1	Image Preparation	46
3.12.2	Experimental Conditions	47
3.12.3	Data Collection	48
3.12.4	Environmental Factors	48
3.12.5	Camera Calibration	48
3.13	Aphid Depth Estimation from 2D Detection Validation Results	52
3.13.1	Data Processing Methodology	52
3.13.2	Results Presentation	53
3.14	Results analysis	57
4	Chapter 4 - Conclusions	59

4.1	Summary	59
4.2	Future Work	61
	Bibliography	63

List of Figures

2.1	Robotic Platform	6
2.2	Rover Assembly	7
2.3	Agricultural robot platform with labeled components. Note: The camera shown is an earlier version and does not represent the current system described in this chapter.	8
2.4	AAEON Boxer 8640AI, the industrial-grade fanless computer housing the Nvidia Jetson AGX Orin used for field deployment. (Source: AAEON) . . .	9
2.5	Luxonis OAK-FFC 4P PoE development kit. (Source: Luxonis)	10
2.6	Arducam AR0234 global shutter camera module. (Source: Luxonis)	11
2.7	System diagram of the robot’s computer vision system.	11
3.1	Simplified Software Architecture	14
3.2	Visualization of the robot’s URDF model in Foxglove Studio, showing the detailed mesh representation of the agricultural robot.	15
3.3	Robot Coordinate Transform Tree	18
3.4	Robot Localization Illustration	20
3.5	Intrinsic Camera Calibration in ROS 2	23
3.6	Example of a hardware accelerated image pipeline. (Source NVIDIA Isaac ROS)	27
3.7	Comparison of inefficient and efficient hardware acceleration strategies. (Source NVIDIA Isaac ROS)	29
3.8	Aphid segmentation results using U-Net. Left: Original image. Right: Segmented output highlighting aphid infestations.	31

3.9	RT-DETR inference results showing aphid detections on an infested sorghum plant, processed in real-time on the Jetson AGX Orin.	32
3.10	Visualization of 3D aphid localization. Detected aphids are represented as markers in the 3D space relative to the robot and plant positions. The spread of estimated aphid positions in the z-axis (depth) is visible, illustrating the discrepancies in depth estimation.	37
3.11	Aphid detection and 3D localization with the plant at the left edge of the camera frame. The image demonstrates how the 2D detections correspond to 3D localizations when the plant is at the left side of the field of view.	38
3.12	Aphid detection and 3D localization with the plant in the center of the camera frame. This image shows the correspondence between 2D detections and 3D localizations when the plant is directly in front of the camera.	39
3.13	Aphid detection and 3D localization with the plant at the right edge of the camera frame. This image illustrates how the system performs 2D detection and 3D localization as the plant moves to the right side of the field of view.	40
3.14	Data flow diagram from RT-DETR Decoder to Sprayer Manager and micro-ROS Sprayer node.	43
3.15	Hardware Accelerated Image Pipeline with NVIDIA ISAAC ROS Nodes	46
3.16	Experimental setup viewed from the camera at a flat (0-degree) angle.	49
3.17	Experimental setup viewed from the camera at a 40-degree upward angle.	49
3.18	External view of the experimental setup with the camera at a flat (0-degree) angle.	50
3.19	External view of the experimental setup with the camera at a 40-degree upward angle.	51
3.20	Comparison of Absolute Errors Across All Configurations (Flat Camera Angle)	56
3.21	Comparison of Absolute Errors Across All Configurations (Angled Camera)	57

List of Tables

3.1	Depth Estimation Results for Flat Camera Angle (0 degrees)	54
3.2	Depth Estimation Results for Angled Camera (40 degrees upwards)	55

Acknowledgments

The author would like to thank:

- His Lord and Savior Jesus Christ, for being present throughout his academic journey as a graduate student abroad.
- His wife Tamara, for her unwavering support and belief in him throughout this endeavor.
- His parents, brothers, sister, and grandmother Romani, for their love and encouragement from afar.
- Dr. Don Gruenbacher, for his constant assistance and guidance during his academic pursuits.
- Dr. Ajay Sharda, for providing the opportunity to be a part of Farms-Lab.
- Dr. Dan Flippo, for his valuable mentorship and insights.
- Garrett Peterson, for his instrumental role as a mentor and knowledge-sharing throughout his time at K-State.
- His dedicated team members Nirajan Piya, Rahul Harsha Cheppally, Ketan Shende, and Aayush Rai for their hard work, collaboration, and mutual support.
- Dr. Ivan Grijalva, for growing aphid plants and infesting them to facilitate this project, in conjunction with Dr. Brian McCornack, who provided the essential datasets needed for this research.

Dedication

I dedicate my work to the glory of God, and to my beloved wife Tamara and our daughter Olivia.

Chapter 1

Chapter 1 - Introduction

1.1 Introduction

Modern agriculture faces significant challenges in optimizing crop management and reducing pesticide usage. Current methods of pest control often result in over-spraying, leading to environmental concerns and economic losses. The need for precision agriculture solutions that can accurately detect pests and apply targeted treatments is pressing, particularly in the case of aphid infestations in crops like Sorghum.

Aphids are notorious pests that can cause significant damage to crops, and their timely detection is crucial for effective pest management. However, detecting aphids in real-time poses significant technical challenges. Traditional approaches to neural network deployment, which are commonly used for image-based pest detection, are limited by wireless communication constraints. These limitations hinder the development of autonomous systems that can detect pests and apply targeted treatments in real-time.

Several researchers have attempted to address these challenges. Li et al.³ proposed a method for pest detection and positioning based on binocular stereo vision to guide a robot for automatic pesticide spraying in greenhouses. Their approach utilized color feature extraction and image segmentation to identify pests, combined with binocular stereo vision techniques to obtain 3D positioning of the pests. While innovative, this method relied on a

stereo camera setup, which can be more complex and expensive than monocular systems. In contrast, the approach proposed in this thesis achieves 3D localization using a single camera and monocular depth estimation techniques, potentially offering a more cost-effective and streamlined solution for field deployment.

In the realm of autonomous agricultural robots, Oberti et al.⁴ presented a ground robot for selective spraying in grapevines. Their system used multispectral imaging for disease detection but did not incorporate real-time neural network inference. The approach presented in this thesis advances this concept by integrating state-of-the-art neural networks for real-time pest detection.

Addressing these limitations, Fuentes et al.⁵ demonstrated the use of GPU-accelerated deep learning models for real-time plant disease detection. Their work showcased the potential of hardware acceleration but was limited to stationary systems. The research presented extends this concept to a mobile platform, leveraging the NVIDIA Jetson AGX Orin for both mobility and high-performance computing.

Similarly, Gao et al.⁶ presented a precision agricultural robotic sprayer with real-time tobacco recognition based on deep learning. Their system achieved impressive detection performance using a YOLOv5n model, with an F1-score of 87.2% at 67 frames per second. They also introduced a novel disturbance-based pressure and flow control method for improved spraying precision. While their work demonstrates significant progress in targeted spraying, their system relied on a laptop with a GPU for processing, which, although powerful, is not as suitable for field deployment as the Jetson AGX Orin-based solution proposed in this thesis. Furthermore, their implementation required custom software development for tasks such as coordinate transforms, which are efficiently handled by ROS 2's built-in libraries like TF2. The approach proposed in this thesis leverages the ROS 2 ecosystem, offering a more modular and potentially more maintainable solution for long-term field use.

The key innovations of this research include:

1. Real-time, on-device aphid detection using advanced neural networks optimized for embedded systems.
2. A novel approach to 3D localization of detected aphids using monocular depth estimation, eliminating the need for complex and expensive depth cameras.
- 3.

Utilization of hardware acceleration techniques, particularly leveraging the NVIDIA Jetson AGX Orin’s GPU and VPI (Vision Programming Interface) capabilities, to achieve high-performance image processing and neural network inference. 4. Integration of these components into a ROS 2-based system, taking advantage of NVIDIA Isaac ROS packages for efficient, hardware-accelerated image processing pipelines. 5. Development of a targeted spraying system that can apply pesticides precisely where aphids are detected, potentially reducing overall pesticide use.

This thesis provides a comprehensive look at the development, implementation, and testing of this autonomous aphid detection and spraying system. It explores the challenges faced, the solutions developed, and the potential impact of this technology on precision agriculture and sustainable farming practices.

1.2 Contributions

This research has made significant contributions to the development of an autonomous pesticide spraying rover for precision agriculture. The key contributions are:

The author successfully selected and tested cameras for aphid detection, ensuring optimal image quality for neural network processing. Furthermore, the intrinsic parameters of the cameras were calibrated to enable accurate 3D reconstruction.

A neural network was deployed and optimized directly on the rover’s computing system, enabling real-time aphid detection in Sorghum crops for targeted pesticide application. This achievement overcomes the limitations of wireless communication, ensuring reliable and efficient pest management.

The integration of cutting-edge technologies and software frameworks, including ROS 2 (Robot Operating System), has enabled the development of a robust and flexible autonomous system. This framework provides a foundation for future research and development in precision agriculture.

The author designed and developed novel software to estimate 3D coordinates of aphids from 2D defections, eliminating the need for complex depth cameras in autonomous pesticide

spraying systems. This innovation enables the use of monocular cameras, simplifying the system design and reducing costs. Additionally, a sprayer controller software was developed, enabling precise control over pesticide application.

These contributions collectively advance the state-of-the-art in autonomous precision agriculture, paving the way for more efficient, effective, and sustainable crop management practices. The author's work demonstrates a significant step towards developing a fully autonomous system capable of detecting and controlling pests in real-time, reducing the environmental impact of pesticides and improving crop yields.

Chapter 2

Chapter 2 - Robotics System

Overview

2.1 Robot Description

A four-wheel drive platform was developed with skid steering capabilities to navigate through row crops with 30-inch spacing while carrying attachments like sprayers. The modified design measures 60 inches long, 21 inches wide, and 58 inches tall. Independent brushless DC worm gear motors from ATO (ATO-GEAR-W450) power the wheels, each delivering a rated torque of 1.07 ft-lbs and featuring hall sensors to detect motor rotation. ODrive S1 motor controllers regulate the motors, supporting 12-48V operation at 40A continuous current and CAN bus control.

The rover's wheel-motor assembly was custom-designed and fabricated in-house. Two 24V, 60Ah Dakota Lithium-Ion batteries connected in parallel provide a total capacity of 120Ah, powering all sub-systems. The lightweight body is constructed from an extruded metal frame and aluminum sheeting, ensuring modularity. The frame supports a box housing the computing platform, cameras, tank, and sprayer boom, with lower-level control circuits integrated into the chassis.

The rover is equipped with hall encoders, an inertial measurement unit, and GPS to

determine its localized position. At its core lies an NVIDIA Orin processor (8-core ARM, 32GB RAM, GPU), providing real-time processing capabilities. ROS 2 framework ensures seamless communication among sub-systems.



Figure 2.1: *Robotic Platform*

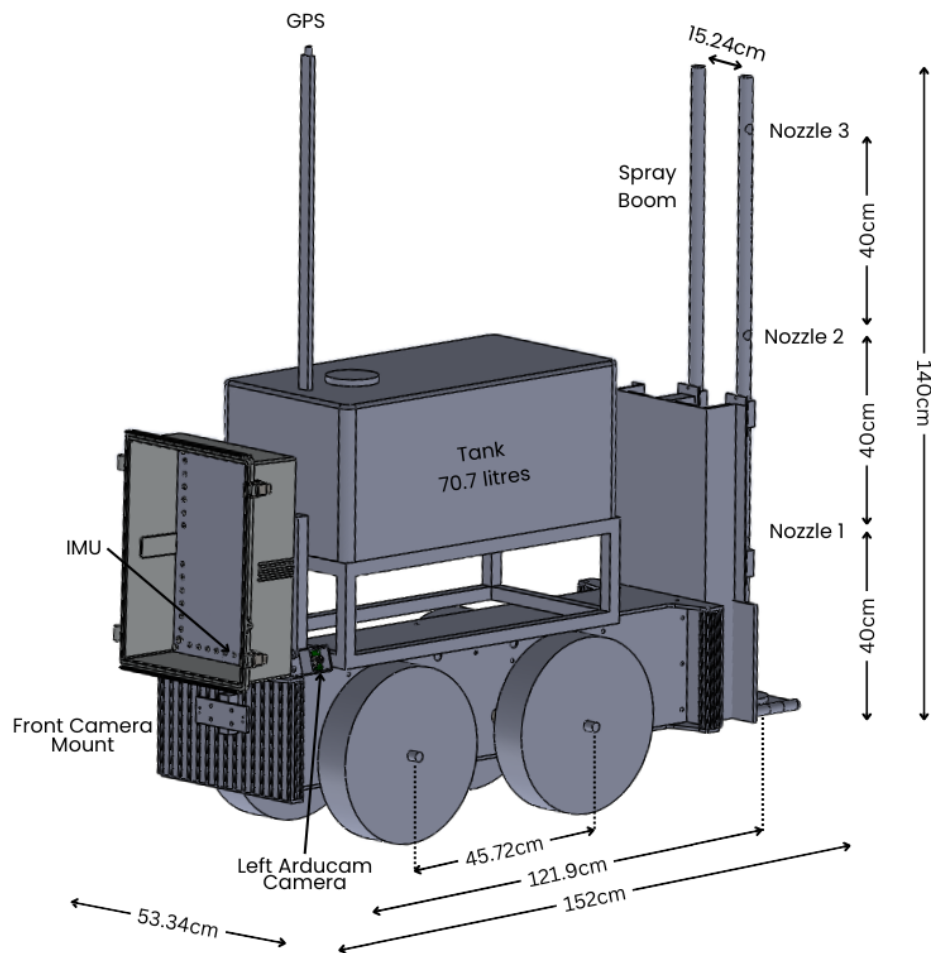


Figure 2.2: *Rover Assembly*

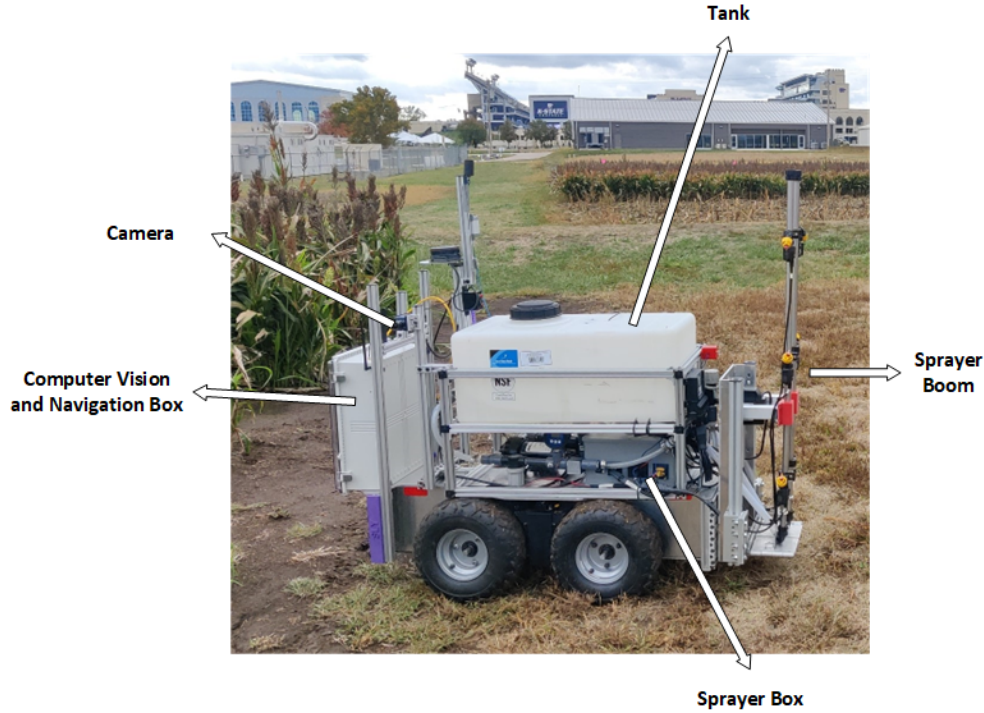


Figure 2.3: *Agricultural robot platform with labeled components. Note: The camera shown is an earlier version and does not represent the current system described in this chapter.*

2.2 Computing Hardware Selection

The primary choice for the robot’s computing hardware platform was the Nvidia Jetson AGX Orin. This decision was influenced by several factors, the most significant of which was that the AGX Orin was the latest and most powerful model in the Jetson line at the time of the design process. Nvidia Jetson has been, and continues to be, a popular platform for deploying neural networks in embedded systems such as the one in this project. It offers substantial computational power, which is crucial for the robot’s computer vision and machine learning systems. Moreover, Nvidia Jetson provides comprehensive documentation and examples, making it an accessible and user-friendly platform for applications that require hardware acceleration and neural network deployment. While platforms with an x86 CPU and a discrete Nvidia GPU may offer superior software support and documentation, they are not feasible for a battery-powered robot due to their power consumption and physical

size. The robot’s software system was developed using ROS 2, and Nvidia provides ROS 2 packages for hardware-accelerated image processing, neural network deployment, H264 image encoding, and more with the Jetson Orin. This was a significant advantage, as these ready-to-use ROS 2 nodes eliminated the need to develop these hardware-accelerated functionalities independently. Initially, the AGX Orin development kit was used, which boasts 275 Tops of AI Performance, a 2048-core NVIDIA Ampere architecture GPU with 64 Tensor Cores, and a 12-core Arm® Cortex®-A78AE v8.2 64-bit CPU. While the development kit typically comes with 64GB of RAM, the specific model used was equipped with a special 32GB RAM Jetson AGX Orin module. Later, the project switched to a third-party carrier board and fanless enclosure, the AAEON Boxer 8640AI, which is powered by the Nvidia Jetson AGX Orin with 32GB of RAM. Although this version has slightly less performance than the Development Kit, with 200 Tops of AI Performance, an 8-core CPU, a 1792-core NVIDIA Ampere architecture GPU with 56 Tensor Cores, and 32GB of RAM, it was necessary for field deployment. The industrial computer form factor and fanless design prevent dust from entering the computer’s fan, which is crucial for field operations.



Figure 2.4: *AAEON Boxer 8640AI, the industrial-grade fanless computer housing the Nvidia Jetson AGX Orin used for field deployment. (Source: AAEON)*

2.3 Camera Selection

The vision system utilizes an Arducam AR0234 global shutter camera module, which is connected to a Luxonis OAK-FFC 4P PoE development kit via a MIPI interface. The OAK-FFC 4P PoE is a camera interface development kit that allows for the connection of up to four camera modules via MIPI. It features two 2-lane MIPI connections and two 4-lane MIPI connections, and it is equipped with an Intel Movidius Myriad X VPU.

The choice of the OAK-FFC 4P PoE development kit was driven by the flexibility it offers in testing different camera sensors by simply swapping different Luxonis FFC camera modules. The decision to opt for the 4 MIPI ports version over the single port version was made to allow for potential upgrades, such as the addition of two cameras on each side of the rover for stereo vision depth perception. In such a scenario, four cameras could be connected to one development kit, with a single PoE connection to the Jetson. The Boxer 8640-ai, with its four PoE ports, can directly power the OAK-FFC 4P PoE.

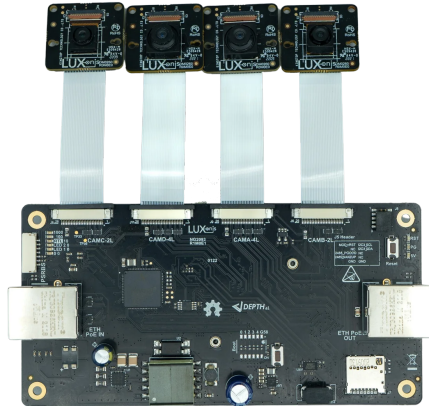


Figure 2.5: *Luxonis OAK-FFC 4P PoE development kit. (Source: Luxonis)*



Figure 2.6: *Arducam AR0234 global shutter camera module. (Source: Luxonis)*

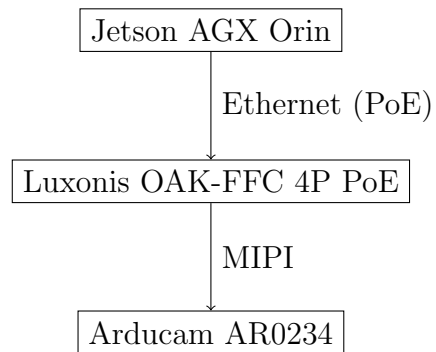


Figure 2.7: *System diagram of the robot's computer vision system.*

Chapter 3

Chapter 3 - Aphid Detection and Sprayer Manager System

3.1 Software Architecture

The autonomous agricultural system, designed to detect aphids in sorghum fields, leverages the benefits of ROS 2 (Robot Operating System)² as its software stack. ROS 2 provides an ideal framework for developing complex robotic systems like this one, which involves integrating multiple sensors, actuators, and processing nodes. The choice of ROS 2 is motivated by its benefits in facilitating the development and integration of complex software subsystems, particularly in robotics⁷. Some of the key advantages of ROS 2 include:

- **Modularity:** ROS 2 enables the system to be broken down into smaller, independent nodes that can be developed, tested, and maintained separately, promoting a more agile development process.
- **Flexibility:** With ROS 2, the system can easily switch between different hardware configurations or add new sensors and actuators as needed, without requiring significant changes to the underlying software architecture.
- **Reusability:** ROS 2 provides a rich set of tools and libraries that can be reused across

different robotic systems, reducing development time and effort.

- **Flexibility:** With ROS 2, the system can easily switch between different hardware configurations or add new sensors and actuators as needed, without requiring significant changes to the underlying software architecture.
- **Scalability:** As the system grows in complexity, ROS 2 allows it to distribute processing across multiple nodes, ensuring that the system remains responsive and efficient.

The basic pipeline for detecting aphids in sorghum fields consists of a series of ROS 2 nodes, each responsible for a specific task. The following nodes comprise the core image processing pipeline:

- **Camera Driver Node:** This node publishes raw RGB images and camera information topics, including camera intrinsics data.
- **Image Rectification Node:** Subscribing to both raw RGB images and camera info, this node rectifies the image for lens distortion, publishing the rectified image and updated camera info.
- **Image Resize Node:** This node resizes the rectified image to the size required by the neural network, publishing the resized image with corresponding camera info.
- **Inference Node(s):** One or more nodes perform inference on the resized image to detect aphids. This may involve pre-processing steps such as normalizing the image, converting it to a tensor, and decoding model output (e.g., segmentation mask or detection array).
- **Detection to Pose Node:** This node estimates the aphid localization with respect to the world frame, using the output from the inference node(s).
- **Sprayer Manager Node:** Receiving the estimated aphid location, this node decides when to spray based on the robot's sprayer approaching the aphid location.

- **micro-ROS Sprayer Node:** This node runs on an STM32 microcontroller and receives commands from the Sprayer Manager Node to control the physical spraying mechanism. It interfaces directly with the hardware to activate or deactivate the sprayer nozzles as indicated in Section 3.10.

A simplified diagram of the software pipeline is shown below:

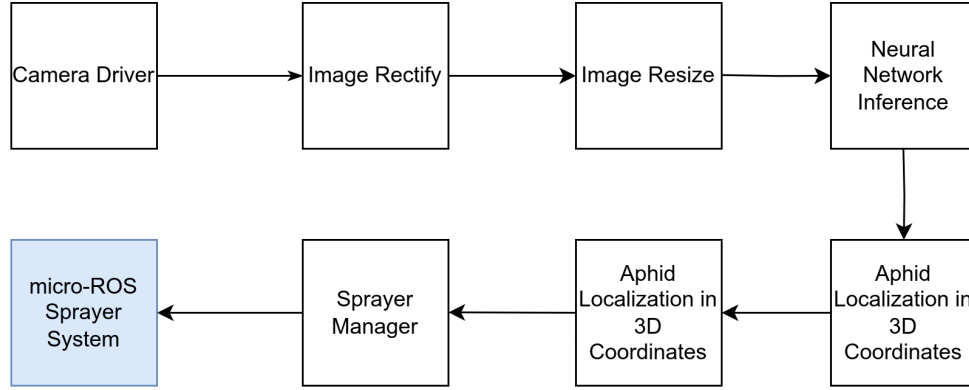


Figure 3.1: *Simplified Software Architecture*

In subsequent sections, the specific details of each node and the neural network architecture used for aphid detection will be discussed in more detail.

3.2 Robot Description

To accurately represent the robot’s physical structure and properties in the ROS 2 environment, the Unified Robot Description Format (URDF) is utilized. URDF is an XML-based format that describes a robot’s physical characteristics, including its kinematic and dynamic properties, visual representations, and collision models.

For this agricultural robot, a URDF file was exported from the CAD model created in SolidWorks using the ROS SolidWorks URDF Exporter plugin. This approach ensures that the ROS 2 representation of the robot accurately reflects its real-world counterpart, which is crucial for tasks such as motion planning, visualization, and simulation.

The URDF file defines the robot's link structure, joint connections, and geometric properties. This information is essential for the proper functioning of various ROS 2 components, including the Transform Library (TF2) and robot localization systems.

Figure 3.2 provides a visual representation of the robot's URDF model as rendered in Foxglove Studio. This visualization demonstrates how the URDF accurately captures the robot's structure and geometry, allowing for precise spatial reasoning and transformation computations within the ROS 2 ecosystem.

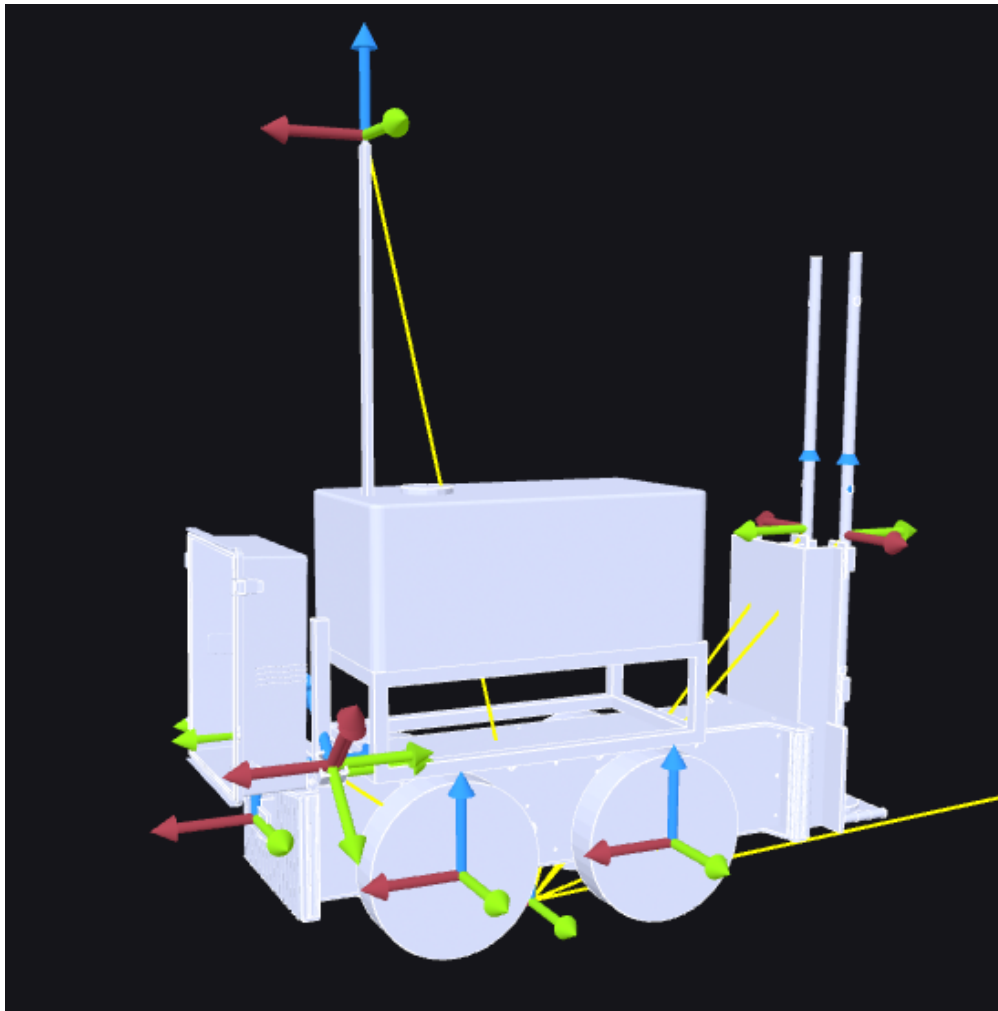


Figure 3.2: *Visualization of the robot's URDF model in Foxglove Studio, showing the detailed mesh representation of the agricultural robot.*

This graphical output not only serves as a visual confirmation of the URDF's accuracy but also aids in debugging and verifying the robot's configuration within the ROS 2 environment.

It allows for visual inspection of the robot’s structure, joint placements, and overall geometry, ensuring that all components are correctly positioned and scaled.

3.3 Coordinate Frames

In order to accurately localize aphids in the field, it is essential to establish a consistent coordinate frame for the robot and its environment. ROS 2 provides a powerful tool called TF2 (Transform Library) for managing coordinate frames and transformations between them.

The robot’s coordinate frame tree is shown in Figure 3.3. Below is a description of each frame in the tree:

- **base_link**: The root frame of the robot, typically located at the center of the robot’s base.
- **chassis_link**: Represents the main body of the robot.
- **wheel[0-3]_link**: Four frames representing each of the robot’s wheels. These are crucial for odometry calculations.
- **imu_link**: Frame for the Inertial Measurement Unit (IMU), used for orientation and acceleration data.
- **gps_link**: Frame for the GPS sensor, used for global positioning.
- **[left-right]_sprayer_link**: Represents the sprayer attachment on the robot.
- **camera_mount**: Frame for the navigation camera (not relevant in this study).
- **left_camera_mount_link**: Represents the mounting point for the aphid detection camera, connected to the robot via a revolute joint for angle adjustment.
- **left_camera_link**: Represents the physical body of the aphid detection camera.
- **left_camera_optical_link**: Represents the optical center of the aphid detection camera, with its coordinate system aligned to the camera’s view.

- **odom**: A frame representing the robot’s position relative to its starting point, based on local odometry.
- **map**: A global reference frame, typically used with GPS data for global localization.

These frames and their relationships, as defined in the URDF, allow for precise tracking of the robot’s components and sensors in 3D space.

It is important to note the relationships and differences between the camera-related frames:

- **left_camera_mount_link** is connected to the robot’s body via a revolute joint, allowing for adjustment of the camera angle. This flexibility is crucial for optimizing the field of view for aphid detection in various crop configurations.
- **left_camera_link** is a child of the `left_camera_mount_link` and represents the physical body of the camera. Its orientation typically aligns with the mount’s coordinate system.
- **left_camera_optical_link** is a child of the `left_camera_link` and represents the optical center of the camera. In this frame, the Z-axis points along the camera’s viewing direction, the X-axis points to the right in the image, and the Y-axis points down. This frame is crucial for computer vision operations as it aligns with the camera’s perspective.

The transformation chain from `left_camera_mount_link` to `left_camera_optical_link` accounts for both the adjustable mounting angle and any rotational offset between the camera’s body orientation and its optical axis. This setup allows for precise 3D perception and accurate projection of points from the camera’s image plane into the robot’s 3D world space, which is essential for localizing aphids in the field.

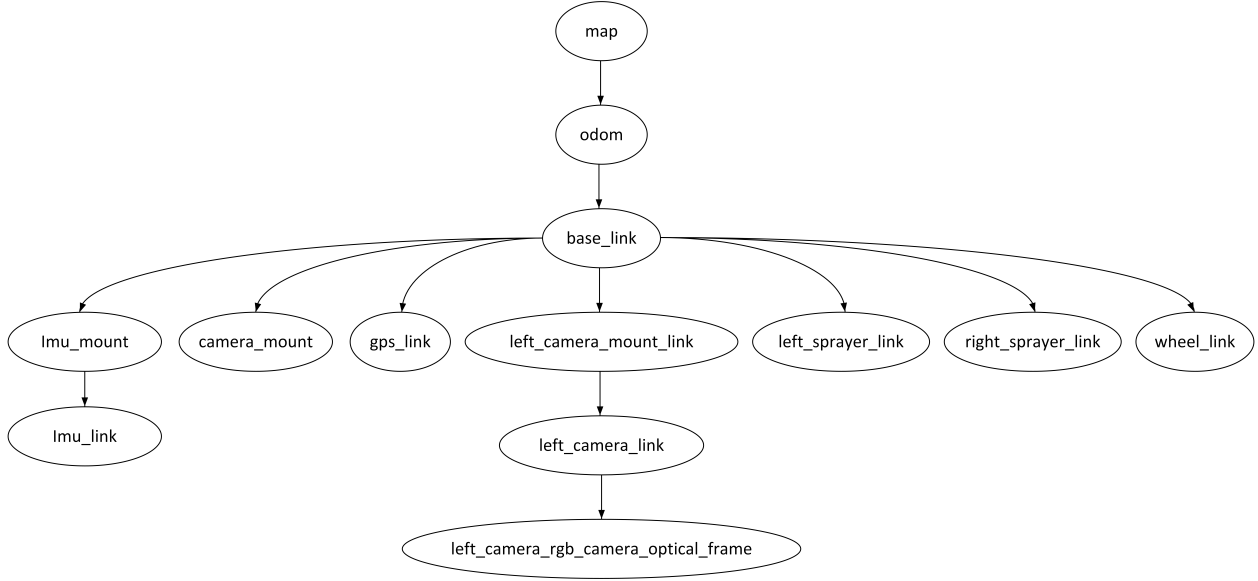


Figure 3.3: *Robot Coordinate Transform Tree*

3.4 Robot Localization

In order to obtain an accurate odometry source for the robot, it is necessary to fuse data from different sensors. Individual sensors may not provide accurate odometry data, or the error in the odometry data may accumulate over time.

The robot localization package provides an Extended Kalman Filter as well as an Unscented Kalman filter for fusing odometry data from multiple sources, such as wheel encoders, IMU, and GPS, to estimate the robot's pose in the world frame.

A pose, in robotics, refers to the combination of position and orientation of an object in 3D space. It typically consists of six degrees of freedom: three for position (x , y , z coordinates) and three for orientation (roll, pitch, yaw angles).

The robot localization node subscribes to the wheel encoder data and IMU data, and fuses them using a Kalman filter to estimate the robot's pose in the world frame. Two instances of the Extended Kalman Filter are used for the navigation system of the robot:

1. One fuses wheel encoder data with IMU, which provides a local odometry source.
2. Another fuses the aforementioned local odometry with GPS odometry, providing a global

odometry source.

Although for this application, only the local odometry is used, and the computer vision-based sprayer system was tested with the local odometry source in an indoor location where there was no GPS signal. The two Extended Kalman Filters also create coordinate frames called "odom" for the local odometry and "map" for the global odometry.

This approach to robot localization is based on the implementation described in "A Generalized Extended Kalman Filter Implementation for the Robot Operating System"⁸.

In ROS, the odom frame and map frame serve different purposes in robot navigation. The odom frame, derived from the robot's internal sensors, provides a continuous, high-frequency estimate of the robot's position relative to its starting point. It is ideal for short-term localization and local navigation tasks, but is prone to drift over time. On the other hand, the map frame represents a fixed, global coordinate system, typically aligned with a known map of the environment. It offers a stable, long-term reference frame, useful for global localization and navigation across multiple runs or robots. While the odom frame excels in providing smooth updates for immediate motion planning and obstacle avoidance, the map frame is crucial for consistent long-term navigation and positioning within a known environment. In practice, ROS often utilizes both frames in tandem, with the odom frame handling local, short-term movements and the map frame providing global context and drift correction.

In this application, the odometry (odom) frame was selected as the world reference due to its superior suitability for precise, real-time robotic interventions in agricultural settings. While high-precision GPS solutions like RTK (Real-Time Kinematic) can significantly improve global localization accuracy, the odom frame still offers distinct advantages for this specific task. The odom frame provides high-frequency, locally consistent position updates based on the robot's internal sensors, which are crucial for detecting and responding to small targets like aphids with minimal latency. Even with RTK corrections, GPS updates typically occur at lower frequencies (often 1-10 Hz) compared to odometry updates (which can exceed 100 Hz), making the odom frame more suitable for real-time control and rapid decision-making. Additionally, the odom frame is immune to GPS signal interruptions that

can occur even with RTK systems, such as when operating under dense foliage or near structures. The map frame, while globally more accurate over time, often incorporates sensor fusion algorithms (e.g., Kalman filters) that can introduce smoothing effects and small, sudden adjustments as new sensor data is integrated. These adjustments, while beneficial for long-term accuracy, can create discontinuities that are undesirable for precise, short-term operations like targeted spraying. In contrast, the odom frame provides a continuous and smooth trajectory, ensuring consistent tracking of the robot's position relative to recently detected infestations. This local consistency is paramount for the immediate task of identifying and treating aphid infestations in the robot's vicinity, where relative positioning is more critical than absolute global positioning. Furthermore, the odom frame supports faster, more deterministic real-time transformations, which is essential for responsive spraying actions. While the odom frame may accumulate drift over extended periods, this drift is typically small over short durations or within confined areas, localized operations involved in this precision agriculture application. Therefore, even with the availability of high-precision RTK GPS, the odom frame remains the optimal choice for maintaining the necessary accuracy, responsiveness, and consistency required for this targeted aphid control system. A simplified diagram of the robot localization and navigation system is shown below:

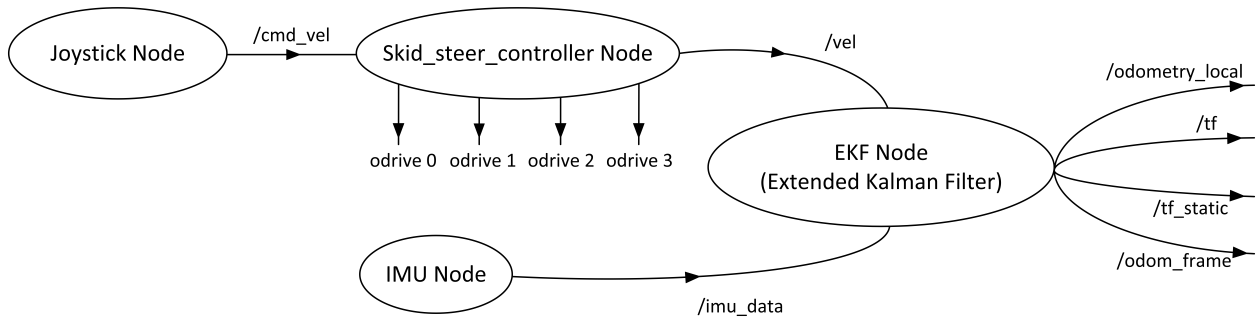


Figure 3.4: *Robot Localization Illustration*

3.5 Intrinsic Camera Calibration

Camera intrinsic calibration is a crucial process in computer vision and robotics applications, particularly when accurate 3D measurements or projections are required. The intrinsic parameters of a camera describe its internal characteristics, which are essential for mapping 3D world points to 2D image points using the pinhole camera model⁹.

3.5.1 Intrinsic Parameters

The intrinsic parameters of a camera typically include:

- Focal length (f_x, f_y): Represents the distance between the camera center and the image plane.
- Principal point (c_x, c_y): The intersection of the camera's optical axis with the image plane.
- Skew coefficient: Describes the angle between the x and y pixel axes (usually assumed to be zero for most modern cameras).
- Distortion coefficients: Account for radial and tangential distortions caused by the camera lens.

These parameters are crucial for accurately projecting 3D points onto the 2D image plane using the pinhole camera model. The pinhole model assumes that light rays pass through a single point (the pinhole) before hitting the image plane, creating a perspective projection. The intrinsic parameters allow us to mathematically model this projection process.

3.5.2 Importance in Projections

Accurate intrinsic parameters are essential for:

- 3D reconstruction: Converting 2D image coordinates to 3D world coordinates.

- Augmented reality: Overlaying virtual objects onto real-world images.
- Visual odometry: Estimating the camera’s motion from image sequences.
- Object detection and localization: Determining the real-world position and size of detected objects.

In the context of this project’s aphid detection system, precise intrinsic calibration is vital for accurately estimating the 3D position of detected aphids from their 2D image coordinates.

3.5.3 Calibration Process

For this project, the ROS 2 camera calibration package was utilized to calibrate the camera. This package is based on the widely-used OpenCV calibration routines and provides a user-friendly interface for the calibration. The process involved using a checkerboard pattern of known dimensions as a calibration target. The ROS 2 camera calibration node was launched, and the checkerboard was moved to cover different areas of the image at various angles until sufficient samples were collected. Once enough samples were gathered, the calibration parameters were computed. The resulting calibration data was then saved and subsequently used in the `camera_info` topic for the ROS 2 system, ensuring accurate intrinsic parameters for further image processing and 3D localization tasks.

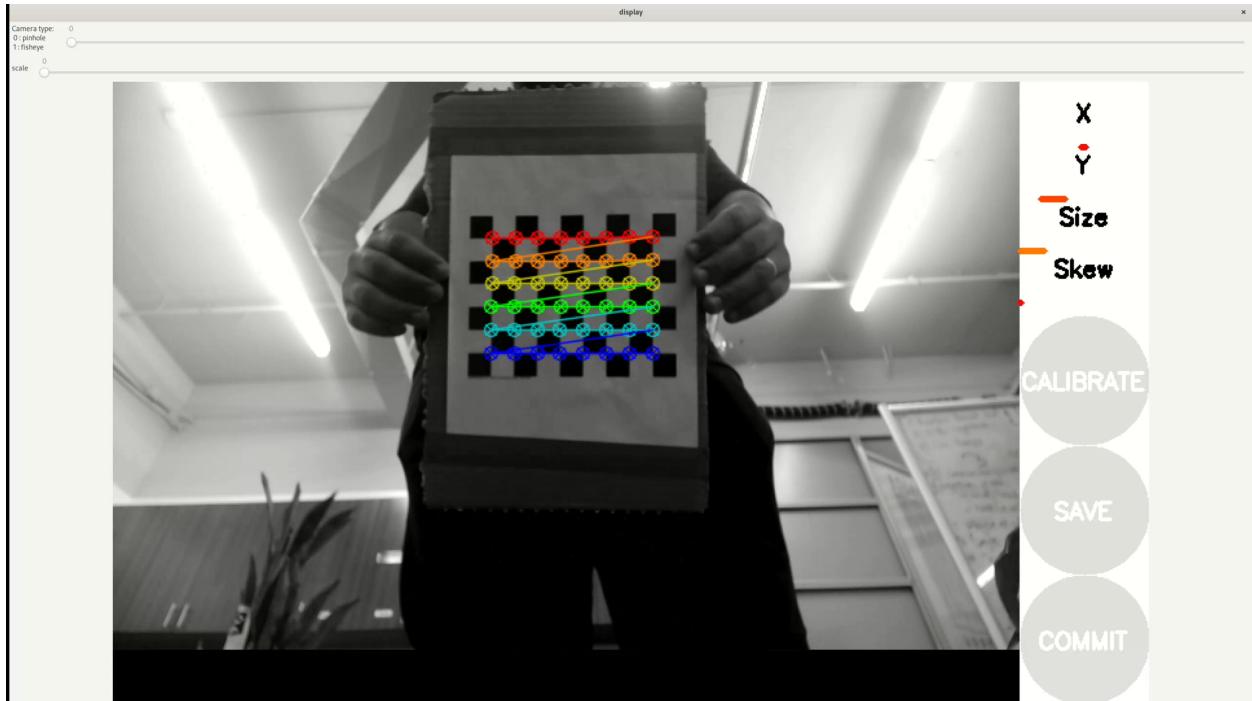


Figure 3.5: *Intrinsic Camera Calibration in ROS 2*

3.6 Hardware Acceleration

Hardware acceleration refers to the use of specialized hardware components to perform certain computational tasks more efficiently than is possible in software running on general-purpose CPU (Central Processing Unit) cores. This approach leverages the principle that specialized circuits can execute specific algorithms or operations much faster and more energy-efficiently than general-purpose processors.

3.6.1 Definition

Hardware acceleration involves offloading computationally intensive tasks from the CPU to dedicated hardware designed to perform these tasks optimally. This hardware, often referred to as an accelerator, is architected to execute specific types of computations with high efficiency and parallelism. Common types of hardware accelerators include Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), Application-Specific Integrated

Circuits (ASICs), and more recently, specialized AI accelerators like Tensor Processing Units (TPUs).

3.6.2 Advantages

Hardware acceleration offers several key advantages:

1. **Performance Improvement:** Hardware accelerators can perform certain tasks orders of magnitude faster than general-purpose CPUs, significantly reducing processing time for complex operations.
2. **Energy Efficiency:** Specialized hardware often requires less energy per operation compared to general-purpose processors, leading to lower power consumption and heat generation.
3. **Parallelism:** Many accelerators are designed with massive parallelism in mind, allowing for simultaneous execution of multiple operations, which is particularly beneficial for tasks like matrix computations or image processing.
4. **Reduced CPU Load:** By offloading intensive tasks, hardware acceleration frees up CPU resources for other critical system operations, improving overall system responsiveness.
5. **Real-time Processing Capabilities:** The speed improvements often enable real-time processing of complex data streams, crucial for applications like computer vision in robotics or high-frequency trading in finance.
6. **Scalability:** Some accelerators, particularly GPUs, offer easy scalability by allowing multiple units to work in parallel, enabling the handling of larger datasets or more complex problems.
7. **Specialization:** Accelerators can be optimized for specific types of computations, offering unparalleled efficiency for those tasks.

3.6.3 Examples

Graphics Processing Units (GPUs)

Originally designed for rendering graphics, GPUs have become crucial in various fields due to their ability to perform parallel computations. They are widely used in:

- Computer Vision: For tasks like object detection, image segmentation, and feature extraction.
- Machine Learning: Accelerating training and inference of neural networks.
- Scientific Simulations: Physics simulations, molecular dynamics, and climate modeling.

Field-Programmable Gate Arrays (FPGAs)

FPGAs offer flexibility along with hardware acceleration. They are used in:

- Signal Processing: For radar systems, software-defined radio, and audio processing.
- Financial Services: High-frequency trading algorithms.
- Data Centers: As accelerators for specific workloads like data compression or encryption.

Application-Specific Integrated Circuits (ASICs)

These are custom-designed chips for specific applications, offering the highest efficiency but at the cost of flexibility. Examples include:

- Cryptocurrency Mining: Bitcoin mining ASICs.
- Video Encoding/Decoding: Hardware video codecs in smartphones and cameras.
- Network Processors: For packet processing in high-speed network equipment.

Tensor Processing Units (TPUs)

Developed by Google, TPUs are specialized for machine learning tasks, particularly:

- Neural Network Training: Accelerating the training of large models.
- Inference: Rapid execution of trained models in production environments.

Digital Signal Processors (DSPs)

Optimized for digital signal processing tasks, DSPs are used in:

- Audio Processing: In smartphones, hearing aids, and professional audio equipment.
- Telecommunications: For signal encoding, decoding, and modulation.
- Industrial Control: In motor control systems and power electronics.

AI Accelerators

A broad category of emerging hardware designed specifically for AI workloads, including:

- Neural Processing Units (NPUs): Found in modern smartphone SoCs for on-device AI tasks.
- Intel's Nervana and Habana Labs processors: Designed for data center AI workloads.

3.6.4 Application in Robotics and Aphid Detection

In the context of robotics and specifically for applications like aphid detection, hardware acceleration plays a crucial role. For instance, a GPU or a specialized vision processing unit (VPU) can significantly accelerate image processing and neural network inference tasks. This allows for real-time analysis of high-resolution images from multiple cameras, enabling the robot to detect aphids accurately while moving through a field at practical speeds.

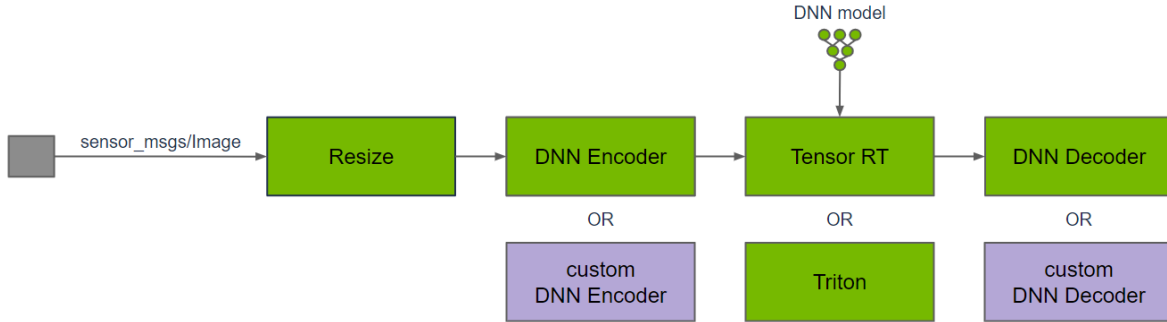


Figure 3.6: *Example of a hardware accelerated image pipeline. (Source NVIDIA Isaac ROS)*

3.7 Efficient Hardware Acceleration

Efficient hardware acceleration in robotics pipelines is crucial for maximizing performance gains and minimizing latency, particularly in real-time applications such as computer vision and sensor data processing. The efficiency of hardware acceleration is largely determined by data movement strategies between the CPU and the accelerator (typically a GPU). Inefficient hardware acceleration often involves frequent data transfers between the CPU and GPU for each operation in a processing pipeline. For instance, in a ROS graph, an inefficient approach might have one node send data from the CPU to the GPU for processing, receive the results back to the CPU, only for the next node to send this data back to the GPU for further processing. This pattern of constant back-and-forth data movement incurs significant overhead in terms of latency and bandwidth utilization, negating much of the performance benefit offered by the hardware accelerator. In contrast, efficient hardware acceleration minimizes data movement by keeping data resident on the accelerator throughout multiple processing steps. This concept of minimizing data movement aligns well with the principles of ROS 2 node composition¹⁰. Node composition in ROS 2 allows multiple nodes to be combined into a single process, enabling them to share memory and communicate more efficiently. This approach significantly reduces the overhead of inter-process communication and data serialization, leading to improved performance, especially in high-throughput scenarios¹⁰.

By composing nodes that perform related tasks or operate on the same data, developers can create more efficient pipelines that minimize unnecessary data transfers and context switches. Building upon these efficiency principles, NVIDIA’s NITROS (NVIDIA Isaac Transport for ROS) takes the concept of minimizing data movement even further, specifically in the context of GPU acceleration. NITROS implements a zero-copy architecture where data remains in GPU memory, and nodes exchange lightweight ”tickets” that reference the data location. This approach combines the benefits of ROS 2 node composition with GPU-specific optimizations, allowing for efficient hardware acceleration across multiple processing steps. The implementation of NITROS provided by NVIDIA involves several key components:

- **Shared Memory Pool:** NITROS utilizes a pre-allocated shared memory pool on the GPU, managed by a custom allocator. This pool allows for efficient memory reuse without the need for frequent allocations and deallocations.
- **Message Passing with Tickets:** Instead of passing entire data structures, NITROS nodes exchange small ”tickets” that contain metadata about the data, including its location in the shared memory pool. These tickets are typically only a few bytes in size, significantly reducing communication overhead.
- **Custom ROS 2 Middleware:** NITROS implements a custom ROS 2 middleware layer that intercepts standard ROS 2 publish and subscribe calls, translating them into NITROS operations when possible.
- **Type Adaptation:** NITROS provides type adapters that allow seamless conversion between standard ROS 2 message types and NITROS-optimized formats, enabling compatibility with existing ROS 2 code.
- **Automatic Fallback:** When NITROS-optimized communication isn’t possible (e.g., when communicating with non-NITROS nodes), the system automatically falls back to standard ROS 2 communication methods.

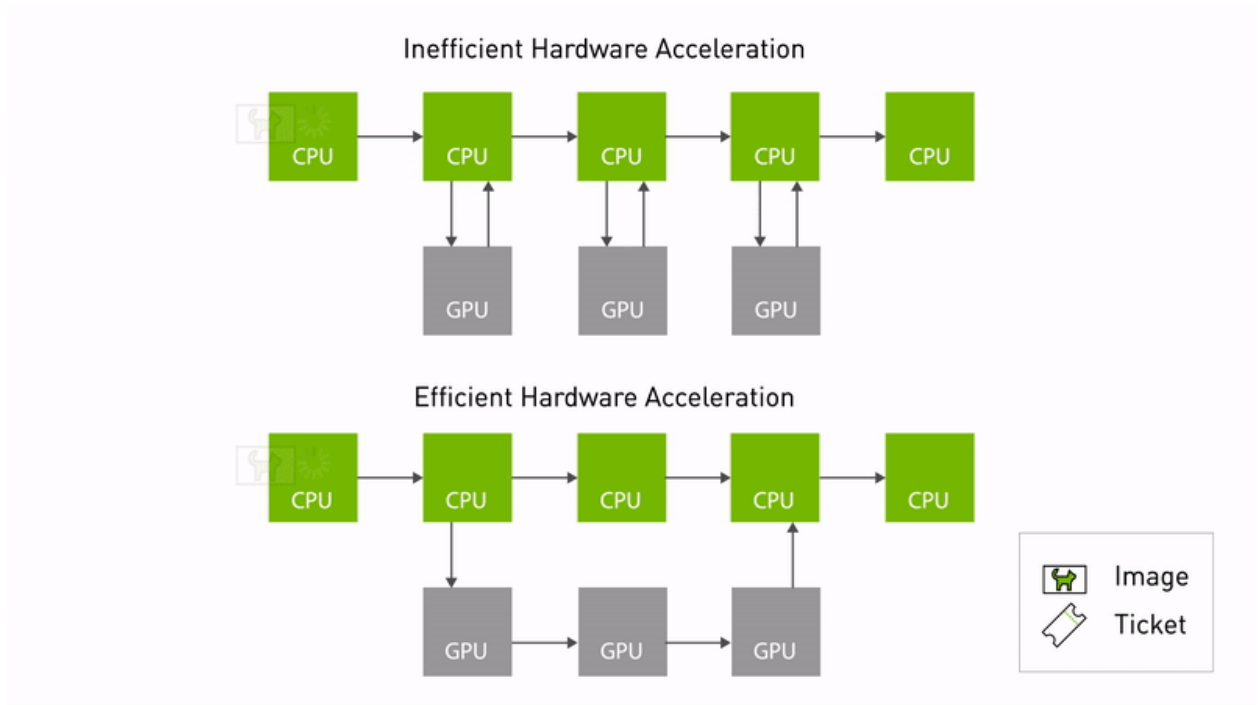


Figure 3.7: *Comparison of inefficient and efficient hardware acceleration strategies. (Source NVIDIA Isaac ROS)*

As it can be observed in 3.7 The efficient model minimizes data transfer between CPU and GPU, utilizing GPU memory for consecutive operations and reducing overall latency. The efficient model uses a ticket system for memory management, allowing multiple operations to be performed on the GPU without unnecessary data transfers. (Source: [nvidia-isaac-ros.github.io/concepts/nitros/index.html](https://github.com/nvidia-isaac-ros/nitros))

3.8 Neural Network Architecture and Deployment

In order to detect aphids in sorghum plants, a neural network architecture is required to process the images captured by the robot’s cameras. The main goals were to achieve high accuracy in detecting aphids, while also having low inference time in order to run in real-time on the robot’s hardware platform. These two goals are usually conflicting, as more complex models tend to have higher accuracy but also higher inference time. Two models

were tested for this application. The datasets were provided by Dr. Ivan Grijalva and Dr. Brian McCornack from the Kansas State University Department of Entomology.

It was also required to optimize the model to the specific hardware where it is being deployed. Inference libraries like ONNX Runtime and NVIDIA’s TensorRT play a crucial role in optimizing deep learning models for efficient deployment on various hardware platforms. TensorRT, specifically designed for NVIDIA GPUs, is an SDK that optimizes neural network models for faster and more efficient execution. It employs several techniques to enhance performance, including precision calibration (reducing from FP32 to FP16 or INT8), layer and tensor fusion, kernel auto-tuning, and dynamic tensor memory management. These optimizations target key aspects such as inference latency, throughput, memory footprint, and energy efficiency. TensorRT analyzes the computational graph of a model, fusing multiple layers into single operations where possible, selects the most efficient GPU kernels, and eliminates unnecessary operations. In the context of an aphid detection system on an agricultural robot, these optimizations enable real-time processing of high-resolution images from multiple cameras, allow for the deployment of more complex and accurate models within the computational constraints of an embedded system, and reduce power consumption, ultimately extending the operational time of battery-powered robots and increasing the area that can be covered efficiently.

To begin with, the U-Net network architecture was tested. U-Net is a popular convolutional neural network (CNN) architecture for image segmentation tasks. Its main application is for medical image segmentation, but it has been used in other applications as well.¹¹ A PyTorch implementation of U-Net was trained with an image resolution of 1024x736 and normalized according to the mean and standard deviation of the dataset as indicated by a previous study that utilized the same dataset to perform segmentic segmentation on aphids¹². The model was later exported to an ONNX file to later be deployed on the Jetson AGX Orin using the Nvidia Isaac Triton Inference Node, with ONNX runtime as its backend. Nvidia Triton is an open-source inference serving software that simplifies the deployment of AI models at scale in production environments. It is designed to help developers deploy trained AI models from any framework (such as TensorFlow, PyTorch, ONNX Runtime, or

TensorRT) on any GPU or CPU-based infrastructure (cloud, data center, or edge). It was intended to use the Nvidia Isaac Tensor-RT node directly instead of Triton, however, since the model had layers not compatible with Tensor-RT, it was necessary to use the Nvidia Isaac Triton ROS 2 node with ONNX runtime as a backend, and partial Tensor-RT optimization and Float16 quantization was applied. The results of aphid segmentation using the U-Net architecture are illustrated in Figure 3.8. This figure demonstrates the effectiveness of the U-Net model in accurately identifying and delineating aphid infestations on plant leaves.



Figure 3.8: *Aphid segmentation results using U-Net. Left: Original image. Right: Segmented output highlighting aphid infestations.*

Initial testing with U-Net Semantic Segmentation yielded promising results, achieving approximately 12 FPS on the Jetson AGX Orin embedded device. While this performance was commendable, the project requirements demanded higher frame rates to accommodate increased rover speeds and ensure comprehensive aphid detection across all captured frames. Furthermore, a shift from segmentation to object detection was considered advantageous¹³, as it would enable the estimation of 3D aphid locations from 2D images using methods to be discussed later. Although instance segmentation could have addressed this need, it was deemed unnecessarily computationally expensive for the task at hand. Consequently, the focus shifted to RT-DETR (Real-Time Detection Transformer), selected for its exceptional balance of inference speed and accuracy¹⁴. Testing revealed impressive performance, with the ResNet101 backbone achieving around 40 FPS and the ResNet50 backbone reaching approximately 60 FPS. These models were deployed using the NVIDIA Isaac TensorRT ROS 2

node with FP16 quantization, further optimizing inference speed. The substantial improvement in frame rate not only allows for higher rover speeds but also provides a comfortable margin to ensure consistent aphid detection. A more detailed analysis of these results will be presented in the subsequent results section.¹⁵

Figure 3.9 below demonstrates the RT-DETR inference output from the rover’s camera, processed in real-time on the Jetson AGX Orin, showing aphid detections on an infested sorghum plant.

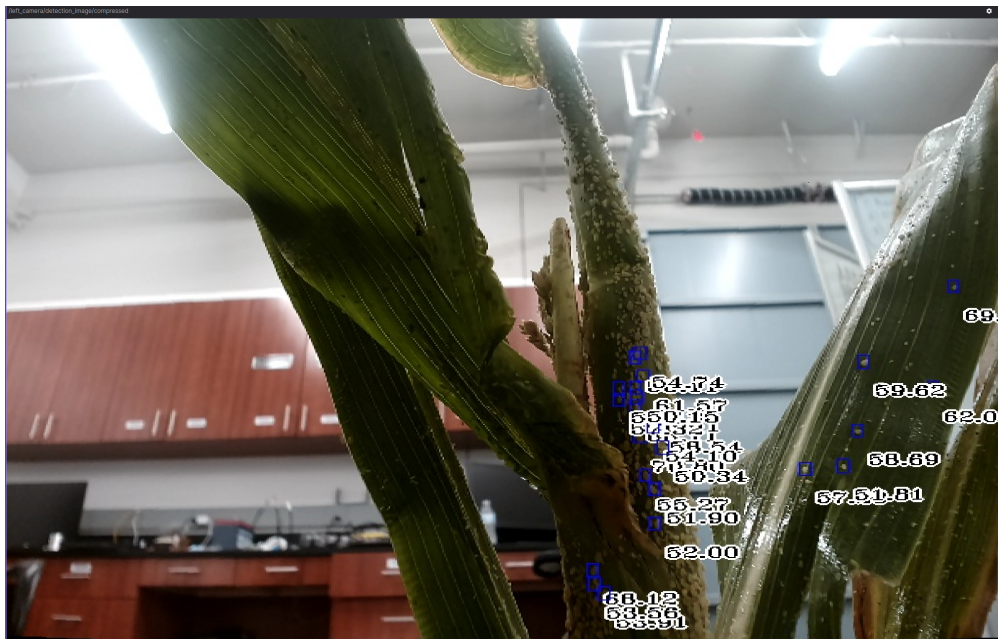


Figure 3.9: *RT-DETR inference results showing aphid detections on an infested sorghum plant, processed in real-time on the Jetson AGX Orin.*

3.9 2D Detection to 3D Coordinates Estimation

In this project, the decision to estimate depth from 2D images assuming a fixed aphid size was driven by practical considerations and project constraints. While stereo cameras and time-of-flight sensors are common choices for depth perception in robotics, they present certain limitations and complexities that were deemed unsuitable for this application. Stereo cameras have a minimum distance threshold below which depth estimation becomes unreliable.

Given the close-range nature of aphid detection on plants, this limitation could significantly impact the system’s effectiveness. Time-of-flight cameras, while capable of providing accurate depth information, introduce additional complexities to the project. Aligning the depth data from a time-of-flight camera with RGB data from a separate camera is a non-trivial task, requiring precise calibration and registration processes. This alignment procedure not only adds to the overall system complexity but also introduces potential points of failure, such as misalignment due to vibrations or environmental factors. Furthermore, specialized hardware that seamlessly integrates RGB and depth data can be prohibitively expensive. In light of these factors and the project’s time constraints, a monocular depth estimation approach was adopted. This method offers a simpler, more robust solution that aligns well with the project’s primary goal of identifying and treating aphid-infested plants. While this approach assumes a fixed aphid size, which introduces some inherent inaccuracies, it was hypothesized to provide sufficient accuracy for the task of localized spraying if the fixed aphid size given to the algorithm was in the average of a sorghum aphid. The efficacy and limitations of this method will be thoroughly examined in subsequent sections of this thesis.

The depth estimation method used in this study is based on the principle of perspective projection¹⁶, where the relationship between an object’s actual size, its size in the image, and its distance from the camera can be expressed as:

$$Z(\text{depth}) = \frac{\text{Object Size} * \text{Focal Length}}{\text{Object Pixel Size}} \quad (3.1)$$

This formula forms the basis of the monocular depth estimation approach for aphid detection used in this study.

After computing the depth of the aphid from the camera, the next step is to determine its full 3D position. This process involves several key steps:

1. **Pixel Projection:** The center of the detection bounding box is projected into 3D space using the `projectPixelTo3dRay` function from the ROS 2 `image_geometry` package. This function returns a unit vector in the camera coordinate frame, pointing in the direction of the rectified pixel (u,v) in the image plane. The resulting vector has z=1.

2. Depth Scaling: The unit vector obtained from the pixel projection is then scaled using the previously calculated depth. This scaling transforms the unit vector into a 3D point representing the aphid’s position relative to the camera.

To implement this process, a custom ROS 2 node was developed. This node performs the following operations:

1. Subscription: The node subscribes to a Detection 2D Array topic, which contains the output from the RT-DETR decoder, providing the 2D bounding boxes of detected aphids.
2. Camera Information: It obtains the camera’s intrinsic parameters from the `camera_info` message, which is crucial for accurate 3D projections.
3. Unit Vector Computation: Using the `image_geometry` ROS 2 package, the node computes the unit vector in the direction of each detected aphid.
4. Depth Calculation: The node calculates the depth of each aphid with respect to the camera using the formula described earlier, assuming a fixed aphid size.
5. 3D Point Calculation: By scaling the unit vector with the calculated depth, the node obtains the 3D point coordinates of each aphid relative to the camera.
6. Coordinate Frame Transformation: The 3D point is loaded into a pose message. The node then transforms this pose from the camera’s optical frame to the world frame, ensuring consistency with the overall robotic system’s coordinate system.
7. Pose Array Publication: For each aphid in the Detection 2D array, its pose with respect to the world frame is appended to a pose array message. This array, containing the 3D positions of all detected aphids, is then published for use by other nodes in the system.

This process effectively bridges the gap between 2D image detections and 3D world coordinates, enabling precise localization of aphids in the robot’s operational space. The use of ROS 2’s `image_geometry` package and coordinate frame transformations ensures that

the resulting 3D positions are accurate and consistent with the robot’s understanding of its environment.

This process effectively bridges the gap between 2D image detections and 3D world coordinates, enabling precise localization of aphids in the robot’s operational space (Figure 3.10). The use of ROS 2’s `image_geometry` package and coordinate frame transformations ensures that the resulting 3D positions are accurate and consistent with the robot’s understanding of its environment.

The implementation of this node demonstrates the integration of computer vision techniques with robotics principles, showcasing how 2D image processing results can be meaningfully translated into actionable 3D information for tasks such as targeted pesticide application.

While the implementation proved effective for the project’s primary goal of identifying infested plants, some discrepancies were observed in the depth estimation. When running the 2D detection to 3D point node in normal mode (without averaging, publishing all poses for each frame), the z-coordinates in the camera frame showed noticeable variations. The estimated positions of aphids appeared spread out in depth, although not significantly (Figure 3.10). It is theorized that this discrepancy is due to the natural variation in aphid sizes, with individual aphids being either smaller or larger than the fixed size parameter given to the algorithm. This size variation directly affects the precision of the depth estimation. This theory was tested and will be discussed in more detail in Sections 3.12 and 3.13.

However, these variations did not substantially impact the application’s effectiveness, as the system primarily relies on the x-coordinate for determining plant infestation and timing the sprayer activation. The y and z coordinates, while less critical for the spraying decision, still play a role in the overall localization. The y-coordinate is less crucial since the entire plant is treated if infested, regardless of the specific infested area. The z-coordinate’s primary importance lies in scaling the unit vector obtained from `projectPixelTo3dRay`. To address potential inaccuracies, the pre- and post-target spraying distances in the sprayer manager node can be adjusted to compensate for depth estimation variations.

It is worth noting that the object size parameters used for depth estimation were im-

plemented as ROS 2 parameters for easy modification. Typically, a size of 3mm was used, reflecting the average size of sorghum aphids (2-4mm), with occasional use of 2mm for comparison.

As shown in Figures 3.11, 3.12, and 3.13, the system effectively detects and localizes aphids as the plant moves across the camera's field of view, from left to right.

A comprehensive validation of this 3D localization method will be presented and discussed in Sections 3.12 and 3.13.

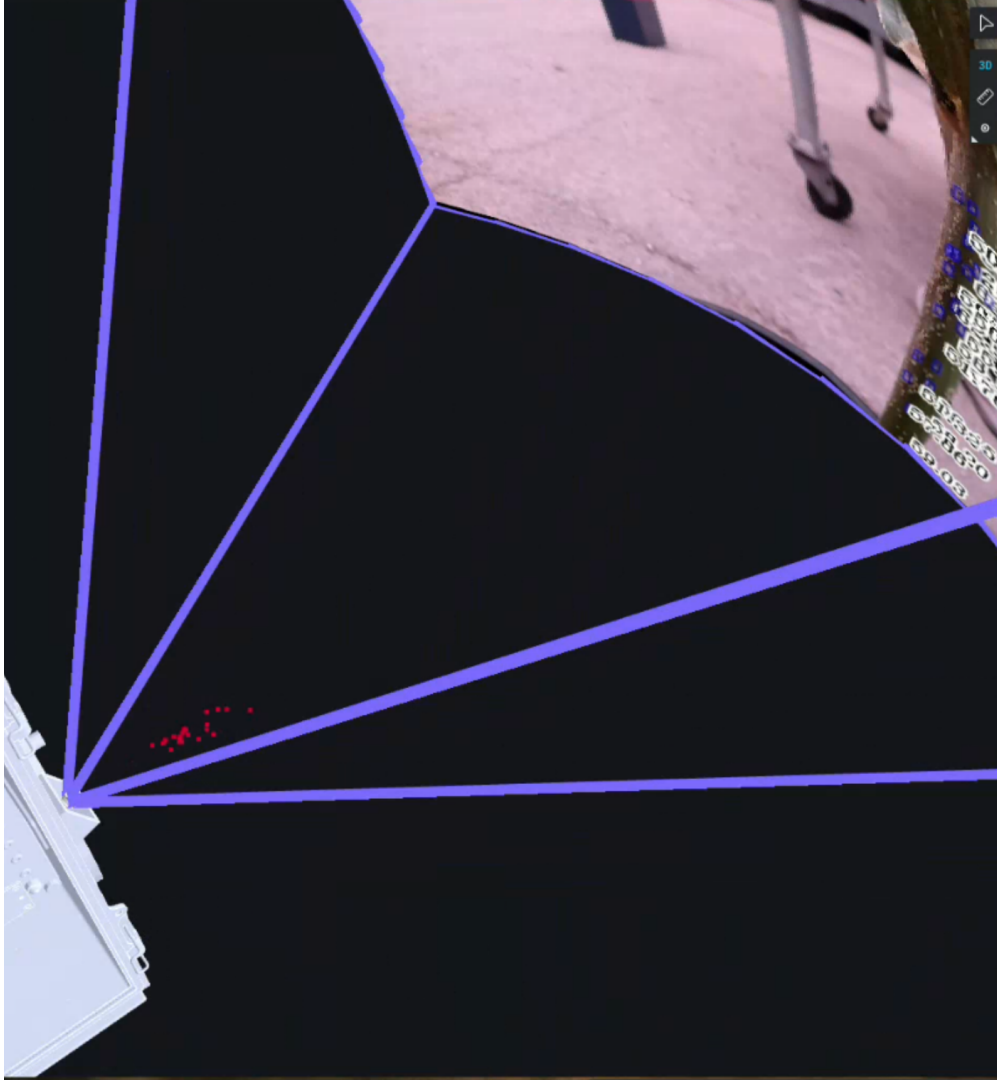


Figure 3.10: *Visualization of 3D aphid localization. Detected aphids are represented as markers in the 3D space relative to the robot and plant positions. The spread of estimated aphid positions in the z -axis (depth) is visible, illustrating the discrepancies in depth estimation.*

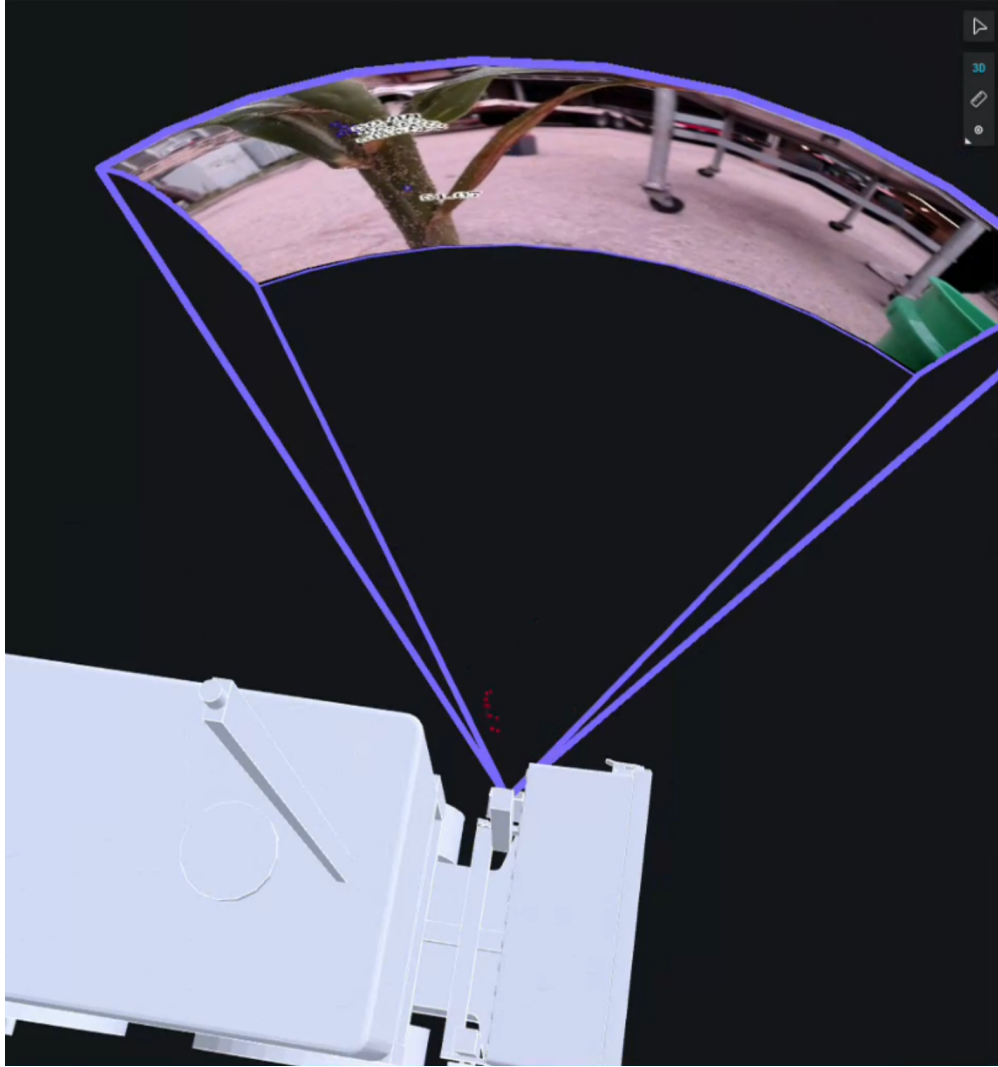


Figure 3.11: *Aphid detection and 3D localization with the plant at the left edge of the camera frame. The image demonstrates how the 2D detections correspond to 3D localizations when the plant is at the left side of the field of view.*

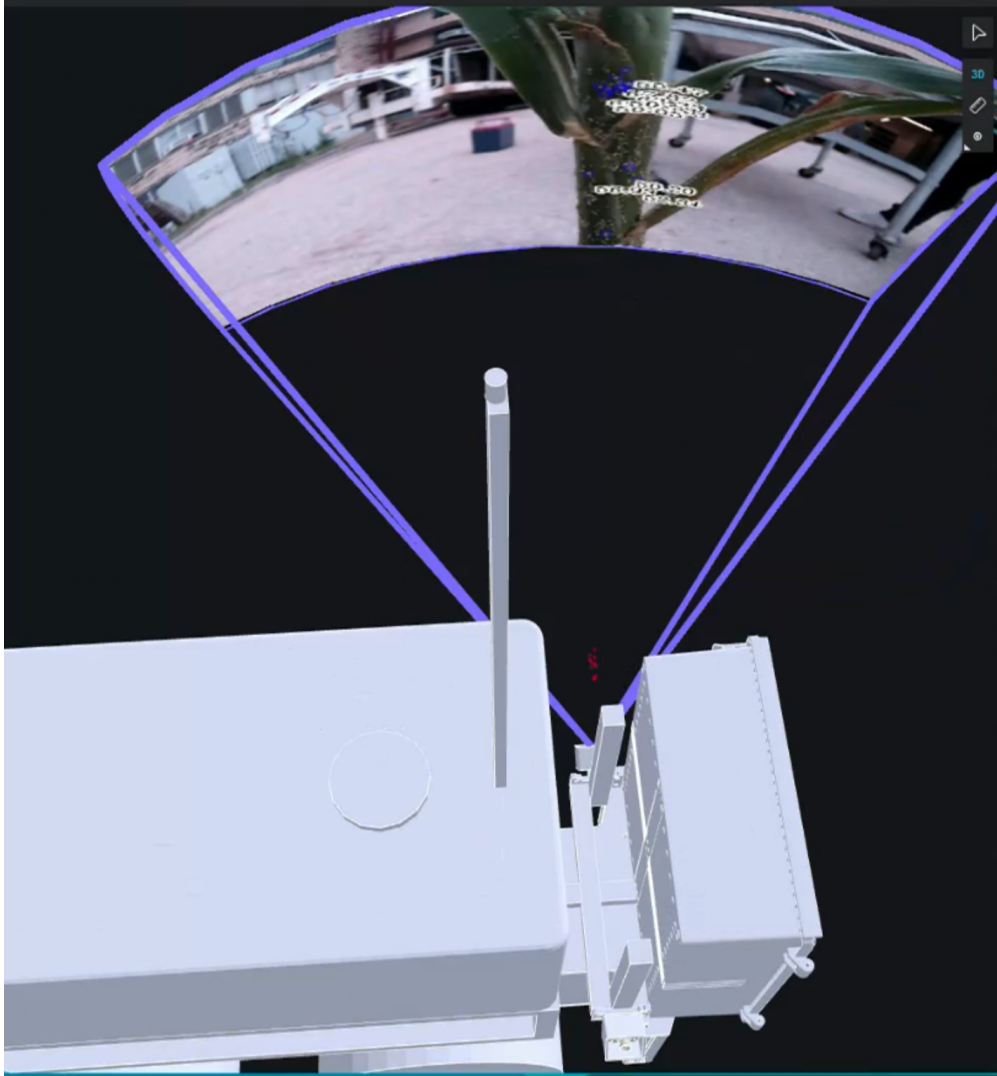


Figure 3.12: *Aphid detection and 3D localization with the plant in the center of the camera frame. This image shows the correspondence between 2D detections and 3D localizations when the plant is directly in front of the camera.*

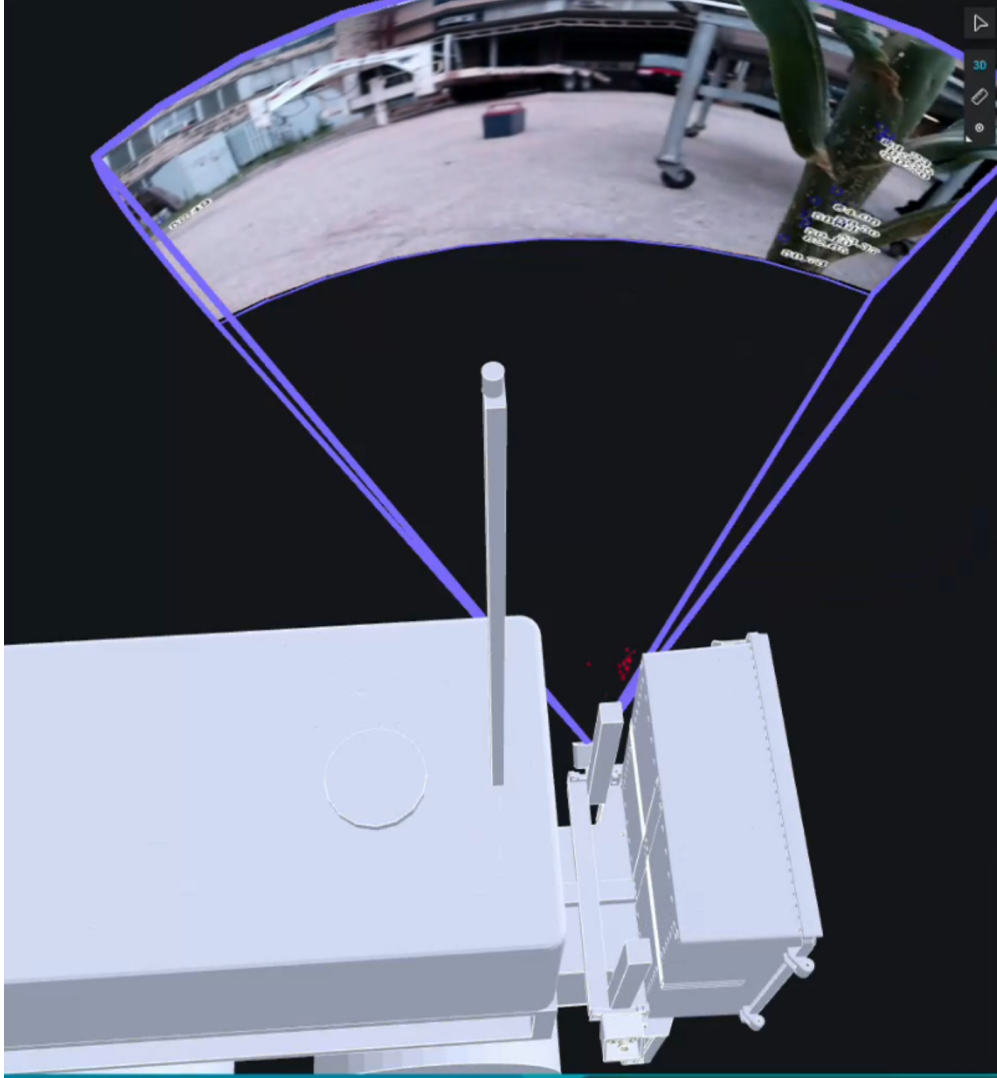


Figure 3.13: *Aphid detection and 3D localization with the plant at the right edge of the camera frame. This image illustrates how the system performs 2D detection and 3D localization as the plant moves to the right side of the field of view.*

3.10 Sprayer Manager Node

A crucial component of the aphid control system is the Sprayer Manager Node, which orchestrates the precise application of pesticides based on the detected aphid locations. This node works in conjunction with a micro-ROS node called Sprayer, running on an STM32 microcontroller, to translate the 3D localization data into targeted spraying actions.

The Sprayer Manager Node subscribes to two Pose Array topics, one for each camera (left and right), which contain the 3D positions of detected aphids in the world coordinate frame. Upon receiving a pose array message, the node populates a buffer with the individual poses, maintaining separate buffers for the left and right camera detections. This buffering mechanism allows for efficient processing of multiple aphid detections and helps manage the temporal aspects of the spraying operation.

A dedicated thread within the node continuously processes the buffered poses. It begins by examining the first pose in each buffer, transforming the aphid coordinates from the world frame to the respective sprayer frame (left or right, depending on the source camera). This transformation is critical for determining the aphid's position relative to the sprayer boom, enabling precise targeting.

The node then evaluates whether each aphid is within the predefined spraying bounds. These bounds are determined by two key parameters: the pre-target and post-target distances. The pre-target distance defines the point at which the sprayer should activate in anticipation of reaching the aphid, while the post-target distance indicates when the sprayer should deactivate after passing the aphid's position.

When an aphid is detected within the pre-target and post-target distance, the Sprayer Manager Node interacts with the micro-ROS Sprayer node. This interaction occurs through a service called `/set_nozzle_status`, which accepts a `nozzle_status` message containing `nozzle_id` and `duty_cycle` fields. For this project, a simplified approach was adopted: the entire left boom is treated as `nozzle_id 0`, and the right boom as `nozzle_id 1`. This decision was made to spray the whole plant if aphids are detected, regardless of their specific location on the plant.

The micro-ROS Sprayer node is hardcoded to control the three nozzles on each side according to the respective `nozzle_id`. Furthermore, for simplicity and to meet project deadlines, the node interprets any non-zero duty cycle as 40%, effectively creating an on/off control for each boom. When an aphid is within the target range, the Sprayer Manager activates all nozzles on the corresponding side (left or right) by setting the appropriate `nozzle_id` with a non-zero duty cycle.

If an aphid’s position exceeds the post-target distance, indicating that the rover has moved past it, the corresponding pose is removed from the front of the buffer, and the manager proceeds to process the next pose in line. Simultaneously, it sends a command to turn off the solenoid for that side if no other aphids in the buffer require spraying. Conversely, if aphids remain within the target range, their poses are retained in the buffer for reassessment in subsequent processing cycles.

This implementation strikes a balance between precision and practicality. While it doesn’t provide individual nozzle control or variable spray intensity, it ensures that entire plants with detected aphids receive treatment. The separation of left and right boom control allows for some degree of targeted application, reducing unnecessary pesticide use compared to blanket spraying approaches.

By implementing this decision-making process, the Sprayer Manager Node, in conjunction with the micro-ROS Sprayer node, ensures that pesticide application is both responsive and efficient within the constraints of the project. It minimizes wastage by spraying only when aphids are detected within the effective range of the sprayer boom. This system exemplifies the integration of robotic control, real-time decision making, and agricultural precision.

Figure 3.14 illustrates the data flow from the neural network decoder through to the Sprayer Manager and the micro-ROS Sprayer node. This diagram showcases the pipeline from initial aphid detection to the final spraying action, highlighting the key nodes and the information passed between them. The RT-DETR Decoder Node outputs detection data, which is then processed by the Detection to Pose 3D Node. This node utilizes transform (`/tf`) and static transform (`/tf_static`) information to convert 2D detections into 3D poses. The Sprayer Manager receives these 3D poses along with transform data, making decisions that ultimately result in service requests to the micro-ROS Sprayer node running on the STM32 microcontroller.

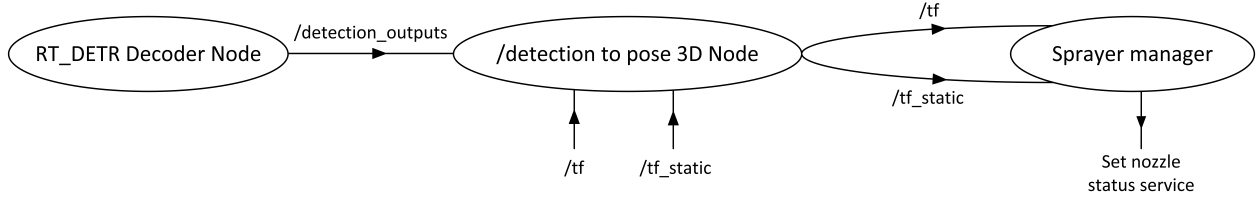


Figure 3.14: *Data flow diagram from RT-DETR Decoder to Sprayer Manager and micro-ROS Sprayer node.*

3.11 Hardware Accelerated Image Processing and Neural Network Inference pipeline

The image processing pipeline developed for this project leverages hardware acceleration to achieve real-time performance in aphid detection and localization. This section details the complete pipeline, with a particular focus on the hardware-accelerated ROS 2 nodes provided by NVIDIA’s ISAAC ROS packages (version 3.0.0). Figure 3.15 illustrates the entire pipeline flow.

3.11.1 Pipeline Overview

The pipeline begins with the camera driver node and progresses through various processing stages, ultimately culminating in the sprayer control. The key components are as follows:

1. **Camera Driver Node:** The pipeline initiates with the DepthAI ROS driver, which is the driver compatible with the OAK-FFC 4P PoE camera used in this project.
2. **Hardware Accelerated ROS 2 Nodes:** This section comprises the core of our hardware-accelerated processing, utilizing NVIDIA ISAAC ROS packages. These nodes employ the NITROS (NVIDIA ISAAC Transport for ROS) framework for efficient data transfer between GPU-accelerated nodes.

3. **Post-Processing Nodes:** Following the hardware-accelerated section, the pipeline includes custom nodes for 3D point estimation and sprayer management.
4. **Micro-ROS Integration:** The pipeline concludes with the micro-ROS agent interfacing with the Sprayer node on the STM32 microcontroller.

3.11.2 Hardware Accelerated Nodes

The NVIDIA ISAAC ROS nodes form the backbone of our hardware-accelerated processing. These nodes, leveraging NITROS transport, are as follows:

1. **Rectify Node:** This node performs image rectification, correcting for lens distortion and preparing the image for subsequent processing steps.
2. **H264 Encoder Node:** Running parallel to the main processing pipeline, this node compresses the raw image to H264 format. This compression enables efficient recording in ros2 bags for later replay or dataset creation.
3. **Resize Node:** Adjusts the image dimensions to match the input requirements of the neural network model.
4. **DNN Encoder Node:** Prepares the resized image for neural network inference by converting it to a tensor format and normalizing it based on the provided mean and standard deviation parameters.
5. **TensorRT Inference Node:** Executes the aphid detection model using NVIDIA's TensorRT for optimized inference on the GPU.
6. **RT-DETR Decoder Node:** While not hardware-accelerated itself, this node receives tensors via NITROS transport. It decodes the model output into a detection array, transferring data back to the CPU for further processing.

The NITROS transport graph encompasses nodes from the Rectify Node through to the RT-DETR Decoder Node, enabling high-speed, low-latency data transfer between these GPU-accelerated components.

3.11.3 Post-Processing and Control

Following the hardware-accelerated section, the pipeline continues with:

1. **2D to 3D Point Estimation Node:** Converts 2D detections to 3D world coordinates.
2. **Sprayer Manager Node:** Processes 3D aphid locations and makes spraying decisions.

While these nodes are composed with the previous ones, they operate outside the NI-TROS transport graph, as they do not utilize GPU or PVA (Vision Programing interface) acceleration.

3.11.4 Micro-ROS Integration

The pipeline culminates with the micro-ROS agent, which interfaces with the Sprayer node running on an STM32 microcontroller. This final stage translates high-level spraying decisions into physical actuation of the sprayer nozzles.

3.11.5 Hardware Accelerated Pipeline Overview

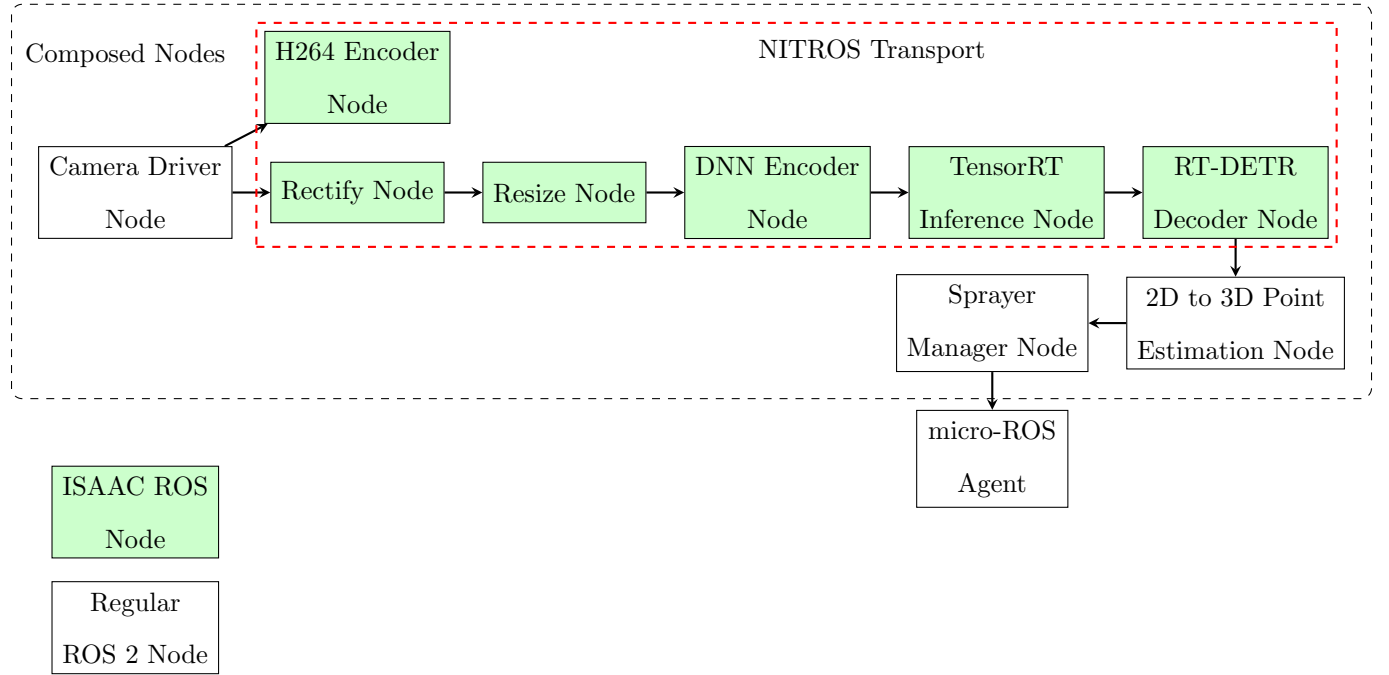


Figure 3.15: *Hardware Accelerated Image Pipeline with NVIDIA ISAAC ROS Nodes*

3.12 Aphid Depth Estimation from 2D Detection Validation Experiment Setup

To investigate the cause of depth estimation discrepancies and validate the 3D localization method, a controlled experiment was designed. The setup aimed to isolate variables affecting depth estimation accuracy, particularly focusing on the impact of aphid size and camera angle.

3.12.1 Image Preparation

Five versions of an aphid-infested plant image were prepared:

- Three images with aphids edited to a uniform size: 2mm, 3mm, and 4mm respectively

- One unedited image with naturally varying aphid sizes, representing real-world conditions

It is important to note that while aphid sizes were controlled, their orientations varied, reflecting real-world conditions (Figure 3.16).

3.12.2 Experimental Conditions

The experiment was conducted under two primary camera angle configurations:

1. Flat angle (0 degrees): The camera was positioned parallel to the image surface.
2. Upward angle (40 degrees): The camera was tilted upwards, simulating the optimal angle for detecting aphids in a 76.2cm (30 inches) width row crop.

For each configuration, measurements were taken at various distances:

- Flat angle: 7cm, 10cm, 14cm, 18cm, and 25cm
- Upward angle (40 degrees): 8cm, 10cm, 14cm, 18cm, and 25cm measured in a flat line.

When considering the camera angle to the flat surface, these distances correspond to approximately 16cm, 17cm, 20cm, 23cm, and 30cm respectively.

It is important to note that measuring distances in an angled setup presents inherent challenges. The complexity of maintaining precise measurements at a 40-degree angle may have introduced some accuracy errors. These potential discrepancies should be considered when interpreting the results.

The slight difference in the closest measurement (8cm instead of 7cm) for the angled setup was due to physical constraints when the camera was rotated. It is worth noting that the actual distances from the camera to the image surface in the angled setup were greater than the measured flat distances due to the trigonometry of the setup.

3.12.3 Data Collection

For each combination of image type, camera angle, and distance, comprehensive data was recorded to enable future recreation of the experiment when replaying the ros2 bag. However, for the direct computation of the experiment results, only the marker data containing the points array was utilized. The recorded topics encompassed a wide range of data types, including transform frames, robot description, joint states, IMU data, GPS information, camera feeds, odometry, heading, and detection outputs. This extensive data collection ensures the potential for further in-depth analysis beyond the scope of the current study, providing a rich dataset for future investigations into the system’s performance and behavior under various conditions.

3.12.4 Environmental Factors

To ensure consistent aphid detection across different distances and angles, lighting conditions were adjusted as needed. An LED light was employed in some cases to compensate for varying ambient light conditions.

3.12.5 Camera Calibration

The camera’s angle was adjusted in the URDF (Unified Robot Description Format) to accurately reflect the physical changes in the robot’s configuration, ensuring consistency between the physical setup and the robot’s internal model.

Figure 3.16 shows the experimental setup from the camera’s perspective at a flat angle, while Figure 3.17 depicts the setup with the camera at a 40-degree upward angle. Figures 3.18 and 3.19 provide external views of the experimental setup for the flat angle and 40-degree angle configurations, respectively.

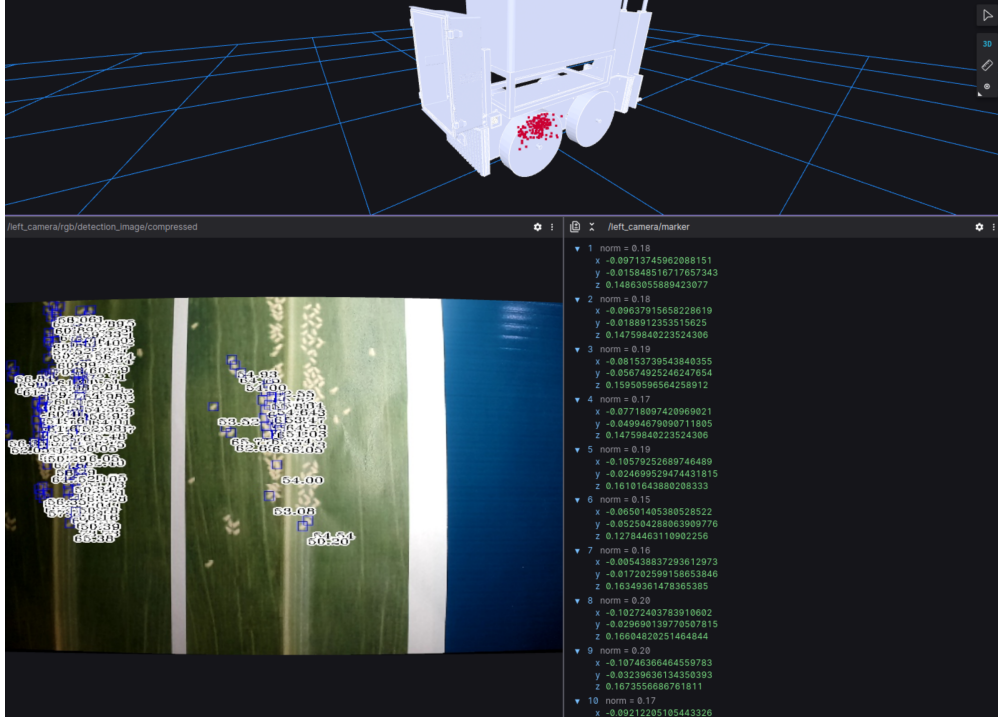


Figure 3.16: *Experimental setup viewed from the camera at a flat (0-degree) angle.*

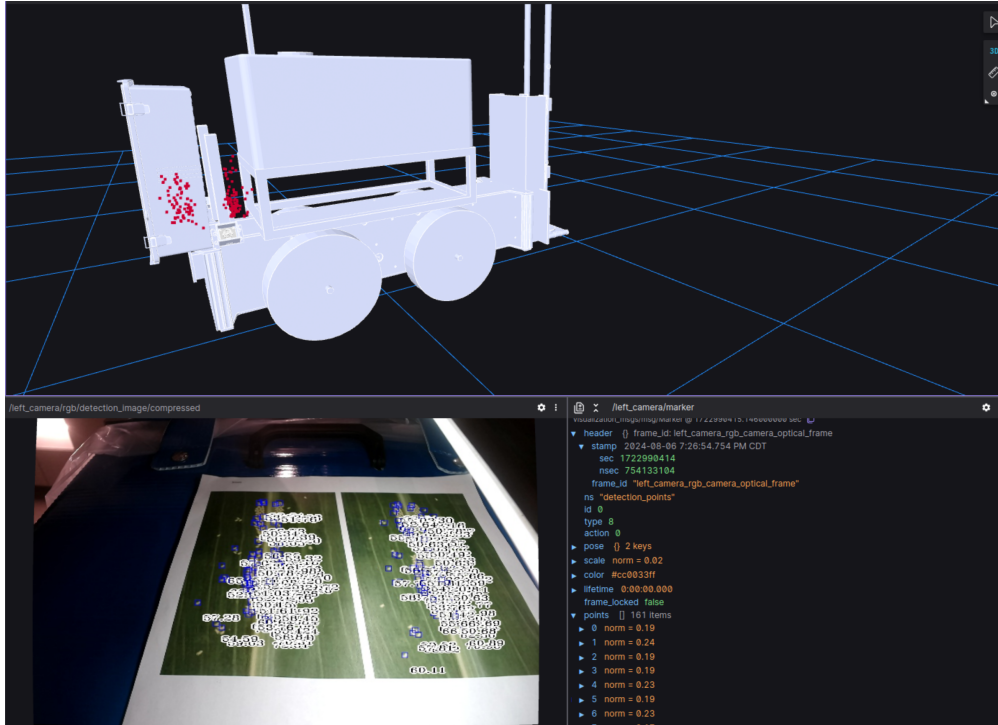


Figure 3.17: *Experimental setup viewed from the camera at a 40-degree upward angle.*

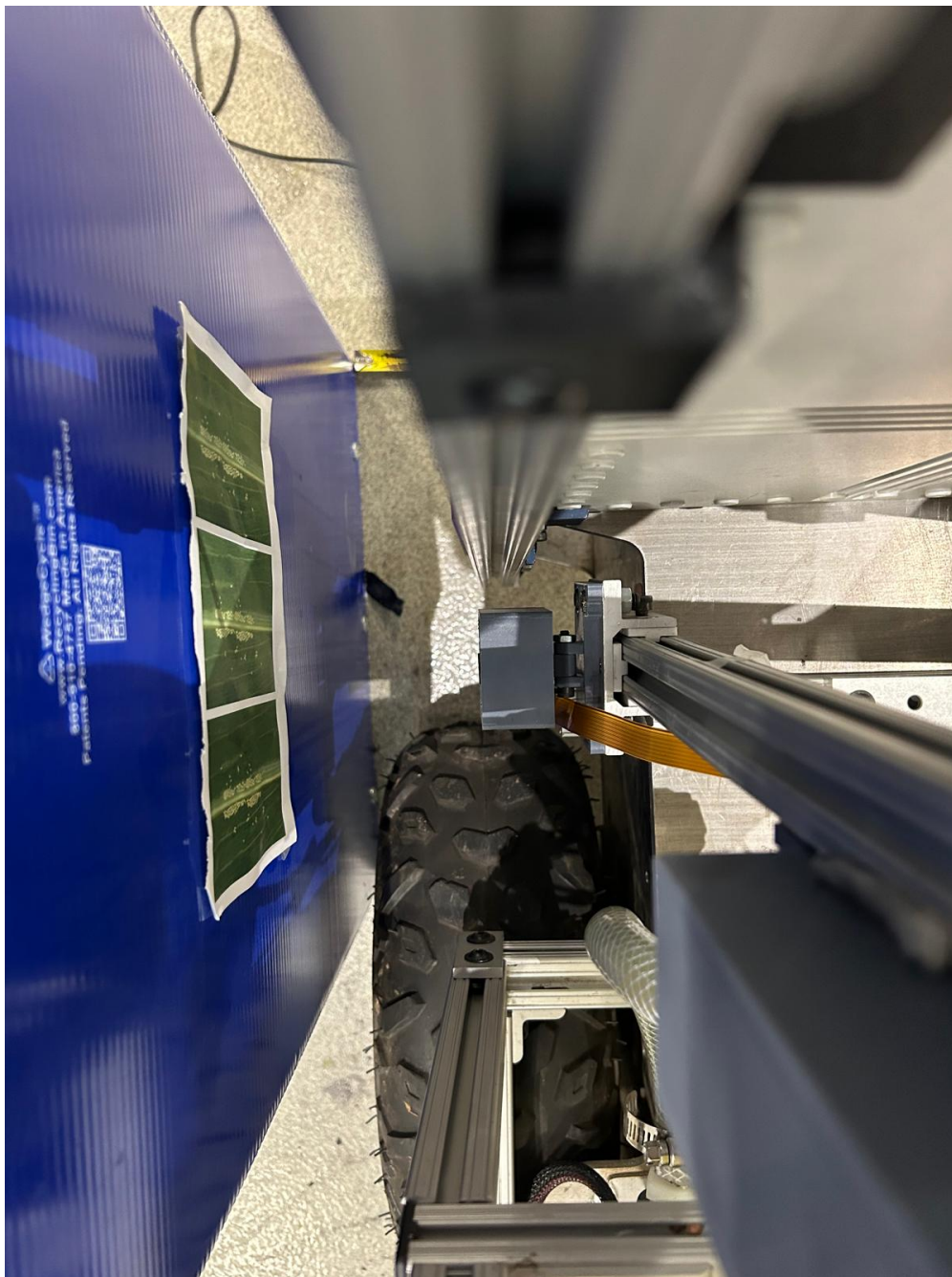


Figure 3.18: *External view of the experimental setup with the camera at a flat (0-degree) angle.*



Figure 3.19: *External view of the experimental setup with the camera at a 40-degree upward angle.*

This rigorous setup allowed for a comprehensive analysis of the depth estimation accuracy under various controlled conditions, providing insights into the system's performance and the impact of different factors on depth estimation.

The results of this experiment, including detailed analysis of depth estimation accuracy across different aphid sizes, camera angles, and distances, will be presented and discussed in Section 3.13. This analysis will provide valuable insights into the performance and limitations of the 3D localization method under various conditions.

3.13 Aphid Depth Estimation from 2D Detection Validation Results

3.13.1 Data Processing Methodology

To analyze the depth estimation accuracy across various configurations, each experimental setup was recorded in a separate ros2 bag file. A custom Python script was developed to extract and process the data from these ros2 bags systematically. The processing methodology was as follows:

1. For each ros2 bag, the 50th marker message from the `/left_camera/marker` topic was selected as a random image frame reference.
2. From this marker, the array of points with respect to the camera optical link was extracted.
3. For the z-axis values of all points in this array, the mean and standard deviation were computed.
4. The total number of points in the array was recorded.
5. All extracted and computed data were recorded in a CSV file for further analysis.
6. Additionally, the absolute error and percentage error were calculated by comparing the estimated depth (mean z-value) to the known true depth for each configuration.

This systematic approach ensured consistent data extraction and initial processing across all experimental configurations, providing a solid foundation for comparative analysis.

3.13.2 Results Presentation

The results of the experiment are presented in both tabular and graphical formats. For each camera angle (0 degrees and 40 degrees upwards), a comprehensive table is provided, showing the computed statistics and error metrics for all aphid sizes and distances. Additionally, a bar chart for each camera angle is presented to visually compare the absolute errors across all configurations.

Tabular Results

Tables [3.1](#) and [3.2](#) show the results for the flat (0 degrees) and angled (40 degrees upwards) camera configurations, respectively. In these tables, aphid size, mean, min, max, absolute error, and standard deviation are presented in millimeters (mm). Angles are in degrees.

Table 3.1: *Depth Estimation Results for Flat Camera Angle (0 degrees)*

Aphid Size	Distance	Mean Z	Std Dev	Num Points	Abs Error	Error %
2	70	87.9582	12.2353	2	17.9582	25.65457143
2	100	114.3149	11.5504	201	14.3149	14.3149
2	140	136.1757	16.6973	171	3.8243	2.731642857
2	180	152.6176	22.9915	110	27.3824	15.21244444
2	250	159.953	25.5245	54	90.047	36.0188
3	70	77.24	7.5973	19	7.24	10.34285714
3	100	130.6163	14.7939	96	30.6163	30.6163
3	140	154.0119	14.4222	256	14.0119	10.0085
3	180	169.7846	18.3939	240	10.2154	5.675222222
3	250	200.0001	20.6923	195	49.9999	19.99996
4	70	83.0241	0	1	13.0241	18.60585714
4	100	131.9552	17.1994	20	31.9552	31.9552
4	140	163.1285	18.3254	169	23.1285	16.52035714
4	180	188.6077	19.8351	178	8.6077	4.782055556
4	250	225.371	28.8916	159	24.629	9.8516
Varied	70	102.1981	22.5941	97	32.1981	45.99728571
Varied	100	139.2201	25.7721	169	39.2201	39.2201
Varied	140	154.4057	34.0089	140	14.4057	10.28978571
Varied	180	166.0737	28.827	98	13.9263	7.736833333
Varied	250	202.9141	34.348	62	47.0859	18.83436

Table 3.2: *Depth Estimation Results for Angled Camera (40 degrees upwards)*

Aphid Size	Distance	Mean Z	Std Dev	Num Points	Abs Error	Error %
2	160	118.6794	4.746	6	41.3206	25.825375
2	170	136.3949	23.7703	21	33.6051	19.76770588
2	200	177.8811	28.5345	40	22.1189	11.05945
2	230	194.9645	26.2183	34	35.0355	15.23282609
2	300	200.0587	28.5573	8	99.9413	33.31376667
3	160	212.2018	31.8476	137	52.2018	32.626125
3	170	216.6998	35.5365	185	46.6998	27.47047059
3	200	205.2463	27.9926	171	5.2463	2.62315
3	230	203.0023	23.8017	142	26.9977	11.73813043
3	300	204.5591	27.2824	109	95.4409	31.81363333
4	160	237.434	56.0746	142	77.434	48.39625
4	170	234.5368	50.2171	167	64.5368	37.96282353
4	200	214.0878	37.5422	187	14.0878	7.0439
4	230	221.3838	30.8881	167	8.6162	3.746173913
4	300	237.7629	37.3451	167	62.2371	20.7457
Varied	160	205.5972	36.7462	110	45.5972	28.49825
Varied	170	202.3291	37.6676	69	32.3291	19.01711765
Varied	200	188.1347	30.0557	50	11.8653	5.93265
Varied	230	188.1507	32.5108	60	41.8493	18.19534783
Varied	300	195.1355	32.3292	31	104.8645	34.95483333

Graphical Comparison

Figures 3.20 and 3.21 present bar charts comparing the absolute errors across all experimental configurations for the flat and angled camera setups, respectively. These visualizations allow

for quick identification of performance trends across different setups.

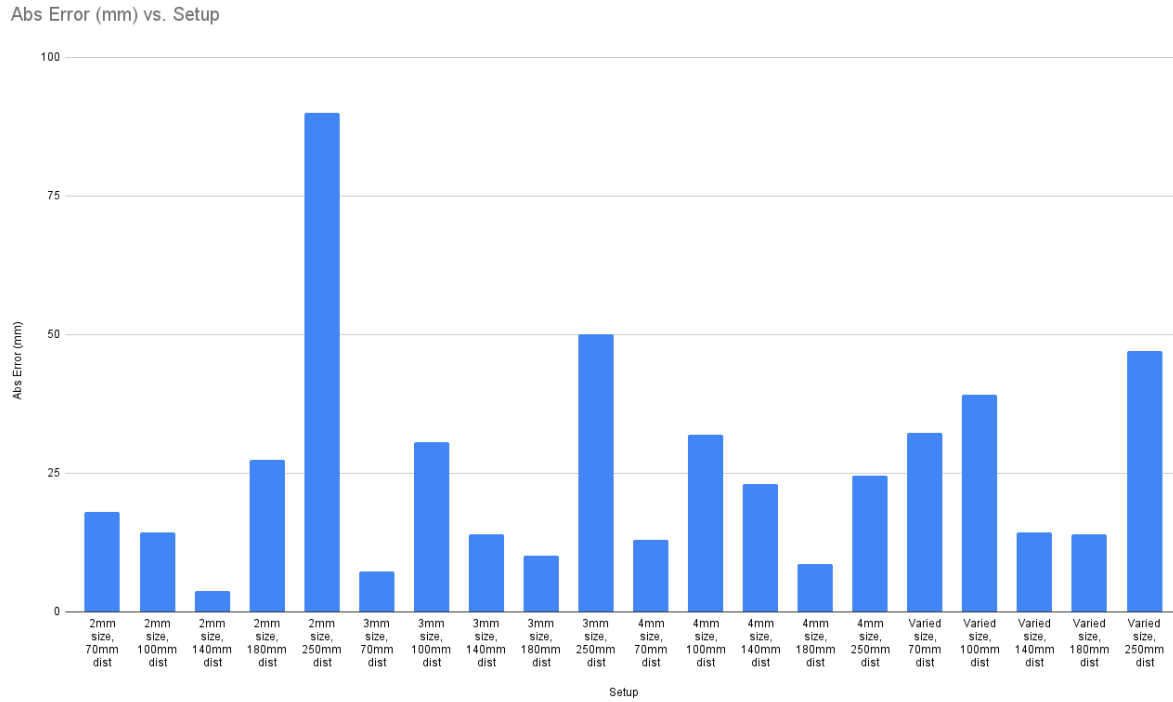


Figure 3.20: *Comparison of Absolute Errors Across All Configurations (Flat Camera Angle)*

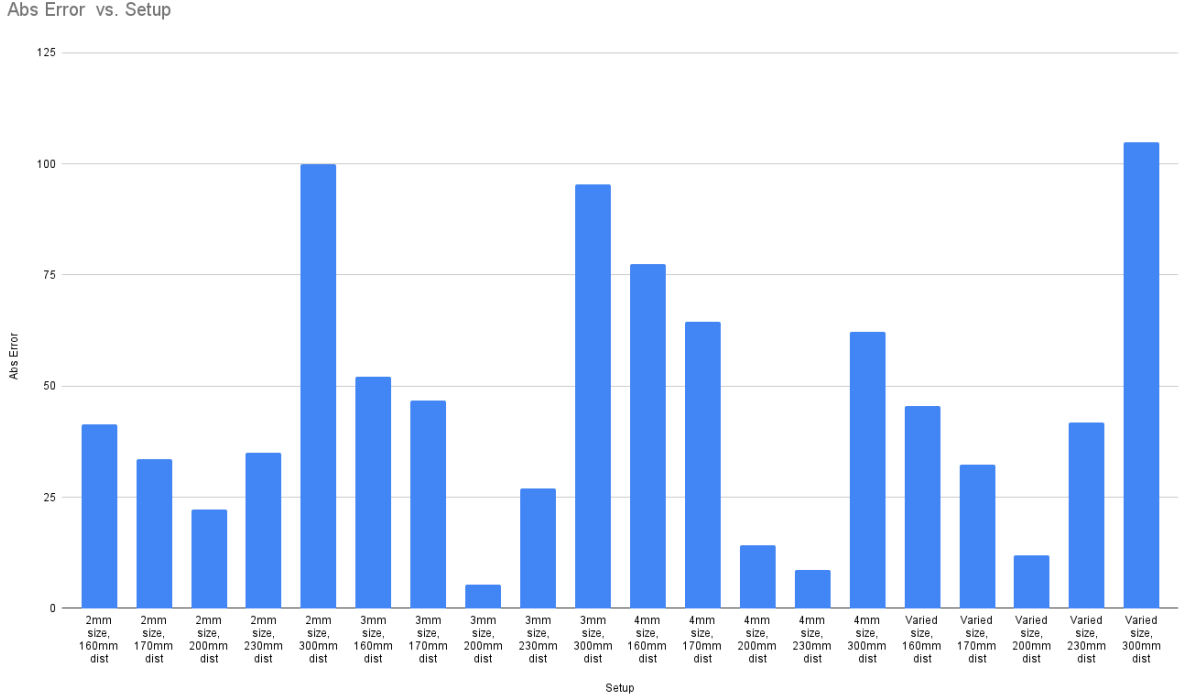


Figure 3.21: *Comparison of Absolute Errors Across All Configurations (Angled Camera)*

3.14 Results analysis

The experimental results for aphid depth estimation reveal several key insights into the performance and limitations of the implemented method across both flat angle (0 degrees) and angled (40 degrees upwards) camera setups. In the flat angle configuration, a trend of increasing error with distance was observed, particularly pronounced at 180mm and 250mm distances. However, the error was generally lower in the 70mm to 140mm range, with the minimum absolute error of 3.82mm occurring with a 2mm aphid size at 140mm distance. The maximum absolute error reached 90.047mm for 2mm aphids at 250mm, while the mean absolute error across fixed aphid sizes was 24.46mm.

The 40-degree camera angle setup presented additional challenges, partly due to the difficulty in precisely measuring the distance from the camera to the image plane. In this configuration, while greater distances generally corresponded to larger errors, some closer

distances also exhibited significant inaccuracies. Interestingly, the 200mm distance often yielded the lowest errors in the angled setup, with a minimum absolute error of 5.25mm for 3mm aphids. The maximum absolute error in this configuration reached 104.86mm for varied aphid sizes at 300mm distance (99.94mm for 2mm aphids if excluding varied sizes), with a mean absolute error of 45.7mm for fixed aphid sizes.

Notably, in both flat and 40-degree configurations, the varied aphid size datasets showed consistently high errors across most distances, although certain ranges (140mm and 180mm for flat, 200mm for angled) produced more tolerable results. This variability likely stems from the challenges in accurately estimating depths for aphids of diverse sizes within a single image. The increased errors in the angled setup can be attributed to perspective distortion, which affects the perceived size of aphids and consequently the accuracy of the depth estimation algorithm.

It is important to note that the RT-DETR model used for aphid detection did produce some false positives, albeit at a very small percentage. While computing the mean of all points in the point array of the marker helps mitigate this issue to some extent, it could have affected the accuracy readings in configurations where the number of sampled points was low. In these cases, the averaging process may not have been as effective in mitigating this noise, potentially contributing to some of the observed errors.

These experiments were conducted with images of aphids on a flat surface, which may not fully represent the complexities of real-world scenarios where aphids can adopt various poses on plant surfaces. The results highlight the method’s strengths at moderate distances and for larger aphid sizes, while also revealing limitations in accuracy for smaller aphids at greater distances, particularly in angled configurations.

These findings suggest that future improvements could focus on refining the estimation algorithm to better account for perspective distortion, developing more robust detection methods for small objects at varying distances and orientations, and implementing more sophisticated filtering techniques to handle false positives, especially in cases with low point counts.

Chapter 4

Chapter 4 - Conclusions

4.1 Summary

This research has demonstrated the potential of advanced hardware acceleration techniques, particularly the NVIDIA Jetson AGX Orin platform and NVIDIA Isaac ROS packages, in revolutionizing precision agriculture through the development of an autonomous aphid detection and targeted spraying system. The integration of these technologies has successfully addressed several key challenges in deploying real-time neural networks for agricultural applications.

The use of the Jetson AGX Orin platform, with its powerful GPU capabilities, has proven instrumental in achieving real-time processing of high-resolution images for aphid detection. The performance results are particularly noteworthy, with the RT-DETR model achieving 60 FPS for the ResNet50 backbone and 44 FPS for the ResNet101 backbone. This real-time performance is crucial for accurate pest detection in a moving agricultural rover, enabling timely and precise pesticide application.

A significant advantage of the NVIDIA Isaac ROS packages is their utilization of the Programmable Vision Accelerator (VPI) rather than relying solely on the GPU. This specialized hardware acceleration for image processing tasks further enhances the system's efficiency and performance. For instance, the Rectify Node demonstrated remarkable performance with 936

FPS and a latency of just 2.5 ms at 30 Hz for 1080p resolution (Source:https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_image_pipeline). By offloading these vision-specific tasks to the VPI, the system can maintain high-speed image processing capabilities while freeing up GPU resources for other computational tasks, such as neural network inference.

A key innovation in this project is the development of a monocular depth estimation technique for 3D aphid localization. While the precision of this method may not be suitable for applications requiring extreme accuracy, such as robotic arm manipulation, it has proven adequate for the purpose of targeted pesticide spraying. By implementing appropriate pre- and post-target spraying distances, the system can compensate for estimation errors, ensuring effective coverage of infested plants. This approach eliminates the need for more complex and expensive depth cameras, maintaining system simplicity and cost-effectiveness.

The energy efficiency of the system is also noteworthy. With the Jetson AGX Orin Fanless version operating at an average power consumption of 30.4 W at maximum power settings, the system demonstrates its suitability for field operations where power management is crucial.

The integration of these components into a ROS 2 framework has resulted in a robust and flexible system. The use of hardware-accelerated nodes for image processing, neural network inference, and data compression has enabled the rover to operate autonomously in real-time, making decisions on pesticide application without the need for constant communication with a remote server.

While the exact quantification of pesticide savings is beyond the scope of this study, the targeted spraying approach enabled by this system holds significant promise for reducing overall pesticide use compared to traditional blanket spraying methods. This not only has potential economic benefits but also aligns with sustainable agricultural practices by minimizing environmental impact.

In conclusion, this research demonstrates the feasibility and effectiveness of deploying advanced AI and robotics technologies directly in the field for precision agriculture. The combination of real-time aphid detection, 3D localization, and targeted spraying, all powered by efficient hardware acceleration – including the specialized use of the VPI for image

processing tasks – represents a significant step forward in the development of autonomous agricultural systems.

This project contributes to the ongoing transformation of the agricultural industry, showcasing how cutting-edge technologies can be harnessed to create more efficient, sustainable, and productive farming practices. As these technologies continue to evolve and mature, they promise to play an increasingly vital role in addressing global challenges in food production and environmental stewardship.

4.2 Future Work

Building upon the achievements of this research, several avenues for future work present themselves to further enhance the capabilities and efficiency of the autonomous aphid detection and targeted spraying system. A critical next step is the integration of the right side camera, which was originally planned for this study but postponed due to project complexity and deadline constraints. This integration will involve ensuring that the Jetson AGX Orin can effectively process both cameras in parallel, potentially exploring the use of a single DNN model with a batch size of 2, rather than running two separate model instances. Additionally, optimizing the system to run the navigation camera pipeline concurrently with the left and right aphid detection cameras will be crucial for maintaining operational efficiency. Further efforts will focus on refining the depth estimation accuracy, which could significantly improve the precision of pesticide application. Expanding the range of detectable pests beyond aphids would increase the system’s versatility and applicability across various crop types and agricultural scenarios. Conducting comprehensive field trials will be essential to quantify the system’s impact on pesticide use and crop health, providing valuable data on its effectiveness in real-world conditions. Leveraging hardware acceleration for camera color space conversion, possibly utilizing the Programmable Vision Accelerator (VPI) on the Jetson platform, could yield significant performance gains in image processing. A comparative study between the current 3D aphid location estimation approach and methods using depth cameras (stereo or Time-of-Flight) would provide valuable insights into the trade-offs between simplicity,

cost-effectiveness, and accuracy. This study would consider factors such as depth estimation accuracy across various conditions, computational requirements, power consumption, cost-effectiveness, and robustness in agricultural environments. By pursuing these advancements, the system could become even more effective in supporting precision agriculture practices, ultimately contributing to more sustainable and efficient crop management strategies while reducing overall pesticide use.

Bibliography

- [1] Redmond Ramin Shamshiri, Cornelia Weltzien, Ibrahim A. Hameed, Ian J. Yule, Tony E. Grift, Siva K. Balasundram, Lenka Pitonakova, Desa Ahmad, and Girish Chowdhary. Research and development in agricultural robotics: A perspective of digital farming. *International Journal of Agricultural and Biological Engineering*, 11(4):1–14, 2018. ISSN 1934-6344. doi: 10.25165/ijabe.v11i4.4278. Publisher Copyright: © 2018, Chinese Society of Agricultural Engineering. All rights reserved.
- [2] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022. doi: 10.1126/scirobotics.abm6074. URL <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>.
- [3] Yan Li, Chunlei Xia, and Jangmyung Lee. Vision-based pest detection and automatic spray of greenhouse plant. pages 920 – 925, 08 2009. doi: 10.1109/ISIE.2009.5218251.
- [4] Roberto Oberti, Massimo Marchi, Paolo Tirelli, Aldo Calcante, Marcello Iriti, and A Nicola Borghese. Selective spraying of grapevines for disease control using a modular agricultural robot. *Biosystems Engineering*, 146:203–215, 2016.
- [5] Alvaro Fuentes, Sook Yoon, Sang Cheol Kim, and Dong Sun Park. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. In *Sensors*, volume 17, page 2022. MDPI, 2017.
- [6] Fazal E. Nasir, Muhammad Tufail, Muhammad Haris, Jamshed Iqbal, Said Khan, and Muhammad Tahir Khan. Precision agricultural robotic sprayer with real-time tobacco recognition and spraying system based on deep learning. *PLOS ONE*, 18(3):1–22, 03

2023. doi: 10.1371/journal.pone.0283801. URL <https://doi.org/10.1371/journal.pone.0283801>.
- [7] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *Proceedings of the 13th ACM SIGBED International Conference on Embedded Software (EMSOFT)*, pages 1–10, 2016.
 - [8] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.
 - [9] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22:1330 – 1334, 12 2000. doi: 10.1109/34.888718.
 - [10] Steven Macenski, Alberto Soragna, Michael Carroll, and Zhenpeng Ge. Impact of ros 2 node composition in robotic systems. *IEEE Robotics and Autonomous Letters (RA-L)*, 2023. doi: 10.48550/arXiv.2305.09933. URL <https://arxiv.org/abs/2305.09933>.
 - [11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL <https://arxiv.org/abs/1505.04597>.
 - [12] Raiyan Rahman, Christopher Indris, Tianxiao Zhang, Kaidong Li, Brian McCornack, Daniel Flippo, Ajay Sharda, and Guanghui Wang. On the real-time semantic segmentation of aphid clusters in the wild. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 6299–6306, 2023. doi: 10.1109/CVPRW59228.2023.00670.
 - [13] Tianxiao Zhang, Kaidong Li, Xiangyu Chen, Cuncong Zhong, Bo Luo, Ivan Grijalva, Brian McCornack, Daniel Flippo, Ajay Sharda, and Guanghui Wang. Aphid cluster recognition and detection in the wild using deep learning models, 2023. URL <https://arxiv.org/abs/2308.05881>.

- [14] Raiyan Rahman, Christopher Indris, Goetz Bramesfeld, Tianxiao Zhang, Kaidong Li, Xiangyu Chen, Ivan Grijalva, Brian McCornack, Daniel Flippo, Ajay Sharda, and Guanghui Wang. A new dataset and comparative study for aphid cluster detection and segmentation in sorghum fields, 2024. URL <https://arxiv.org/abs/2405.04305>.
- [15] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detsr beat yolos on real-time object detection, 2024. URL <https://arxiv.org/abs/2304.08069>.
- [16] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.