THE UNIVERSITY OF SYDNEY

The University of Sydney

School of Computer Science

COMP3888 Computer Science Project

Final Group Report

Group 4 - Monday 3pm

Author:

Jackson Bamber	510425814
Chengye Wan	530070973
Wentao Gao	470375578
Daniel Mooney	500481433
Kivaan Mudaly-Naidoo	510462048
Xavier Singarayar	510517359
Andy Xie	530322469

COMP3888_M15_04

P13 - Developing a solution to measure the height of cattle using image analysis or LiDAR

February 4, 2026

Chapter 1

Introduction

1.1 Background

Measuring cattle is an important practice for estimating physical traits and production potential. These measurements help determine whether animals are ready for beef markets and can also highlight potential health issues. By comparing size, weight, and age, farmers can assess whether cattle are growing as expected or if there may be underlying problems.

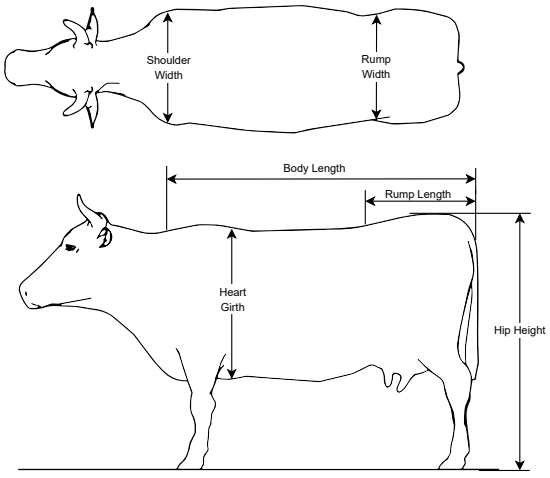


Figure 1.1: Common body measurements taken on cattle, including hip height, rump width, and shoulder width.

Among these traits, hip height is the most widely used because it provides a reliable estimate of frame score, a measure of skeletal size [7]. Other dimensions, such as rump width and shoulder width, give additional insight into muscle development and fat cover, both of which are valued differently across meat markets [6].

Traditionally, these measurements are taken manually with a measuring stick after the

animal has been confined in a cattle race. This process is shown in Figure 1.2. While effective, it is slow, labour-intensive, and can be stressful for both cattle and handlers, particularly in large herds.

(a) Cattle passing through a race.

(b) Manual measurement of shoulder height using a stick.

Figure 1.2: Manual measurement process: animals are restrained in a race (left) before height and other traits are recorded (right).

Automated measurement offers a promising alternative. Similar vision-based systems are already widely used in industrial manufacturing, where cameras and sensors perform quality assurance tasks such as checking for missing components and defects on printed circuit boards (PCBs). An example of such a system is shown in Figure 1.3. Applying the same principles to agriculture, cattle can be scanned as they move naturally through the race, eliminating the need for manual restraint. This approach accelerates data collection, reduces human error, and is less invasive, improving both accuracy and animal welfare.

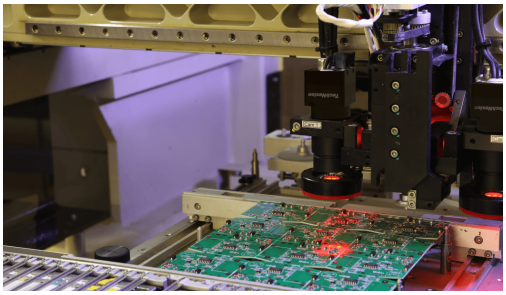


Figure 1.3: Automated inspection of a PCB using a camera.

1.2 Project Vision

The vision of this project is to replace manual measurement of cow features with a hardware system that is capable of capturing and analysing 3D image data to determine these features automatically. The system will allow for greater ease, recording and data analysis of livestock features for farmers and reduce the need for invasive practices.

1.3 Project Aim and Objectives

The aim of this project is to develop a hardware-software system which can extract the hip height of a single cow/bull passing through a confined area. A stream of hip height data (and potentially other measurements) will be available for a human observer to assess in real-time, with each cattle being uniquely identified.

- SEN: Capture scene information of a cow moving through a cow shoot
- MES: Calculate a hip height estimation of cow from scene
- DAT: Provide feedback of hip height to an isolated user in realtime
- OVR: The system should be robust

The following Deliverables must be met to reach the Objectives of the project:

Ref.	**Short Term Deliverables**	**Date**
SEN.1.0	Provide dense scene information of a cow moving through a cattle chute with a resolution of 1.5 cm at 1.5 metre distance.	08 Sept 2025
SEN.1.1	Provide dense scene information at at least 5 frames per second.	01 Nov 2025
MES.1.0	Segment a single cow from an image with an intersection over union score greater than 0.9.	08 Sept 2025
MES.2.0	Measure the hip height of a cow with an accuracy of 3 cm compared to the manual method.	01 Nov 2025
DAT.1.0	Transfer each cow measurement data point to an external user interface at a rate of two data points per second.	01 Nov 2025
DAT.2.0	External user interface should be accessed through network based communication	01 Nov 2025
OVR.1.0	The hardware should withstand Dense Dust Environments and withstand outdoor temperatures up to 40 degrees celsius	01 Nov 2025

Long Term Expectations

- Eventual deployment in commercial farms
- Reduced labour requirements for cattle management
- High quality market classification data and analysis
- Expansion to features beyond hip height and for all livestock
- Improved animal health and welfare

1.4 Stakeholders

The following table lists the groups and/or organisations that have interest, can influence or affect the project. The list is sorted in terms of priority, and include both the internal and external stakeholders of the project.

Stakeholder	**Priority Level**	**Desciption**
Luciano Gonzalez	Highest	Luciano Gonzalez is the primary project stakeholder. He is the client, and an expert in the field that can contribute his knowledge to help guide the project
School of Life and Environmental Sciences (SOLES)	High	SOLES is the school that Luciano works for. They contribute by providing funds to the project.
COMP3888_M15_04	High	Responsible for delivering on all project milestones and managing communication between the clients and themselves
Our Project Team	High	Our team will all have large influence on the project and the direction it takes. As the project is very open-ended, it has a large scope for what the final product may look like, dependent on our team's decisions. The team are also impacted by the project's outcomes and overall performance in an academic sense but also through the impacts it will have on our workload and wellbeing.
School of Computer Science	Medium	The school is where all group members come from. They can help provide guidance on the project, and give advice on client interactions/expectations.
Cattle Farmers	Medium	The team has no direct communication with cattle farmers; however, they must be kept satisfied, as the device will be deployed on farms, and used by farmers.
Farm Workers	Low	Farm workers are low on the list, as they most likely won't be using the device; however, the device needs to be designed so that it won't negatively impact on their work

1.4.1 Group Interactions

The group has agreed upon a clear plan on how they will collaborate and interact with one another as well as clients. The first method of communication amongst the team is through face to face meetings. Through the 3-5pm Monday tutorial each week this type of communication will be guaranteed and will allow for the group to speak flexibly and more

openly in comparison to online communication. It allows for the team to have full group discussions but also split up into teams to discuss and work together on separate tasks. Further it allows for higher degrees of collaboration as people can physical look at, use or test what others have (which may not be able to be shared otherwise). Any additional need for face-to-face meetings can be facilitated using mainly the university campus as a destination. The team also have agreed to meet every Wednesday 2-4pm online through slack's video call service. This meeting mostly serves to inform the entire group what each individual is working on and looking to complete in future weeks. This meeting is more structured than the Monday tutorial sessions but offer a way to keep team members accountable and ensure everyone is up to date and aware of how the project is shaping. The final method of communication is through online messaging on Slack. This final method is highly flexible, where no scheduling is required and is perfect for when people need quick responses and small issues addressed. It also serves as a place to share resources and keep records of conversations so that everyone can remember what has been said.

Communication with our client is to be done primarily through email. This is the desire of our client who has significant responsibilities outside of this project. The emails with our client is completed by our customer liaison, Jackson Bamber. Meetings are only to be had on a need-to-have basis where issues that can't be communicated efficiently through email can be completed. Through these discussions all team members are able to speak openly with the client however our customer liaison will ensure to be the primary communicator to allow for consistency.

1.5 Resources and Risks, Mitigation Strategy

Challenge/Resource Limit	**Risk Level**	**Mitigation**
RGB Data and Depth Data Being Integrated: The RGB data and depth data online had misalignments with depth and RGB making it unusable for us. This however created issues with the success of the project requiring both.	Medium	The team mitigated against this by using a sensor which retrieves RGB and Depth data with known intrinsic and extrinsic parameters. These was used to align the data. The only issue we ended up running into tied with the next point, was that it took a while to collect
Inability to Collect Data at Farm: Due to the clients conditions we were unable to visit the farm ourselves to collect data. This posed a huge risk where we had limited direction in the type of data we can collect. Further the amount of data we could retrieve was limited due to time constraints of our client.	High	To mitigate this, we delivered very clear instructions to our client of what type of data we want and how to precisely set it up. We also created a set up that would then run autonomously not requiring anyone to man it. Overall we ran into still some issues with this challenge due to the hard drive corrupting and us not being there to identify and fix the issue. This caused over a week delay in collecting data and also made us not able to do testing on hardwrae for this week. When data did come in though it was of high quality due to these steps.

Hardware Lead Time: Due to delays with shipping, hardware such as the Raspberry Pi5, AI Hat, NYX660 Camera, and other hardware requirements took weeks to arrive (much longer than shipping estimates). This prevents production of the full system from being developed.	High	To mitigate this risk, we focused on getting our programs fully ready to be integrated with the hardware as soon as it came. We also utilised any data we had available to test with - whilst not perfect it did provide a good stepping stone before the hardware allowed for us to retrieve our own data.
Stereo Camera Calibration: The calibration was very difficult to determine well and also isn't always stable under new conditions and over time.	Medium	The team moved from stereo to time of flight cameras which are far more robust and reliable and have extremely strong precision and resistance to motion blur.
Cow Movement: The cows moving was shown to create huge issues in the data found online, colours skew and spread and sometimes background blends in with the cow.	High	The hardware team chose a very robust time of flight camera as described above which reduces this effect. Further the cow detection team produced an algorithm to detect the blurriness of images and only use non blurry ones - turned out that all images with the ToF camera were strong.
External Commitments and Workload: The project we have been given was a very ambitious project as described by our client and our time and effort is a limited resource.	Medium	The team ensured work was shared and that collaboration was high to reduce risk of burnout. The team also kept moral and support high which allowed for greater satisfaction and sustainability. Lastly organisation and routine was important - the group managed to incrementally progress the project each week, not leaving it all to the end.
Lighting Issues With Sensor: Changing environment lighting was a potential risk with both outdoor and indoor.	Low	The hardware team again successful chose a device that managed to deal with this extraordinarily well, however the only weakness here was if there was a surface of water - seen in the final testing which made depth unable to be detected on this surface.
Lack of CPU Processing Power: When dealing with point clouds, matrix multiplications, and large amount of data, this can cause the CPU on the RPi5 to throttle which results in delayed data, or system crashing	Low	An AI Accelerator hat was sourced for the RPi5 which made it far stronger at running these with a high FPS. There was an issue with the Yolo model being too large and therefore it was offloaded to the computer and only our own segmentation model was kept.

Chapter 2

System Specifications and Design

2.1 Specifications and Requirements

2.1.1 Functional Requirements

- Functional Requirement 1: The system must determine whether a cow is present within the field of view of the hardware.
- Functional Requirement 2: The system must determine where the hip height should be measured along the cows back.
- Functional Requirement 3: The system must determine the hip height of a cow relative to the race floor.
- Functional Requirement 4: The system must provide feedback of the estimated hip-height to the user in real time.
- Functional Requirement 5: The system must have a way to compare the hip heights of different cows.
- Functional Requirement 6: The system must be able to work without any requirement for an operator to be present at the hardware.
- Functional Requirement 7: Hip Height must be obtained without any hindrance or restraint to the cows.

2.1.2 Non-Functional Requirements

- Non-Functional Requirement 1: The system must collect depth and rgb data with definition of 480x480px or more.
- Non-Functional Requirement 2: The system must generate estimates of hip-height within 3cm of the accepted hand measurements.

- Non-Functional Requirement 3: The system data processing pipeline must run at greater than 2 frames per second to allow for realtime feedback.
- Non-Functional Requirement 4: The system shall operate in dust, sun with heats up to 40 degrees and withstand vibrations and accidental collisions of force up to 500N.
- Non-Functional Requirement 5: The system can be accessed via an interactive front-end using wifi connection.
- Non-Functional Requirement 6: The system can show a tabular view of the different cows and their hip heights and be downloadable in csv.
- Non-Functional Requirement 7: The system should meet the Australian Welfare Standards and Guidelines for Cattle and the Prevention of Cruelty to Animals Act 1979, not injuring and minimise stress and pain where avoidable to ensure the product doesn't cause legal and social issues.

2.1.3 Implemented User Stories

ID	User Story	Req Num	Testing
US01	As a cattle livestock producer, I want to know the hip height of a cow to assess the maturity and suitability of that cow for farming.	FR3	6
US02	As a cattle livestock producer, I want to see the hip heights of cows remotely, in order to be able to monitor cows without being physically present.	NFR5	5
US03	As a cattle livestock producer, I want the height to be comparable between cattle, measured relatively to the floor and to the same spot.	FR2, FR3	4
US04	As a farm owner, I want the system to work without need for manual intervention or measurement, so labour and costs are reduced.	FR1, FR6, NFR4	1
US05	As a cattle assessor, I want the hip height to be detected in real time so my workflow isn't slowed down and I can make immediate decisions with each cow.	FR4, NFR3	5
US06	As a cattle assessor, I want the hip height measurement to be accurate so that it properly represents the cows maturity and suitability for farming.	NFR1, NFR2	2

ID	User Story	Req Num	Testing
US07	As an animal welfare officer, I want the measurements to be made in a way that is non-invasive and doesn't require restraining or cause distress to the cows upkeeping welfare guidelines.	FR7, NFR7	2
US08	As a cattle livestock producer, I want the data to be able to be downloadable so that I can easily maintain records of collected measurements.	NFR6	5
US09	As an agricultural researcher, I want to be able to compare the data and extract trends and analytical results using cow hip heights.	FR5	5
US10	As a developer, I want the system to log any bugs and incorrect data so that these can be fixed allowing for the system to work smoothly for the user.	NFR2	6

Table 2.1: User Stories for the Hip-Height Measurement System

2.1.4 Link between User Stories and Implementation

User Story 1

US01 is the primary desire of the project to produce a hip height of a cow and what our client wants. Through it FR3 is achieved where the hip height is provided of a cow relative to the ground. This user story therefore requires the full system's end to end result. As this is the main objective of the project, it's implementation explanation is saved for the rest of the report. In short, a time of flight camera retrieves RGB and depth Image data, a model determines where the cow is in the image and where the hip point is in the image. Then using the depth image, the height of this location can be retrieved and sent to the dashboard system which provides feedback to the user.

User Story 2

US02 dictates how the need for the system to be able to provide the collected information over a network so that it can be retrieved whilst not being present at the farm thus providing actuation to NFR4. For the user to not be present, the system was implemented to transfer the collected information over wifi to a Socket.IO backed UI.

User Story 3

US03 is important for ensuring the hip height is valid and comparable between cattle where the hip location is correctly determined (FR2) and the height is determined of this relative to the ground (FR3). The following were implemented utilising a groin detection deep learning

model which identifies where the hip height should be taken. The distance from this point to the floor is then determined using a RANSAC determination of where the ground plane is and then using 3D camera and depth mapping to obtain the 3D point cloud allowing the height to be retrieved.

User Story 4

US04 provides greater insight into the desire for the system to run autonomously. It therefore provides achievement to FR1 where the cow must be detected by the system on its own, success is tied with the system being robust to the outdoor elements and not fail (NFR4). The robustness to the outdoors of the system was implemented through designing a housing for the system which cooled, protected and organised all the components. The system also was designed to detect if there is a cow present and perform calculations automatically when this is determined.

User Story 5

US05 demonstrates the need for the system to provide realtime results similar to old manual practices, thus achieving FR4 and linked with NFR3. The implementation of this was done through utilising lightweight data processing with our small neural networks and efficient hardware which were both small and mobile but had a targeted AI accelerator and high performance.

User Story 6

US06 builds on US01 but provides a requirement for high accuracy. This is directly correlated to performance based non-functional requirements - NFR1 and NFR2. The standard for accuracy was delivered therefore by having a time of flight camera with large depth density readings and HD imaging. Further internal models which were developed were only accepted with high enough performance to ensure accurate results.

User Story 7

US07 promotes the wellbeing of the livestock and the requirement for the system to be non-invasive (FR7) as well as ensuring the product complies with laws and regulations (NFR7). The hardware implementation used non contact sensors and doesn't emit any hazardous byproducts. The system was also designed iteratively to be small and as hidden as possible.

User Story 8

US08 dictates that the system should allow for downloadable data of cows with their hip heights which was done with a csv file (NFR6). The implementation of downloadable datasets was completed where csv files could be generated using the UI interface which took the stored information and outputted to the user.

User Story 9

US09 necessitates the need for the UI to have a proper comparable tabular display (FR5) which allows for insights by users. This was implemented using tabular data representation automatically shown and also downloadable in the UI.

User Story 10

US11 requires for the system to be able to provide debugging controls to ensure the output is accurate for any users (NFR2). The system was implemented with a debug log which is easily reachable by anyone with computer system knowledge which allows for the intrinsics of the system to be reconciled.

Testing of each user story can be found in the Quality of Work Section

2.1.5 Non-Completed User Stories

- As a cattle livestock producer, I want to know automatically which cattle corresponds to the hip height received, so I can make cow specific decisions using this.
- As a user, I want the hip height to have a corresponding confidence score so that I know if it isn't reliable.
- As a cattle livestock producer, I want to know about other features such as fat deposition and rump width so that the system replaces all my measurement requirements.
- As a non-cattle livestock producer, I would like this technology to work for other animals so that I may assess the attributes of these other animals.

For the first non complete story, farms already scan cows as they move through a race and given the order of the cows are the same these are already able to be matched. We weren't able to get access to this technology so we ultimately decided it wasn't needed.

For the second item, whilst we experimented with techniques to evaluate how good a score would be, we were never given highly accurate ground truth data from our client which would allow us to determine the efficacy of such mechanisms.

The other above non completed features were set out as low priority at the start of the project on a nice to have basis. This is because needs provided by our client was very much on getting the hip height of cow. These other features fall primarily in the future work of such a project like this.

2.1.6 Reflections

Our Client was overall very satisfied with most of the user story and requirements we were set to to cover. The only additional aspects he desired was for the downloadable CSV records and for the system to operate up to 40 degrees celsius. Therefore both these were added

to our specifications. **Self-Reflecting** the group decided to not overload ourselves aiming for the extension user stories but rather define clear user stories so that the work that is completed are of high quality.

2.2 System Overview

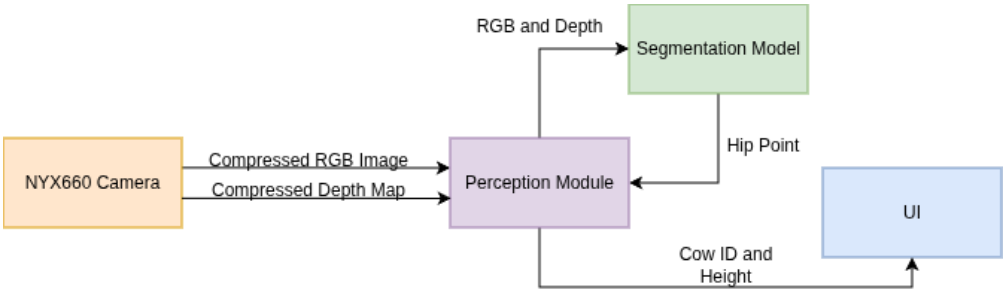


Figure 2.1: System Block Diagram of Project

This system can be broken down to four main modules of workflow. The NYX660 camera, which publishes RGB-D (red, green, blue and depth) information. This data is sent to the Perception Module, which aligns the depth and RGB image and sends it to the Segmentation Model. This model returns the hip point on the depth image, which is converted to the hip height, and sent to the User Interface by the Perception Module. From here, the dashboard display module uses a website, which will display the cow being measured, and the measurement data.

2.3 Hardware - Enclosure & Physical Setup

The hardware enclosure was designed to provide a robust, weather-resitant, and thermally stable housing for the prototype's sensing and computing components. It supports the Perception/Hardware module within the described system architecture, ensuring reliable operation in field environments such as cattle chutes. The enclosure design focuses on:

- Protecting sensitive components from dust, heat and vibration.
- Maintaining a fixed and stable camera pose for accurate height estimation.
- Enabling single-source power input to simplify field setup.
- Allowing serviceability without compromising seal integrity.

Six iterative versions were developed in SolidWorks with successive improvements guided by testing feedback, and functional requirements are discussed by the team.

(a) Isometric View

(b) Top View

Figure 2.2: Client Prototype

2.3.1 Final Design Overview

The final enclosure measures $235 \times 190 \times 86mm$ with $5mm$ **ABS** wall thickness, printed using FDM additive manufacturing with **PolyLite ABS filament** [8]. The structure include a slide-on lid secured with a single screw, allowing tool-light maintenance while maintaining environmental sealing. The internal layout as seen from a top view is described below:

- The **NYX660 ToF Camera** [10] is mounted upright on the top-right side, facing outward through a sealed aperture.
- The **Raspberry Pi 5** [9] sits on the top-left of the enclosure, mounted via hex stand-offs.
- A **Core Electronics 25W DC-DC Buck Converter** [4] is located centrally between the camera and Pi.
- The lower portion of the enclosure provides dedicated space for the coiled **M12 X-Coded Ethernet Cable** connecting the NYX660 to the Pi. The cable is zip-tied to prevent vibration-induced movement.

Ventilation meshes are positioned on the top and bottom walls (top view), with external dust filters mounted over them [1]. This arrangement provides a direct airflow path and minimises dust ingress. The filters are attached using brackets, enabling straightforward replacement. An external fan provides positive airflow through the mesh vents, supported by passive convection. The Pi and buck converter are coated with thermal paste to improve heat dissipation.

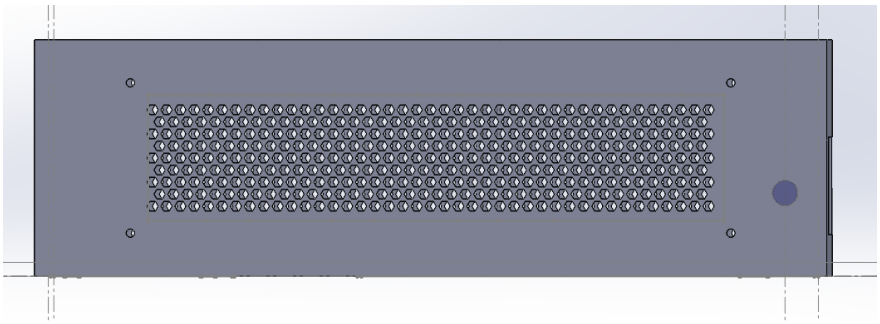


Figure 2.3: Enclosure Version 4 - Front View

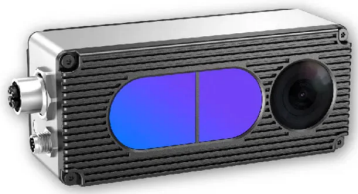
Power is supplied via a single external input, directly power the ToF, and Pi through the buck converter. A panel-mount power connector is placed opposite the camera's power port to maximise sealability and reduce strain. The design uses a male-to-female pigtail fastened internally to maintain a tight seal.

The slide-on lid allows the enclosure to be opened without tools and provides direct access to the Raspberry Pi, converter, and cabling. All electrical components are fully contained inside the housing, with only the camera face exposed externally for sensing. This design minimises electrical hazards and facilitates quick field maintenance.

2.3.2 Components and Functions

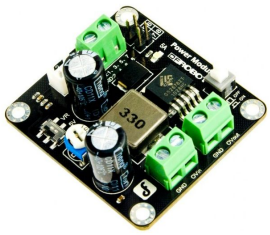
- **NYX660 Time-of-Flight Camera**: serves as the primary depth sensor, producing high-resolution RGB-distance data suitable for cattle height estimation [10]. Its rigid upright mounting ensures repeatable calibration, and the camera is connected to the Pi through a rugged M12 X-coded Ethernet interface for reliable data transmission. **Raspberry Pi 5** [**9**]: provides local computation and data management, running image capture, preprocessing, and communication tasks. It is mounted on standoffs for improved airflow, with heat transferred through thermal paste to the baseplate and circulated by the internal fan.

- **DC-DC Buck Converter (25 W)**: the Core Electronics 25 W converter [4] steps down the single external 12 V input to 5 V for the Raspberry Pi and camera. Its central placement minimises wire lengths and ensures equal current distribution.



(a) NYX660 ToF Camera

(b) Raspberry Pi 5



(c) DC-DC Buck Converter

Figure 2.4: Hardware Components

- **Dust-Filtered Ventilation System**: mesh ventilation panels on the top and bottom faces of the enclosure provide cross-flow cooling, filtered by external dust filters. The design balances airflow efficiency and dust ingress protection.

Figure 2.5: $225 \times 61\,mm$ Dust Filter

- **Cooling Fan**: An external fan maintains positive internal pressure, ensuring that air passes through the dust filters and not unsealed gaps. It reduces heat buildup buildup during continuous operation in warm environments.
- **Mechanical Structure**: The ABS enclosure provides mechanical protection, rigidity, and shock resistance. Its $5\,mm$ walls and reinforced corners resist vibration and handling stresses while maintaining manageable print times and mass.
- **Power and I/O Connectors**: a panel-mount power connector simplifies field setup and enhances enclosure sealing. Internally, power is routed cleanly via the buck converter, and all I/O connections remain internal for protection.

2.3.3 Iterative Design History

1. **Version 1: Initial Prototype**
   The first enclosure established the base geometry for the system, with external dimensions of **197 × 113 × 63 mm** and a substantial **10 mm wall thickness**. It featured 4 mm edge fillets for structural smoothness and M6 screw holes (6 mm diameter, 12 mm depth) for basic fastening. This version primarily validated internal space requirements for the NYX660 [10] and Raspberry Pi 5 [9], confirming that the core components could be enclosed securely.

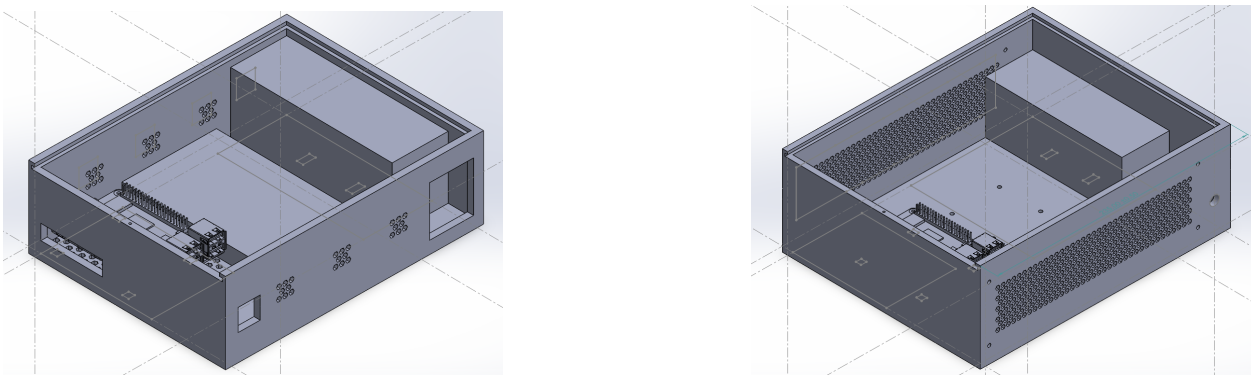
2. **Version 2: Port and Ventilation Prototype**
   The second design refined the proportions to **195 × 145 × 60 mm** and reduced the wall thickness to **5 mm**, improving print efficiency and internal volume. Edge fillets and screw holes were removed to simplify early iterations. Several port openings were added for the camera face, camera's power, Ethernet cables, Pi Ethernet port, and Pi USB-C and Micro HDMI ports. These modifications allowed direct access for initial integration and testing of all electrical components.

3. **Version 3: Slide-Lid and Ventilated Enclosure (Client Prototype)**
   This version introduced significant functional enhancements, expanding the dimensions to **205 × 160 × 65 mm** with a consistent 5 mm wall thickness. The design included a **slide-on lid** which could be fastened down. Ventilation was improved with multiple mesh openings along the base and walls. This version was used as the **client prototype** for data collection, successfully tested in limited field conditions.

4. **Version 4: Mounting and Filter Integration**
   Version 4 introduced a substantial redesign to integrate mounting points and accommodate internal components more securely. The enclosure increased to **285 × 173 × 85 mm**, maintaining a 5 mm wall thickness. Dedicated mounting holes were added for the NYX660 camera, the buck converter, and the Raspberry Pi. The base ventilation meshes were removed, replaced with a single large rectangular side mesh designed to align with an external dust filter [1]. Corresponding mounting holes were added for the dust filter. Additionally, the previous rectangular Ethernet/power opening was replaced with a **9 mm circular panel-mount hole** for an M8 male–female connector, improving sealability. This version established the structural basis for the final prototype, supporting internal mounting, dust filtration, and single-power input integration.

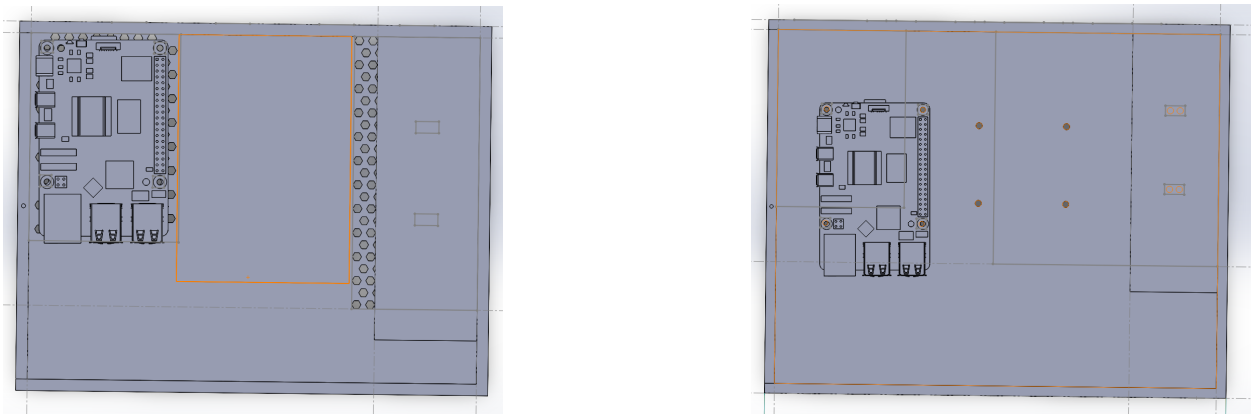
5. **Version 5: Volume Optimisation**
   The fifth iteration refined the external dimensions to **255 × 190 × 85 mm**, achieving a more compact form while preserving adequate space for cable management and airflow. This downsizing reduced print material and time, improving portability and manufacturability without compromising internal clearances.

6. **Version 6: Final Enclosure**
   The final design, used for deployment and documentation, measured **235 × 190 × 85 mm**. It incorporated several key layout refinements: repositioned Raspberry Pi screw holes to better accommodate the thick Ethernet cable, elimination of external port openings for the Pi to enhance sealability, and repositioned buck converter mounting to optimise space and airflow. Most significantly, the reduction in length was applied as the base plate of the 3D printer utilised is $256 \times 256\,mm$, and the previous length did not facilitate a viable print. These adjustments improved both usability and environmental protection, yielding a compact, well-integrated enclosure ready for field use.



(a) v3 (Client/Data Collection Prototype) (b) v6 (Final Prototype)

Figure 2.6: Isometric View Comparison from v3–v6



(a) v3 (Client/Data Collection Prototype (b) v6 (Final Prototype)

Figure 2.7: Top View Comparison from v3–v6

2.3.4 Power, Thermal, and Airflow Performance

The enclosure operates on a single external 12 V supply, with total system draw estimated below 25 W. The buck converter maintains stable 5 V output, and the simplified power path reduces external connectors and potential ingress points. Preliminary tests using the client prototype (v3) demonstrated satisfactory thermal performance (albeit with limited exposure); internal temperatures remained within acceptable limits even under continuous use, though detailed thermal logging for the final version was not completed due to time constraints. The final design is expected to perform better with its filtered airflow system and simplified power routing.

2.3.5 Interfaces with the System - Mechanical Integration

Mounting solutions were explored for field deployment, including:

1. A dual-tripod mount with a stabilised cross-bar cradle.

2. A lightweight aluminium truss clamp for adjustable elevation and orientation.

Neither could be realised within project timeframes; as such, during testing, the unit was positioned on a table overlooking the cattle chute for optimal camera perspective.

Figure 2.8: View from Enclosure in Field Setup

2.3.6 Validation and Field Observations

Although the final prototype was not field-tested due to time limitations, the v3 prototype (which shared most structural features) was deployed by the client and demonstrated resilience to dust, temperature, and vibration during limited data-collection sessions. No major mechanical or electrical failures were observed. The improved ventilation and single-supply power of the final iteration are expected to further ehance durability and ease of deployment.

2.3.7 Integration with Software and System Frameworks

The hardware design directly supports the perception stack deployed on ROS 2 Humble, running detection and measurement nodes on the Raspberry Pi 5. Data transfer and visualisation use the Flask-SocketIO framework consistent with the system's architecture defined in previous stages.

2.4 Perception

The Perception module can be split into 3 sub-modules; data handler, ground plane calibrator and height calculation. The flow of this data can be seen in figure 2.9.

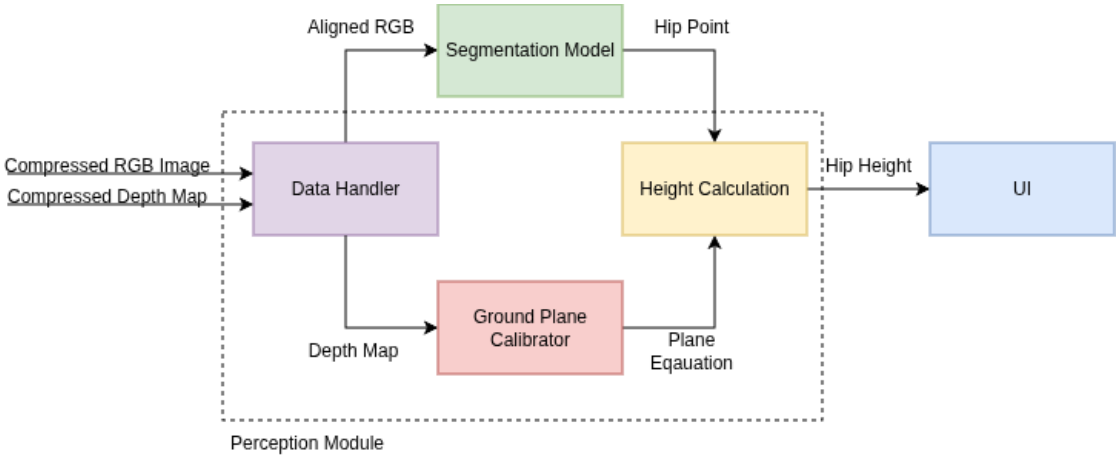


Figure 2.9: Perception Pipeline

- **Data Handler:** Takes in the compressed RGB and Depth data, decompresses it and aligns the RGB image to the depth map
- **Ground Plane Calibrator:** Takes the depth map, and finds the ground plane
- **Height Calculation:** Uses the determine hip point and the ground plane to determine the height of the cow

2.4.1 Data Handler

First, the data handler takes in the compressed depth map and RGB image data, then decompresses it. The RGB image is then aligned with the depth map, currently the RGB has a resolution of 1600 x 1200 pixels, whereas the depth map has a resolution of 640 x 480 pixels. Since the Time of Flight (ToF) sensor and camera are also apart from each other, it means to two sensors see different things. This can be clearly seen in figure 2.10, where the cable covers the two sensors, but it's position is different for both.

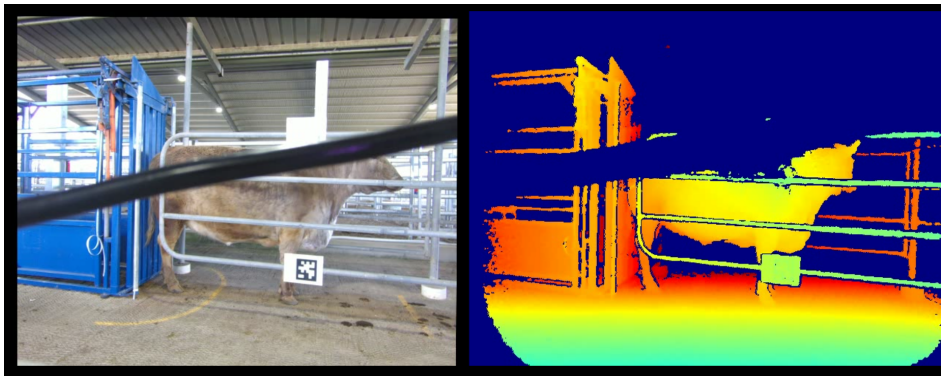


Figure 2.10: RGB Image (left) and Depth Map (Right) showing with different views

To overcome this, the camera's intrinsic and extrinsic parameters are used to re-project the RGB image onto the depth map. The intrinsic parameters can be represented as a matrix

K, where K_{RGB} and K_{depth} are the intrinsic parameters for the RGB and Depth sensors, respectively:

$$K_{RGB} = \begin{bmatrix} 1108.9476 & 0 & 795.0526 \\ 0 & 1122.2745 & 574.2967 \\ 0 & 0 & 1 \end{bmatrix}, \quad K_{depth} = \begin{bmatrix} 414.0901 & 0 & 339.3002 \\ 0 & 447.4697 & 239.3013 \\ 0 & 0 & 1 \end{bmatrix}$$

Similarly, the extrinsic parameters can be put into the form of a matrix T:

$$T_{RGB} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_{depth} = \begin{bmatrix} 0.9999931 & -0.0021066 & 0.0028361 & 0.0305948 \\ 0.0021066 & 0.9999978 & -0.0011662 & 0.0001089 \\ -0.0028361 & 0.0011662 & 0.9999967 & -0.0022923 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The relative transform from depth to RGB can be determined:

$$T_{depth \to RGB} = T_{RGB}^{-1} T_{depth}$$

For each pixel u, v in the depth map, project it into 3D space:

$$x = (u - c_{x_{depth}}) \frac{z}{f_{x_{depth}}}, y = (v - c_{y_{depth}}) \frac{z}{f_{y_{depth}}}$$

Thus, for each pixel, we have a point in 3D space:

$$p_{depth} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

These point can that be transformed into 3D points from the RGB frame:

$$P_{RGB} = R_{depth \to RGB} P_{depth} + t_{depth \to RGB}$$

Using the Camera intrinsics we can than re-project this back into the image frame:

$$\begin{bmatrix} u_{RGB} \\ v_{RGB} \end{bmatrix} = \begin{bmatrix} f_{x_{RGB}} \frac{x_{RGB}}{z} + c_{x_{RGB}} \\ f_{y_{RGB}} \frac{y_{RGB}}{z} + c_{y_{RGB}} \end{bmatrix}$$

Using u_{RGB}, v_{RGB}, this gives a mapping between depth and RGB. The result of this can be seen in figure 2.11, which shows a perfect alignment between the depth map and aligned image.

Figure 2.11: The depth map overlayed with the aligned RGB image

The issue with this process, is that all the matrix multiplications requires a lot of computing time and power, so a method to counter-act this is to pre-compute the depth map. This assumes that the depth map is a flat wall at 1m from the ToF sensor. This saves time on calculating the mapping each time, but it comes with it's drawback. In 2.12, it can clearly be seen that the RGB image is slightly shifted to the left of the depth map.

Figure 2.12: Pre-Computed mapping in two different environments

This can be counter-acted by shifting the RGB image to the right, as seen in figure 2.13. While it's not as accurate as live computing, it saves processing time, which is limited on the RPi5. This module is configurable to work in either mode in case it's running on a better processor.

Figure 2.13: Pre-Computed mapping shifted to the right in two different environments

2.4.2 Ground Plane Calibrator

Depending on where the base link is set for ToF sensors, simply acquiring a point in the point cloud and determining it's distance from the origin will rarely return the height of the cow. This would also introduce requirements for the setup, such as placing the system down at a certain height and angle at all times, going against the minimal manual interventions user story. So, in order to accurately determine height, the ground plane needs to be determined.

The most common, and widespread method for determining the ground plane, is through RANdom SAmple Consensus, otherwise known as RANSAC. This is an algorithm that fits models, such as planes, by taking a small subset of data, estimating the model fit, and determining how many points fit within this plane. This makes it great at detecting ground planes, as they are quite often large, flat areas within the point cloud [3].

For use in this project, this algorithm starts by projecting the depth map into 3D space to generate the point cloud, then selects 3 random points from the cloud. The cross product on these three points are used to determine the normal vector:

$$\hat{n} = (p_2 - p_1) \times (p_3 - p_1)$$

If the three points are collinear, the cross product is zero, so these points would be ignored. The plane is normalised, and the offset d is calculated:

$$d = -\hat{n}\dot{p}_1$$

The distance between all points in the cloud, and the found ground plane are calculated using:

$$d_i = |ax_i + by_i + cz_i + d|$$

The number of inliers (points that are close enough to the plane) are recorded. This continues on for a pre-defined number of iterations, and the plane with the largest amount of inliers is kept. Figure 2.14 shows an example of a found ground plane within the point cloud

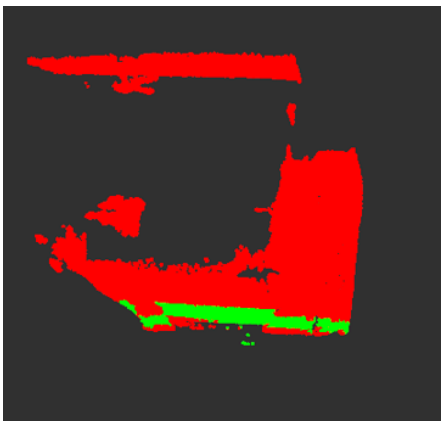


Figure 2.14: Point cloud with determine ground plane (green) and non-ground plane (red)

This process is quite time consuming, especially when dealing with iterations of upwards of 1000, and when large point clouds are present. Fortunately, once the ground plane has

been found, it won't change unless the system has been moved. This allows for the user to calibrate the system when starting, and store the plane information to calculate the hip height. The number of iterations and threshold are also configurable, to suit different processors.

2.4.3 Height Calculation

Using the pre-computed ground plane, and the returned pixel index for the hip, this sub-module is able to return the hip height of the cow. First, the pixel index needs to be converted into a 3D point:

$$x = (u - c_{x_{depth}})\frac{z}{f_{x_{depth}}}, y = (v - c_{y_{depth}})\frac{z}{f_{y_{depth}}}$$

$$P_{hip} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

where the Ground Plane is defined as:

$$T_{ground} = \begin{bmatrix} A & B & C & D \end{bmatrix}$$

The normal distance between the point and plane can be calculated using:

$$d = \left| \frac{Ax + By + Cz + D}{\sqrt{A^2 + B^2 + C^2}} \right|$$

By itself, this value may not be great, as there could be outliers, and the cow's hip moves up and down slightly as the cow walks. So, while the cow is in frame, all of it's hip height's are recorded and stored. When the cow moves out of frame, the outliers are removed, and the average is calculated.

The Cow ID and it's hip height are then sent to the UI by storing it within a .json file, and sending a socket signal to udpate the display.

2.5 Cattle Detection

2.5.1 Trade-off Table Post-1st-Report

Table 2.2: Model VS Data Categories Trade-off Table

Data Categories/Model	**K-means Clustering**	**Watershed**	**Pre-trained Model**	**Classic Contour Detection (Sobel/-Canny)**
RGB/HSV	Medium accuracy	Decent accuracy	High accuracy	Decent accuracy
Depth	Decent accuracy	Decent accuracy	Decent accuracy	Decent accuracy
Grayscale	Medium accuracy	Decent accuracy	Only take RGB	Decent accuracy
RGB + Depth	No data	No data	No data	No data

2.5.2 Development of Methodological Framework After the First Phase(Post-1st-Report)

Following the submission of the initial project report, the team undertook a systematic review of all previously implemented methods. Among the approaches that were independently designed and developed within the group, none achieved results that could be considered satisfactory. Their predictive accuracy was limited, and some exhibited unstable performance when exposed to diverse lighting conditions, body postures, or occlusions. Conversely, several existing and fully developed pre-trained convolutional neural networks—produced considerably stronger results. These established models demonstrated higher accuracy and more consistent inference. However, they lacked originality and adaptability, giving us very little room for methodological innovation or self-directed approaches. We believe that originality is important for our project, and the team sought to design a pipeline that balanced both accuracy and creativity.

The detection team decided to train our own DL model after discussion. This approach would enable the project to maintain the reliability observed in pre-trained architectures foster a customized, case-specific, and genuine methodological innovation. The decision marked a critical transition in our workflow, and re-defined the project from model adoption to one of model construction. This can help to enhance computational precision and demonstrate independent research capability.

To achieve this methodology shift, a preliminary pipeline was constructed and iteratively refined. The conceptual structure of this pipeline, which depicts the sequential data-processing stages through which raw cow imagery is transformed into measurable outputs.

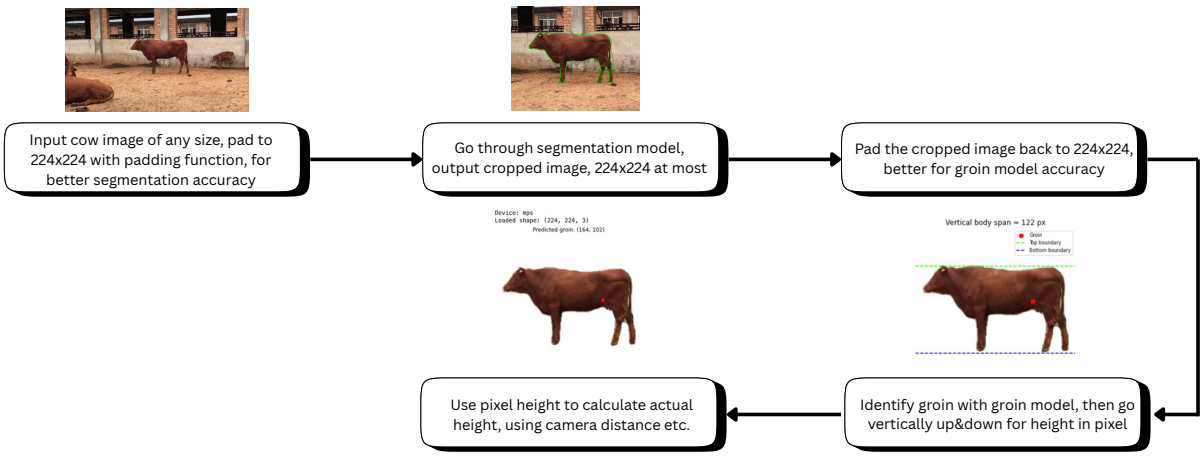


Figure 2.15: Cow Hip Height Detection-Measurement Working Pipeline

Each stage was designed to simplify the visual complexity of the raw images and progressively extract spatial information relevant to body structure and scale.

The general workflow can be described as follows, and shown in Fig 2.15:

1. **Data acquisition:** Cow images are collected under field conditions and stored for analysis.

2. **Pre-processing:** Each image is resized to a standardized resolution of **224×224** pixels to ensure consistency across the dataset and compatibility with the input requirements of deep learning architectures.

3. **Segmentation:** The standardized image is passed into a deep learning model trained for **cow contour segmentation**. This step isolates the cow from its environment, removing background elements such as fences, shadows, or grass.

4. **Cropping:** Based on the segmentation mask, the system automatically crops the cow region, effectively producing a clean image containing only the animal.

5. **Groin identification:** The cropped image is then processed by a second model designed to detect the **groin point**—a distinct anatomical feature located near the junction between the hind legs.

6. **Vertical projection:** From the groin point, a vertical line is projected upward to approximate the **hip position**. The top of this line corresponds to the upper body region around the hip bone, while the bottom intersects with the ground plane.

7. **Measurement:** The pixel distance between these two endpoints is calculated and converted into physical units, resulting in the final estimate of the cow's **hip height**.

The logic underlying this pipeline is both geometric and anatomical. The hip bone is the most direct and biologically accurate reference for measuring hip height. In practice,

it is very challenging to detect automatically, requires way more data than available or we can produce. The cow's groin or dorsal region, which particularly located in the corner between cow's back legs and the torso on side-view, providing sharp landmarks that a vision model could consistently identify. Variations in posture, coat pattern, and lighting conditions further obscure the hip region in side-view imagery. In contrast, the groin area forms a clear visual indentation and maintains a relatively consistent location across different individuals and poses. From an algorithmic standpoint, this makes it a much more reliable anchor point for automated detection.

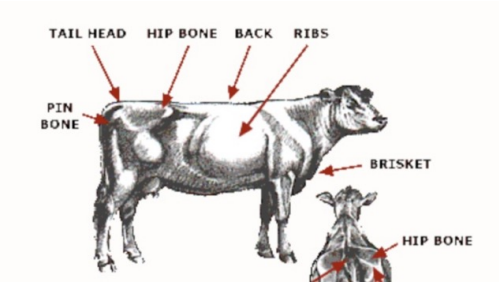


Figure 2.16: Bones and Points of Interest for Hip Height

By leveraging this observation, the team formulated a method: a vertical line extended from the groin point upward and downward would intersect two physically meaningful points—the upper body near the hip bone and the ground beneath the cow's hind limb. See Fig 1.16. These serve as critical points for calculating hip height. This approach, while simple in structure, captures essential anatomical relationships and allows the model to derive accurate spatial measures even without directly detecting the hip bone itself.

Based on this logic, the proposed method involves training two distinct yet complementary deep learning models:

1. **Cow Contour Segmentation Model** – responsible for separating the cow from the image background and generating the mask required for downstream processing.
2. **Cow Groin Point Identification Model** – tasked with detecting the precise coordinates of the groin point, which serves as the geometric reference for projecting the hip height.

Together, these two components establish a structured workflow that should be able to measure cows' hit height. The dual-model framework enhances the reproducibility of the approach, while also providing a foundation for further development in livestock measurement automation.

Training Data Creation

The successful training of deep learning models depends critically on the availability of large, high-quality datasets. In this project, two distinct datasets are required, one for the cow contour segmentation model and another for the cow groin point identification model:

1. The segmentation dataset required a collection of cow images captured from side views, with corresponding annotations outlining the external contours of each cow. Or to have the contour labeled in some way.

2. On the other hand, the groin detection dataset requires side-view cow images in which the precise (x, y) coordinate of the groin point was labeled.

These made the data we need very specific, and very rare to find online.

Following an extensive search through existing agricultural imaging datasets, academic archives, and open-source animal pose estimation collections, no publicly available dataset met the necessary conditions. Given that effective deep learning models typically require at least 2,000 well-labeled training samples for stable convergence and generalization, it became clear that the dataset would need to be constructed by us from scratch.

Therefore, data creation methods are constructed.

1. The first stage involved identifying a sufficient quantity of cow side-view images that could serve as raw data. After systematically scraping and curating images from multiple online and research-based sources, approximately **3,800 side-view cow images** were found. Each image was inspected to ensure adequate lighting, resolution, and visibility of the cow's full body outline, as partial or obscured views would negatively impact both segmentation and point labeling accuracy. The dataset was then organized and indexed to facilitate efficient retrieval and annotation.

2. **For the segmentation model** a semi-automated annotation process was adopted. Each image was passed through the previously found pre-trained **YOLO (You Only Look Once)** model, which will clearly detect the contour lines. A program was created to write coordinates of the points that formed the contour lines, along with the corresponding cow image name into the same row in a CSV file. Specifically, each row of the file contains the name of the cow image, and large amount of coordinates that forms the contour of cow. In other words, we created data that can train a model that can mimick the capability of YOLO model, that is specifically for this project(side-view). This process allowed the team to rapidly produce a large, reliable set of labeled segmentation data that could then be used to train the contour segmentation model. See Fig 2.17.

3. **For the groin point identification model**, first, cows will be segmented and has their background removed for smoother training process, then, a custom Python annotation tool was created. The tool consisted of a simple yet effective graphical interface that displayed one image at a time and awaited a single mouse click from the annotator. When the user clicked on the groin point of the cow, the program recorded the (x, y) coordinate of the click along with the corresponding image file name into a CSV file. This process was repeated for all 3,800 images, resulting in a comprehensive dataset of cow images paired with groin point coordinates. Eventually, the cows and their corresponding groin coordinates are stored in a CSV file. See Fig 2.18.

Figure 2.17: Segmentation Model: Contour Coordinates Display

Figure 2.18: Groin Model: Groin Coordinate Display

After creation process, all data are padded(not rescaled) in to 224×224 due to this is the proper size for best model training. The resulting datasets for both models exhibit a high degree of completeness and consistency. The segmentation dataset contains clearly defined cow contours, suitable for training a segmentation model, while the groin dataset provides accurate coordinate annotations that can be directly used as targets for the groin point detection model. Together, these two datasets not only enable effective training of the proposed deep learning pipeline but also represent a valuable original contribution, given the scarcity of structured and labeled livestock imagery in open datasets. The successful creation of these datasets laid a critical foundation for the subsequent stages of model training, validation, and performance evaluation.

Groin Model Training

Groin model training process can be properly carried out after dataset has been created. As previously mentioned 224×224 data images and corresponding groin coordinates are fed into the training program from the CSV file. The images are converted into PIL, then Tensor, then ImageNet normalization using appropriate mean and std values. The dataset is splited

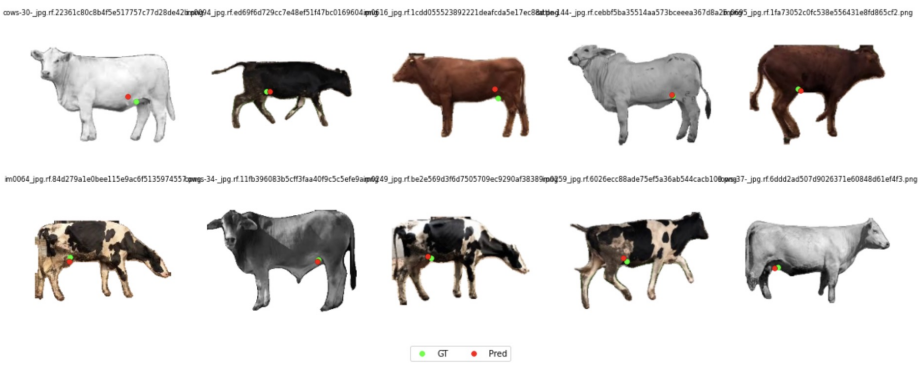


Figure 2.19: General Groin Model Performance

into 80-percent train and 20-percent test with fixed random seed. Batch size set to 32, with shuffling enabled for training.
The model architecture is based on a ResNet-50 backbone[5] initialized with ImageNet-pretrained weights. The original classification head is replaced by an identity mapping, and a lightweight regression head is appended consisting of a fully connected layer, ReLU activation, dropout ($p = 0.25$), and a final linear layer that outputs two continuous values corresponding to the normalized x and y coordinates of the groin point. The network is optimized using the Smooth L1 (Huber) loss, defined as:

$$\mathcal{L}_{\text{Huber}} = \begin{cases} 0.5(y_i - \hat{y}_i)^2, & \text{if } |y_i - \hat{y}_i| < 1 \\ |y_i - \hat{y}_i| - 0.5, & \text{otherwise} \end{cases}$$

No explicit data augmentation is applied beyond normalization, allowing the model to rely on the pretrained backbone for invariance.

Training proceeds for 10 epochs using GPU acceleration when available. Deterministic seeds are set across Python, NumPy, and PyTorch to maintain reproducibility. In each epoch, the model is set to training mode, gradients are computed, and optimizer updates are performed. After each epoch, evaluation is conducted on the test split using a pixel-space Mean Absolute Error (MAE) metric:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} (|x_i - \hat{x}_i| + |y_i - \hat{y}_i|),$$

This evaluation metric provides direct interpretation of error and aligns with the application's goal of identifying a single anatomically meaningful point.

The current epoch achieves a lower MAE than previously observed, and model weights are saved to disk. See Fig 2.19 for general groin model performance.

Segmentation Model Training

The segmentation model was trained using a manually annotated dataset in which each record of a CSV file contained an image filename and corresponding contour coordinates

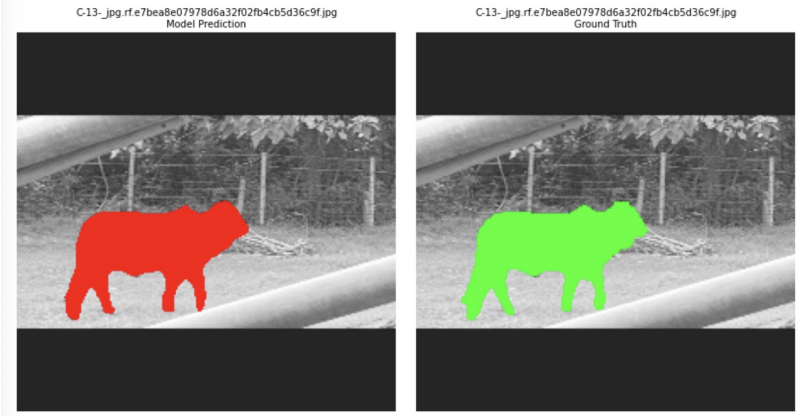


Figure 2.20: General Groin Model Performance

delineating the outline of a cow's body. Each RGB image was resized to 224 × 224 pixels and normalized using the ImageNet mean and standard deviation to ensure consistent input distribution. The coordinate arrays, stored as lists of (x, y), are converted into binary mask images representing the cow silhouette. These masks served as labels for a pixel-wise segmentation task. The dataset was divided into training and validation subsets using an 80/20 random split with a fixed random seed to preserve reproducibility.

The model architecture was based on a convolutional backbone enhanced with an Atrous Spatial Pyramid Pooling (ASPP) module. ASPP was incorporated to expand the receptive field of the convolutional layers by applying with varying dilation rates. This allows the network to capture multi-scale contextual information while preserving spatial detail.[2] The encoder extracts feature maps from the input image, and the ASPP module aggregated context across multiple scales before up-sampling to produce dense segmentation outputs.

The training objective employed a binary cross-entropy loss between predicted and truth-masks:

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right],$$

where y_i and $\hat{y}_i$ denote the true and predicted pixel labels, respectively. The training loop was executed for 25 epochs, alternating between training and evaluation phases in each iteration. During validation, coefficient metrics were computed to assess segmentation accuracy and overlap consistency between predicted and annotated masks. Finally, some qualitative assessments were conducted by overlaying predicted contours on the original images, confirming the model's ability to delineate cow boundaries with high spatial precision.

Result of general model performance shown in Fig 2.18.

2.5.3 Testing Outlines

Evaluation Metrics

To evaluate the effectiveness of our final program, here is our current evaluation metrics.
• Segmentation Accuracy: Accuracy of segmenting cow out of an image.

• Robustness: False positive rate in backgrounds.
• Measurement Accuracy(Next stage): Accuracy of hip measurement of cow.
• Time Efficiency: Average runtime per image(frame).
• Resource Usage: Memory footprint, GPU/CPU load.
• Scalability: Performance on large datasets.

Further Trade-off Analysis

A summary table can be presented to illustrate the model performances, and details will be discussed in "Quality of Work" section.

Model	Architecture	Evaluation Baseline	Accuracy	Performance Comment
Segmentation Model	ResNet50 + ASPP (DeepLabV3)	YOLO (treated as ground truth)	86.88%	Strong overlap; robust in controlled tests
Groin Keypoint Model	ResNet50 + Regression Head	Manual keypoint ground truth	86.0%	Reasonably accurate, with slight x-axis offset

Table 2.3: Summary of model architectures and evaluation performance.

Acceptance Criteria

The primary acceptance criterion for this project is that the final hip measurement of the cow must achieve an accuracy with an error margin of no more than 3 cm. All detection, segmentation, and depth estimation methods will be evaluated against this benchmark. Regardless of variations in environment, lighting, or background complexity, the system will only be considered successful if the overall pipeline consistently produces measurements within this 3 cm tolerance.

2.6 User Interaction

The user will largely interact with the system through the front-end interface. The front-end interface is designed to be user friendly and display relevant information in multiple different formats depending on the user needs.

2.6.1 Testing

A combination of unit testing and system testing has been performed on the front-end.

- Positive case 1: Landing page loads

- Positive case 2: Cow detail page loads
- Positive case 3: CowCam page loads
- Positive case 4: Sorting works for all columns
- Positive case 5: Searching works for all columns
- Negative case 1: Unknown cow ID gracefully fails
- **Negative case 2: Corrupted push gracefully fails**
- Edge case 1: Update of cow pushes live
- Edge case 2: Update of cow with partially identical attributes works

2.6.2 RealtimeKPI testing

And data tests were conducted on the realtime dashboard page to measure the changes in KPIs. First, we write a program on the back end to simulate using random data. Finally, the KPI fluctuations of virtual data were successfully displayed in several metric boxes on the website page, completing the first preliminary test. Secondly, we had the front end read the cow-data.json file and directly write the assumed variable parameters into the json file for testing instead of randomly generating them from the back end. All of these can be successfully recognized automatically by the dashboard.

2.6.3 Technology stack

The technology stack used for the front-end consists of Flask, Socket.IO, JSON, Jinja, OpenCV, and Bulma

Flask

Flask is a web application framework that is written in Python. It was chosen as the framework because it was lightweight, provided good integration with the other technologies used in the stack and the use of Python enabled simple file interactions, eliminating the need for a dedicated database.

Additionally, the developer was already familiar with the development of applications within Flask and had deployed it successfully both for University assignments and also private projects.

Socket.IO

Socket.IO is a JavaScript library that enables real-time, bidirectional communication between web clients and servers. It was chosen for its ability to handle event-driven communication efficiently, enabling the front-end to show new data without having to constantly poll the

server. Its compatibility with Flask and Python integrations further supported its use in the project stack.

Within the project, it enables the pushing of new cow data to the front-end as it is written to the JSON file. Without Socket.IO, the server would have to constantly poll the file for changes and force refreshes, which is not ideal as it does not update in real time and also increases the computational power required.

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for both humans to read and machines to parse. It was selected for its simplicity in structuring data and for its widespread use as a standard in web communication. JSON's text-based format allows it to be easily stored in a file, eliminating the need for a database and keeping the overall size and complexity of the front-end down.

Jinja

Jinja is a templating engine for Python that allows developers to embed dynamic content into HTML pages. It was chosen as it integrates directly with Flask, enabling the efficient separation of application logic and presentation.

Some key features that are very useful are the integration with DataTables, enabling the dynamic rendering of the landing page, as well as templates enabling the creation of individual cow detail pages without having to duplicate code.

Bulma

Bulma is a modern CSS framework based on Flexbox. It was chosen for its lightweight design, responsive grid system, and ease of use in rapidly creating clean, consistent front-end layouts. It is used to add styling and formatting to the web pages.

2.6.4 Reflections

Our Client wasn't too concerned about the specifics of our architecture, he is very happy with us determining the best way forward ourselves. The only request he had that related to this part was that the camera be side facing. This influenced our decision making in method for determining the hip height point leading us to consider a neural network approach from the side as this would only be possible from this angle, a much lower risk approach. **Self-Reflecting** as could be seen the group kept the overall structure of the system the same but provided greater layers within separate sections to create a successful design. Overall the group moved from exploratory stages to finalising strategies which are evident above.

Chapter 3

Quality of Work

No.	User Story	Acceptance Criteria	Testing Cases		
			Normal Testcase	**Boundary Testcase**	**Abnormal Testcase**
1	US04 - Plug and Play System	Voltage > 5V, System powers on	Wall Plug Connected	N/A	N/A
2	US06, US07 - RGB Alignment	Visual check for edge alignment	Cow at 2-3 meters, no obstructions	Object at limits of depth map (close and far)	Occlusion and non-reflective surfaces
3	US01, US07 - Cow Detection and Segmentation	IOU > 80%	Image with cows present	Varying image resolution	No cow present, or partially in view
4	US03 - Groin Detection	RMS < 20 pixels	Segmented cow	Varying image resolution	Poor segmentation
5	US02, US05, US08, US09 - Dashboard UI	latency < 1s	Input new data and livefeed imagery	Induce latency spikes	Corrupted Data Structures
6	US01, US10 - Integrated System	Height in range 1.3m - 1.5m	Cow present in image	N/A	N/A

Table 3.1: User stories, acceptance criteria, and corresponding test cases for each model.

3.1 Electrical Testing

The project at hand required the development and procurement of electronic components, which like software, must be tested. All hardware components are commercial-of-the-shelf (COTS) and it is assumed that all internal sub-components (e.g. a circuit inside the camera) have been fully tested by the manufacturer. Conducting similar testing ourselves would void product warranties and may risk damaging the component. Testing is thus conducted in an end-to-end fashion.

The electrical system's sole objective is to provide a 5V power supply to the Raspberry Pi 5 from a 12V input source. This is achieved using a DC-DC buck converter, which steps down the voltage. The output voltage of the buck converter is measured using a multimeter and shown in figure 3.1.

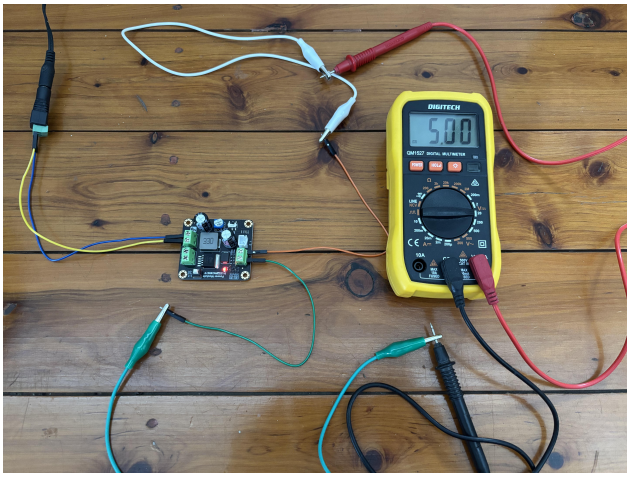


Figure 3.1: A multimeter measuring a 5V output voltage from the DC-DC buck converter.

The output is measured at the required 5V. Importantly, when the 5V output of the buck converter is used to power the Pi, it powers on and boots correctly, shown in figure 3.2a. The NYX660 camera similarly powers on when provided the 12V directly, shown in figure 3.2b.

(a) Raspberry Pi 5 successfully booting.

(b) NYX660 camera successfully booting.

Figure 3.2: Components powering on.

3.2 Perception Testing

3.2.1 RGB Alignment

Normal Testcase: Cow Within Range and No Obstruction

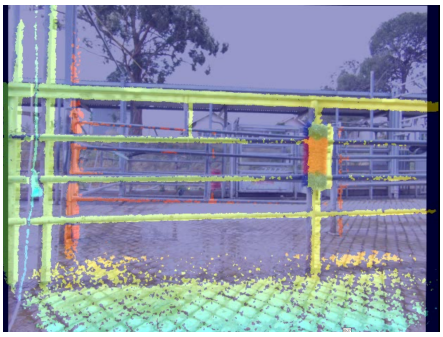
This test case appears when the system is setup at approximately 2-3 meters from the object being measured. This is considered the normal operation of the system, and is also in an ideal environment (indoors, no obstruction and good reflective surfaces).

(a) Pre-computed alignment mapping

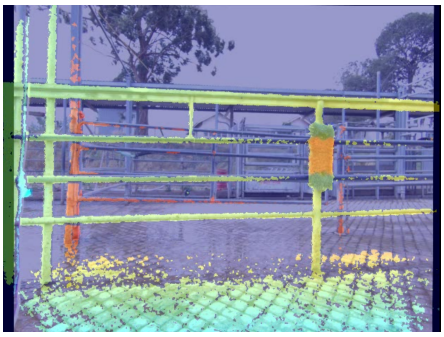
(b) Live computed alignment mapping

Figure 3.3: Normal case of operation

As seen in both figures 3.3a and 3.4, the RGB image is able to align very well in both pre-computed and live methods. The overlayed depth map completely covers the RGB image, with no visible offset; however, this was not the original case.



(a) Pre-computed alignment mapping (unshifted)



(b) Pre-computed alignment mapping (shifted)

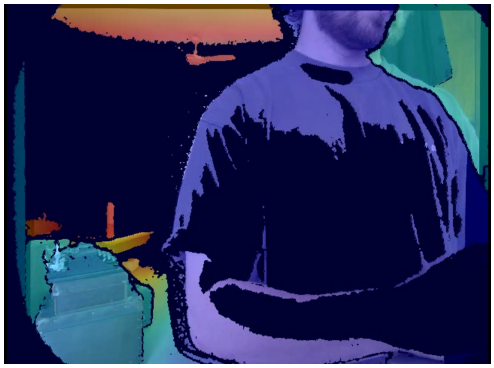
Figure 3.4: Pre-computed mapping fix

During initial testing for the pre-computed map, the RGB image seemed shifted to the left in comparison to the depth map, which can be seen in figure 3.4a. This is due to the pre-computed depth map being computed at a flat 1 metre surface, while typical operation occurs at 2-3 meters. This was overcome by shifting the RGB image to the right by 8 pixels, resulting in a perfectly aligned image in figure 3.4b.

Boundary Testcase: Object at Limits of Depth Range

This test case is for when operating at the limits of the nyx660's depth capabilities. From the NYX660 ToF Camera Specification, it's depth range is between 0.3 - 4.5 meters, although this also depends on reflectivity of the measured object. This test case occurs for both alignment methods (pre-computed and live) by having a person stand really close, and far away from the camera.

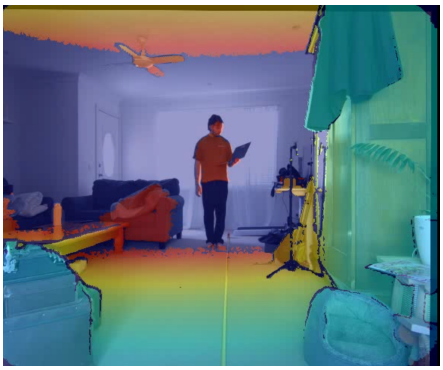
(a) Pre-computed alignment mapping



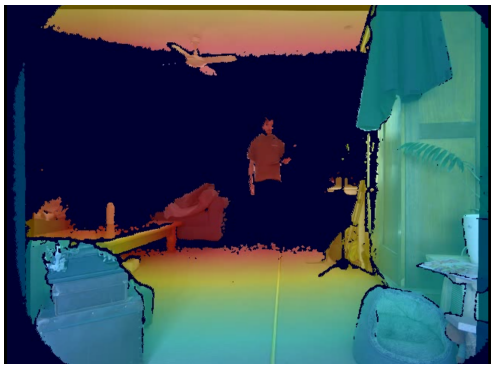
(b) Live computed alignment mapping

Figure 3.5: RGB alignment to depth while object is very close.

When the object is very close, we can see some mis-alignment for both methods. In both cases, the RGB image seems shifted to the left, in comparison to the depth map. It can also be seen, that since the object is too close, the depth map fails to receive any valid values, which is why there are dark patches.



(a) Pre-computed alignment mapping



(b) Live computed alignment mapping

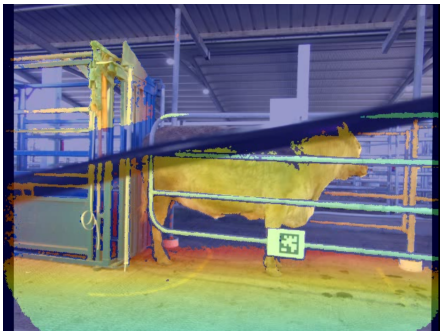
Figure 3.6: RGB alignment to depth while object is very far.

The further away the object, the better the alignment appears to be. The depth map seems to completely encompass the object; however, since the object is so far away, there are similar issues where the depth map struggles to read valid data.

Since raw depth values can't be estimated if objects are too far away or too close, the ideal solution to this is ensuring the setup is roughly 2-3 meters away from the object being measured.

Abnormal Testcase: Occlusion and Non-Reflectie Surfaces

There are two situations that have been encountered that causes abnormal behaviour with the RGB Alignment, and that's due to occlusion, or poorly reflective objects. Occlusion occurs where one of both of the cameras are partially covered, which causes the two sensors to see two different things.

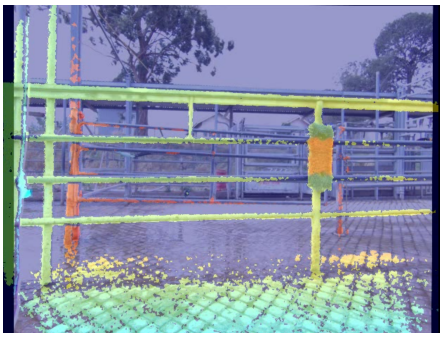


(a) Pre-computed alignment mapping

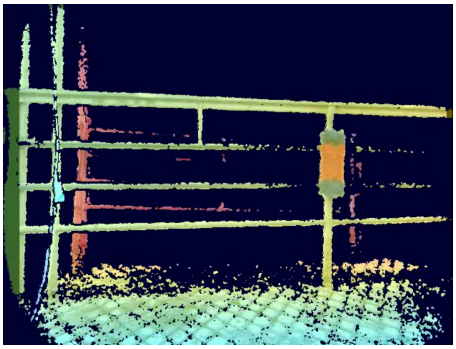
(b) Live computed alignment mapping

Figure 3.7: Partially occluded RGB image and depth map.

As seen in figures 3.7a and 3.7b, there was a cable in front of the sensors. Both sensors are able to see different things around the cable; however, this can cause a mismatch in information. When segmenting and finding the hip point, it's possible that the image point is not a valid depth point. This was prevented by checking if the point was had a valid depth, and if it didn't, shifting the point downwards until it reach a depth value.



(a) Pre-computed alignment mapping



(b) Live computed alignment mapping

Figure 3.8: Poorly reflective surfaces.

Figures 3.8a and 3.8b show an outside environment where the ground is wet. Since the Time of Flight (ToF) sensor is unable to receive back all of it's data points, it results in the ground being incomplete and broken. This can also affect the cow's being measured, if the cow is wet, or their coat is particularly unreflective, it can result in incomplete depth data. To prevent this issue, the height is taken and stored for the duration of the cow being in frame, increasing the chance of valid depth points.

3.3 Cow Detection Testing

3.3.1 Cow Contour Segmentation Model Evaluation Approaches

Evaluation via Mask Overlap Against YOLO Segmentation Baseline

This evaluation approach feeds cow sample images through the pre-trained YOLO model and our own segmentation model respectively. Then, it overlays the result of our own model on the result of YOLO model and calculate the percentage of overlay, that is how much of the segmentation model's result agreed with that of YOLO model. This is because, since the training data is produced by YOLO model to begin with, our model is just mimicking YOLO model's capabilities and case-specific to what we are doing. Therefore an evaluation method like this is valid.

Images used for this section were of extremely diverse environments: indoors, outdoors, rainy, large backgrounds, extremely diverse lighting conditions. This was to test that the model performed generalised well amongst all conditions (given the testing data was also extremely diverse this was an expected outcome).

The segmentation model achieved a Mean IoU of **86.88%** (Median **89.51%**), with a Mean Dice score of **92.55%** (Median **94.46%**). Additionally, the model exhibited a Mean Coverage of **93.42%** relative to the YOLO reference masks.

Overall, our segmentation model yeilds average of **86.88%** accuracy. It is performing fairly well, as it usually overlaps the correct cow shape closely, with clean boundaries and little extra background included. Example illustration see Fig

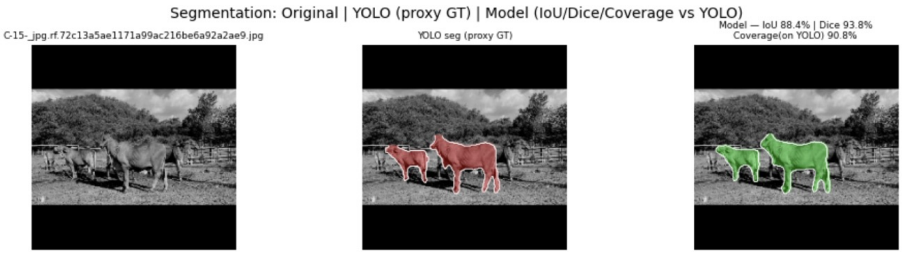


Figure 3.9: Segmentation Model Evaluation Example Illustration

Boundary Testcases: Resolution

Tests were also completed to evaluate the segmentation model under different Resolutions of cow images. This was important as we were not completely sure how far the camera would have to be from the cow as determined by various environmental factors mostly determined by the cattle farmers. This would force the definitions of the cows to reduce considerably and not take up the entire 1080p frame.

The segmentation training was all done on a fixed size of 224×224 pixels. Therefore when images of differing resolutions were fed directly into the model, performance degraded sharply, making it necessary to apply a padding-and-resizing preprocessing function before inference to this original 224 by 224 size. Therefore the highest resolution of the image was

by default 224 by 224. Smaller images are sized up to this size as well by the algorithm. Testing on how well these smaller sizes performed was done and the results can be seen below.

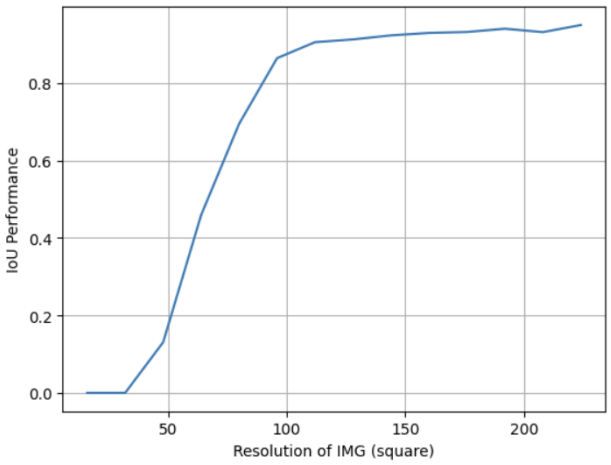


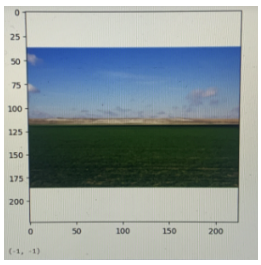
Figure 3.10: Resolution vs IoU Performance for Segmentation

As can be seen the segmentation performs very strongly above 100 by 100 resolution which would allow only 1/10th the size of the entire frame to be segmented accurately. Thus the system performs extremely well here.

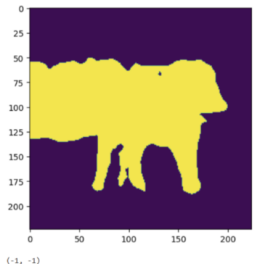
Abnormal Testcase: No Cow in Field of View or Cow only Partly in Field of View

The segmentation was tested to check if it could determine if there were no cows in the view and if so it should return (-1, -1) to represent this. Testing was completed with 100% accuracy for these images. An example is seen below with a return of -1,-1.

Testing was also completed for cows that were not fully in the image, and it was decided that the same return value should be given. Whilst segmentation did occur, it didn't fully complete as seen below with a corresponding return again of -1, -1 as desired.



(a) Empty Image Testing



(b) Cow on Edge of Frame

Segmentation model degradation in field conditions

Due to unforeseen events and issues collecting real world data with our client, we were unable to obtain field performance evaluation results until very late. Further due to limited time our client was only able to provide us with images with steel bars in front of the cows which we were informed beforehand we could assume wouldn't be an issue.

Although the segmentation model demonstrated high accuracy during controlled testing, its performance in the field was considerably weaker. In the live environment, the model frequently failed to correctly segment cows, which led us to revert to using the YOLO segmentation model as a fallback strategy. A likely contributing factor is the presence of metal bars and enclosure structures in the field setting, which were not represented in our training dataset. The segmentation model had only been trained on clean, unobstructed side-view cow images, and therefore lacked robustness toward visual occlusions. This suggests that training data variation—specifically the inclusion of images containing enclosures, fences, and partial obstructions—would be necessary to improve generalization and prevent this form of failure. But if the results are given to us earlier as they are supposed to, we should be able to fix this issue. Average accuracy among these images were 55% however led to key parts of the cow being lost as seen below.

Figure 3.12: Field Conditions With Bars Segmentation

3.3.2 Cow Groin Detection Model Evaluation Approaches

Evaluation via Keypoint Distance to Manual Ground Truth

In this approach, cow segmentation samples are fed to the groin model. Then, the predicted groin point and actual groin point from the CSV file will both be plotted on the corresponding image. Here, the difference in pixels between the predicted and the actual one are calculated. Along with the fact the all image fed into model are in 224×224, accuracy percentage can be calculated from the set pixel values.

Cow images inputted were segmentations using Yolo of the diverse image types used in the cow segmentation testing. One thing to note however was images were all side on (given it is impossible to identify the groin from front on or top down images). This was also a reasonable assumption given our system would be set up in a cow race where cows are only able to walk in a single file and in one direction. Cow images were diverse in the angle they were facing where some would be more diagonal than side on and some had very different head positions and body postures.

The groin keypoint model achieved a Per-axis Mean Absolute Error of **43.45 px** in the horizontal direction and **5.01 px** in the vertical direction, corresponding to an average per-axis error of **24.23 px**. The overall spatial discrepancy between predicted and ground-truth keypoints, measured using Mean L2 distance, was **44.21 px**. When normalized by the diagonal length of the image, this yields a regional localization accuracy of **86.0%**, indicating that the model consistently identifies the correct anatomical region, but displays a systematic horizontal offset.

The groin keypoint localization model demonstrates strong vertical accuracy, with a mean y-axis error of only 5.01 pixels. However, the x-axis error is substantially higher at 43.45 pixels, indicating a systematic horizontal offset in predictions. The overall spatial distance between predicted and true points is 44.21 pixels on average, corresponding to a normalized regional accuracy of 86.0% relative to the image diagonal. These results suggest that the model generally identifies the correct region of the cow but requires correction in the horizontal localization alignment.

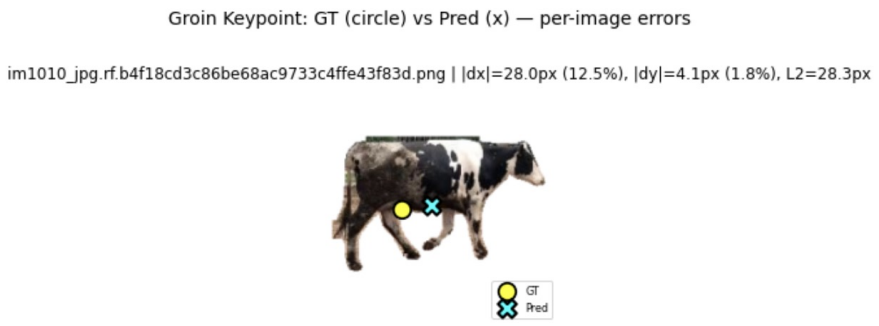


Figure 3.13: Groin Model Evaluation Example Illustration

Boundary Testcase: Resolution

As with the segmentation model, the groin location model also resizes images to 224 by 224 using padding so that it has a stable input type. So whilst images above 224 by 224 size would be transformed to the same resolution, any below this would likely cause reduced performance as information is lost. The testing of the effect of resolution on performance can be seen below.

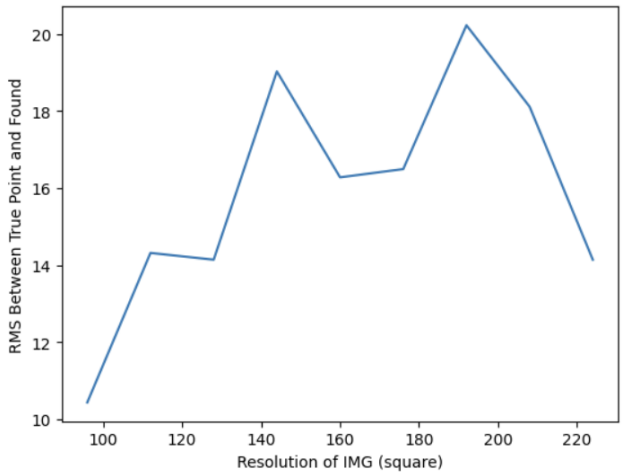


Figure 3.14: Effects on Resolution on RMS of Estimated vs True Groin Locations

The testing shows that the performance really doesn't change that much due to the resolution and therefore demonstrates that the groin model is fairly resolution robust. This is again important given that cow images may be small with the camera being potentially far away from the cows that are being captured.

Abnormal Testcase: Poor Segmentation

Whilst the pipeline doesn't look at cows that are only partially in the frame, sometimes a similar cut out cow may occur due to incorrect segmentation occurring as seen above 3.12. The effect of this on the groin model was tested.

(a) Bad Segmentation: Working Groin Model

(b) Bad Segmentation: Not Working Groin Model

Here it can be seen the groin accuracy is very heavily dependent on the quality of the segmentation. It doesn't do too badly especially in the x direction however when the segmentation doesn't even have the groin point in it it does become impossible. Hence the need for the segmentation model to be better is much more important than trying to get the groin detection to operate with poor segmentation. Given the groin point is going to be projected up the y direction isn't too important, therefore the model actually does quite well despite great limitations from segmentation.

3.3.3 Model-pipeline Performance in Practice

Sensitivity to input resolution and scale invariance limitations

Both the segmentation and groin keypoint models exhibited sensitivity to input resolution. The original training pipeline assumes inputs of a fixed size of 224×224 pixels. This highlights a structural limitation: the models are not scale-invariant. Expanding the dataset to include multi-resolution inputs or retraining using architectures that inherently support variable spatial dimensions (e.g., U-Net variants or fully convolutional models without fixed-size linear heads) may improve robustness during deployment.

3.4 Dashboard Testing

3.4.1 Dashboard Testing – Backend Data Communication

The backend testing of the Cattle Realtime Dashboard focused on verifying the reliability and accuracy of data transmission between the JSON-based data source and the front-end interface. The testing aimed to ensure that the system could continuously update performance metrics such as FPS, IoU, latency, ID confidence, and point resolution without interruption or data loss.

During testing, multiple JSON samples were created and integrated into the /json directory to simulate live data input from the detection model. The dashboard's fetch() function within the Flask/Jinja template was evaluated for its ability to retrieve and render new values every few seconds. Particular attention was given to response timing, data consistency, and handling of edge cases such as delayed or malformed JSON responses.

To validate real-time behavior, the testing involved intentionally modifying JSON values and observing instant updates on the dashboard. Scenarios such as sudden frame-rate drops (FPS less than 5), high latency spikes (more than 300 ms), and corrupted data structures were injected to confirm that the dashboard maintained functionality and displayed fallback values instead of crashing.

Integration testing further verified that the API endpoints correctly served updated data to the front end, with no caching issues or synchronization delays. The system's resilience was confirmed through repeated refresh cycles and cross-browser validation, ensuring that both Chrome and Edge could render the real-time data flow smoothly.

Overall, backend testing demonstrated that the Cattle Realtime Dashboard can effectively handle dynamic JSON inputs and maintain real-time updates with minimal latency, proving its suitability for deployment in continuous monitoring environments.

3.5 Integration Testing

Integration encompasses combining the separate processes to enable a picture of a cow to be used to determine it's hip height, and be displayed in the website. Since each module had been tested rigorously, integration and upwards testing was conducted here on the separate

processes.

In the test case below, the cow can easily be seen through the race with no obstruction. The cow is light in colour, and the depth map is full. When originally testing, we found that sometimes the identified hip point was a non-valid depth value. This resulted in the hip height being zero. Due to our extensive use of logs, this was easily identified, as seen in figure 3.16:

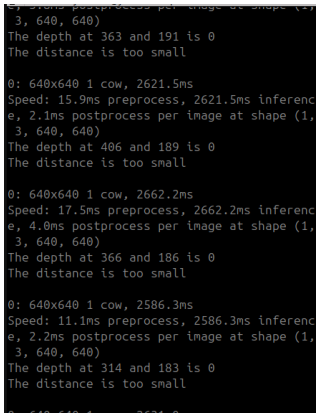
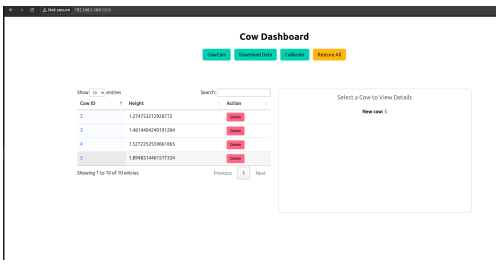
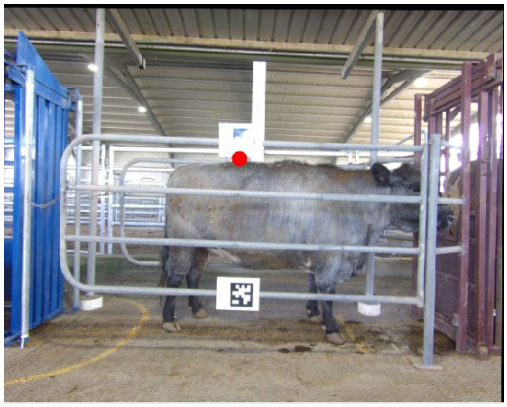


Figure 3.16: Logging information showing hip height value is too small

The hip point algorithm was changed to ensure it always returned a valid depth point. With this issue fixed, the integrated system was able to determine the hip height, as seen in figure 3.17



(a) Height information sent to website for visualisation



(b) Hip point detected on cow

Figure 3.17: Integrated cow detection and hip measurement

The collected data was then automatically streamed to the website for visualisation in real time, seen in figure 3.17a

Chapter 4

Quality of Group Processes

4.1 Allocation of Group Roles

Name	Allocated Work
Daniel Mooney	Manager: Organised and documented group meetings as well as overlooked all other roles and project sections.
	Hardware and Perception.
Xavier Singarayar	Doomsayer: Ensured any issues or potential downfalls of the project were dealt with
	Cow Detection.
Wentao Gao	Programmer: Kept an overview of programming tasks to ensure they were kept up to date and of appropriate quality
	Cow Detection
Jackson Bamber	Client Liaison: Dealt with all communication with the client and other stakeholders
	Perception (primary) and Cow Detection.
Chengye Wan	Tester: Produced atomic and integration tests to determine the efficacy of the completed work
	Dashboard (Primary) and Cow Detection.
Andy Xie	Tracker: Determined a suitable timeline for project outputs and kept progress
	Perception and Dashboard
Kivaan Mudaly-Naidoo	Tester: Produced atomic and integration tests to determine the efficacy of the completed work
	Hardware and Perception

Roles were kept constant during the project due to the complex nature of the task requiring stability and specialisation. Whilst members were primarily focused on certain their given task, fluidity to help others when workload was low was an expectation and many followed through.

4.2 Evidence of Collaboration

4.2.1 Meeting Minutes

The last two meeting minutes are included in the appendix .1 for evidence of collaboration. All meeting minutes are available on GitHub. Meeting minutes keep track of general notes, contain the plan for what needs to discussed and allows for the team to set tasks to be complete before the next meeting or by a given deadline. Meetings on Monday served as a good time for status reports to both the tutor and to the rest of the group.

4.2.2 Client Communication

The client communication has been completed mostly through emails as explained in the Group Interactions section of the introduction, completed by our customer liaison. Communication has been clear and to the point to not overload our client. Whilst communication has been ongoing there were a couple of setbacks with interacting with the client. This included issues with getting our hardware ordered on time, issues with collecting our data in a timely manner and issues with returning hardware. Whilst these were all resolved, it did delay many timing deadlines ultimately leading to extreme catch up required by our team. Key client communication can be seen in the Appendix and the client demo was recorded in meeting minutes. Client demo served to provide feedback which has been mentioned throughout report.

4.2.3 Management Tools

GitHub

The team has utilised a GitHub organisation where multiple repositories can be constructed which is useful given our clear division of sections in our project. The GitHub has been made private so that the software is protected from external parties. Further the code samples are downloadable to all members of the team which allows for them to utilise other group members code and make controlled edits using version control. The final integration repository was called CowPal which contains all the required code for the final product, the front end was also placed separately in the front end repository. The repository list can be seen in the appendix and largely houses testing for each section. This can be seen in the appendix .3. Especially with the final integration version control was utilised to ensure no writing over of shared work, this is evidenced in the appendix .3.

Slack

The project Slack has been used for organisation and record of discussion. The team has utilised channels to split up different discussions and share resources. This aids in separating conversations about different topics and keeping retrieval of information easy. The channels that exist are the documents general (where we keep a folder of important miscellaneous documents including assignment information), the general channel (used for messaging to the whole group), the learning resources chat (for research and sharing of skill learning resources), a channel for each team (sharing information and collaborating on tasks), and a share links channel (contains folders of meeting minutes and datasets). This organisation and an example of a chat history can be seen in the appendix .4.

4.3 Critique

4.3.1 Effective Group Processes

- Splitting up the project into clear technical tasks has allowed for individuals to specialise and research early on so that they can fufil their technical role
- Having the Monday Meeting mainly aimed at discussion among sections have allowed for room for collaboration taking advantage of the only face to face meeting
- The well organised clear channel topics in slack have allowed for smooth navigation of different sections of the project and important documents
- The team has gotten along well for the most part and are able to discuss topics fairly freely without feeling uneasy
- Teams attendance at meetings and tutorials have been strong with valid communication during absences and strong efforts to make meetings

4.3.2 Issues and Fixes

- Wednesday Meeting was underutilised due to it being so close to other meeting. As it is the only time throughout the week that everyone can meet regularly, we kept it, however when large deliverables were due we organised additional workshop sessions where everyone met up and did work for integration and collaboration.
- Lack of regular meetings with client reduced both accountability and guidance. The team managed to stay in contact largely through email but this wasn't always best communication with client missing things we said. However, due to our constant communication, we did have an understanding of the client expectations.
- There were uncontrollable delays in obtaining hardware and then subsequently data, where we could not obtain data ourselves from the farm. In spite of these challenges, the team did well to get lots of work in preparation for data and hardware, however ultimately this did force some elements to be rushed.

Chapter 5

Conclusions

5.1 Risk Analysis and Management

During the course of the project, several challenges and risks were encountered. The two risks that most heavily affected the project was hardware lead time, and inability to be on-site during data collection.

Due to delays in shipping for the NYX660, this drastically slowed down our full system integration. Hardware assembly couldn't be started, and the Perception software couldn't be validated. We overcame this by focusing on Cow Segmentation and the UI, while planning for hardware integration as soon as the hardware arrived. For future projects, we'll need to make a bigger push on sourcing hardware so we aren't as affected by delays.

The second issue was due to data collection. When the hardware arrived, we set up a recording system for the client so he could plug it in and not worry about it. Unfortunately, we didn't realise that the formatting for the harddrive prevented any files to be saved that were larger than 4GB, which resulted in a loss of 2 hours of recording. After this incident, we re-formatted the harddrive and sent it back out for testing; however, this resulted in a weeks delay due to no hardware or data. Next time, all systems should be more thoroughly tested, and for longer durations, to ensure that everything works as intended.

5.2 Limitations

The work presented in this report has acheived end-to-end functionality, by incorporating each subsystem: Hardware, Perception, Detection and Dashboard. Each module has been tested in isolation, and integrated together. Despite this, there are still some limitations to the entire system.

Unfortunately, the team was unable to acquire an RFID scanner, to detect when a cow was present and obtain it's ID. This means, the current method to detect a new cow, is by waiting for the current cow to leave the frame, and then a new cow to enter. The corresponding ID is just a counter for the number of cows gone by. The system must also be placed in such a way that it has a large view of the ground, otherwise it will fail to find the ground plane and give false readings.

5.3 Primary Strengths

Despite it's limitations, the system has multiple strengths that can make it robust in practical, real-world environments. The methods used to extract measurements and segment cows are robust against noise and incomplete depth data. Running the website locally minimizes downtime, and allows for flexible usage in any location. The system can also handle scenarios where no cow is present, ensuring no false hip heights reported.

5.4 Programming Practices

This project utilised a hybrid approach, combining parts of Extreme Programming with delegated sections and module leads. This hybrid approach allowed for each team member to specialise within their delegated module, while also contributing to the entire team through their XP role. Given the size of the project, this approach was beneficial, as it allowed for parallel development for each module, while the XP Practices helped maintain code quality across all modules.

Version control was managed with a GitHub organization, which allowed for each module to be constructed in isolation before integration. This approach reduced common issues, such as version control, while still enabling multiple team members to contribute to the same module when necessary, allowing for collaborative progress.

Coding and documentation standards enhanced maintainability between modules, while also allowing us to share documentation with the client during recording sessions.

Overall, this hybrid approach allowed for collaboration and iterative design, while keeping people responsible for specific modules. This method allowed the team to produce a robust system, which was able to meet our functional, and non-functional requirements.

Bibliography

[1] AC Infinity Inc. Ventilation dust filter kits specification sheet, 2024. URL https://acinfinity.com/. Accessed: 2025-09-06.

[2] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587, 2017. URL https://arxiv.org/abs/1706.05587. Accessed: 2025-09-06.

[3] S. Choi, J. Park, J. Byun, and W. Yu. Robust ground plane detection from 3d point clouds. In *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, pages 1076–1081. IEEE, 2014.

[4] Core Electronics. Dc-dc power module 25 w product page, 2024. URL https://core-electronics.com.au/dc-dc-power-module-25w.html. Accessed: 2025-09-06.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. URL https://arxiv.org/abs/1512.03385. Accessed: 2025-09-06.

[6] B. Littler. Live cattle assessment, 1970. URL https://www.dpi.nsw.gov.au/animals-and-livestock/beef-cattle/appraisal/publications/live-cattle-assessment. Accessed: 2025-09-06.

[7] B. McKiernan. Frame scoring of beef cattle, 2007. URL https://www.dpi.nsw.gov.au/animals-and-livestock/beef-cattle/appraisal/publications/frame-scoring. Accessed: 2025-09-06.

[8] Polymaker. Polylite abs 3d printing filament technical data sheet, 2023. URL https://polymaker.com/product/polylite-abs/. Accessed: 2025-09-06.

[9] Raspberry Pi Ltd. Raspberry pi 5 technical specifications, 2023. URL https://www.raspberrypi.com/products/raspberry-pi-5/. Accessed: 2025-09-06.

[10] Terabee S.A. Nyx-660 time-of-flight camera datasheet, 2024. URL https://www.terabee.com/shop/time-of-flight-cameras/nyx-660/. Accessed: 2025-09-06.

Appendix A: Meeting Minutes

20/10/2025

Attendees: Andy, Chengye, Jackson, Kivaan, Xavier, Wentao, Daniel

Agenda:

- Integration
  - Waiting on recorded data
  - Do all you can with interfaces between
- Perception
  - Hardware currently being used by Luciano
  - Wrappers for ROS
- Segmentation
  - Ensure tested and works holistically
- Dashboard
  - Need to import CSS library to be local rather than fetch from a CDN
  - Small redesign of page - collapse detail view, include more filtering, add delete button
  - Work on integration between dashboard and ROS - Andy and Jackson

Notes:

- Group report and presentation slides due end of Week 11
- Peer review due Monday Week 12

Deliverables:

Deliverable	**Responsible**
Report Write Up	Everyone
Final Integration	Everyone
Testing	Everyone

13/10/2025

Attendees: Daniel, Jackson, Kivaan, Xavier, Chengye, Andy, Wentao

Agenda:

- Testing prototype run down
- Review document for Luciano
  - Ensure clarity and logical structure
  - Verify ease of understanding
- Section updates
- Plan for rest of semester
  - Hardware + Electronics upgrades
  - Detection integration (once data collected)
  - UI integration
  - Decide on message passing interface
- Detection + UI Workshop
- Minimum final requirements

Notes:

- Group report due end of Week 11
- Team members have their own hard drives to store and share bag files

Deliverables:

Deliverable	**Responsible**
Ask Luciano to take photos of camera rig in the wild	—
Find storage solution for bags collected by Luciano	—
Research/code callbacks in Python	Andy
Research code profiles	Xavier and Wentao

07/10/2025

Attendees: Daniel, Wentao, Andy, Chengye, Xavier, Kivaan, Jackson

Agenda:

- Section updates

- Last sprint + data collection

Notes:

- Download function added to dashboard
- Andy to research ROS
- Dashboard integration to be conducted as group coding session
- Working cow segmentation model developed in-house
- Hip detection model operational
- Check memory constraints for storing model weights on Raspberry Pi
- Feedback for Luciano to validate data collection functionality

Deliverables:

Deliverable	**Responsible**
Hardware parts order spreadsheet + mount	Hardware Team
Hardware handover	Hardware Team
Prepare wrappers for integration	Software Team

Appendix B: Client Communication

Figure 1: Communication: Directions for Hardware Setup Email 13/10

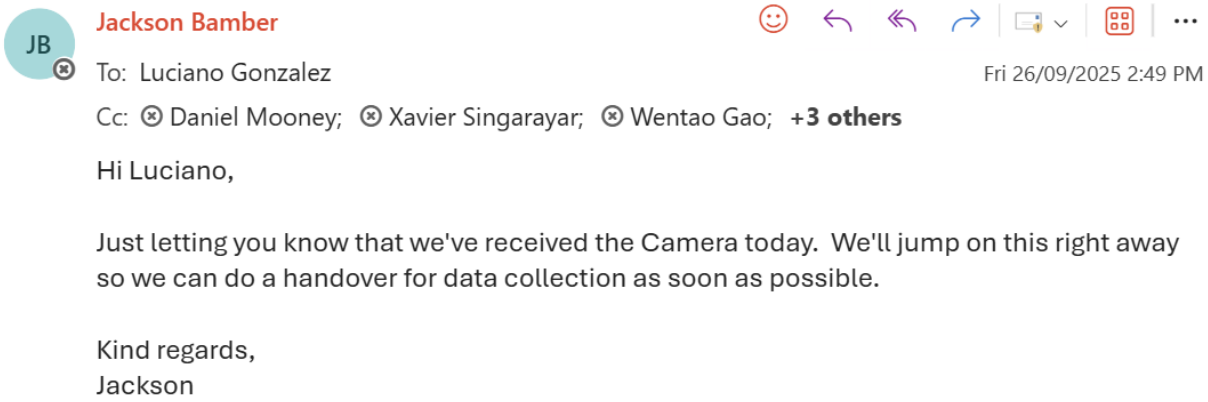
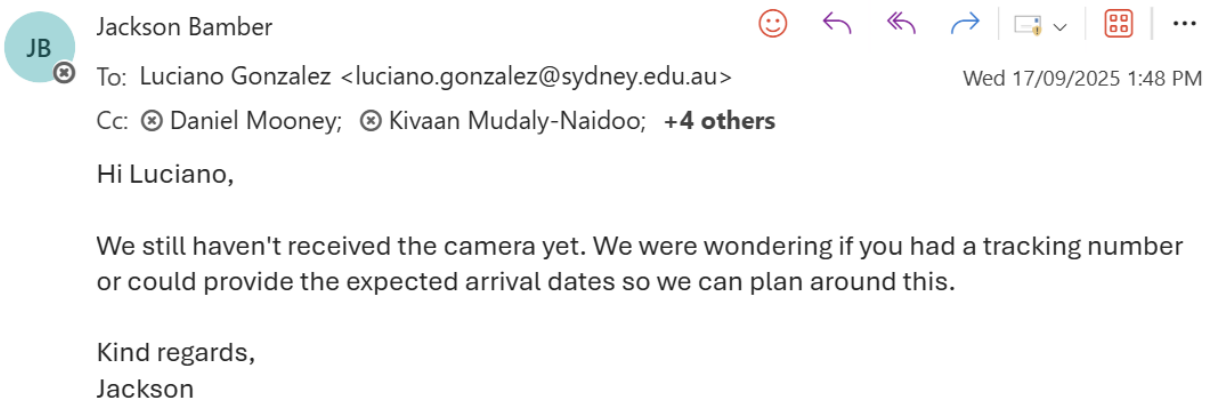


Figure 2: Communication: Confirmation of the Arrival of Camera Email 26/9



Jackson Bamber

To: Luciano Gonzalez <luciano.gonzalez@sydney.edu.au>

Wed 17/09/2025 1:48 PM

Cc: Daniel Mooney; Kivaan Mudaly-Naidoo; **+4 others**

Hi Luciano,

We still haven't received the camera yet. We were wondering if you had a tracking number or could provide the expected arrival dates so we can plan around this.

Kind regards,
Jackson

Figure 3: Communication: Camera Delivery Issue Email 13/9

Appendix C: GitHub

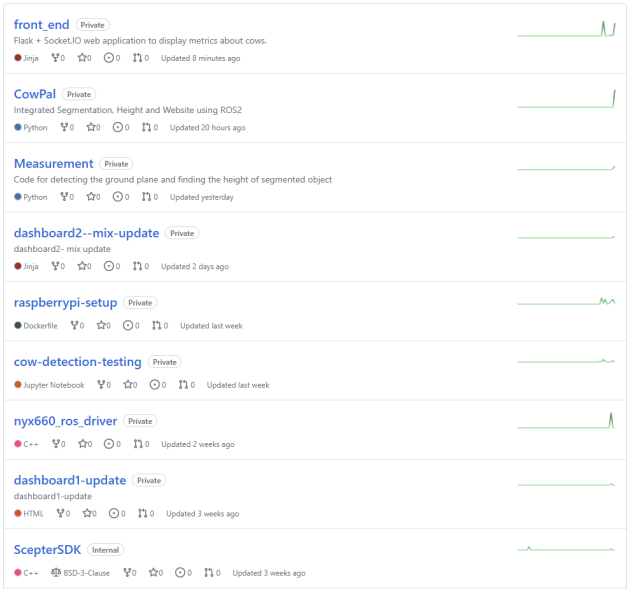


Figure 4: Repositiories Used in GitHub

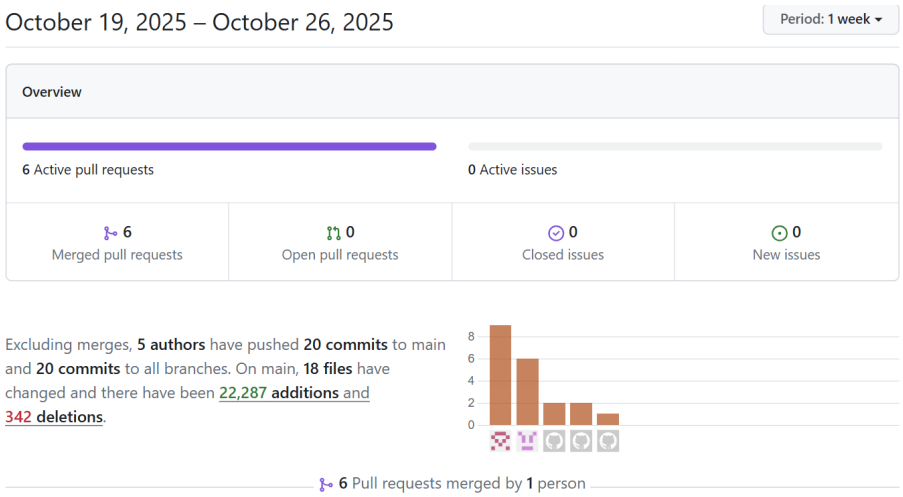


Figure 5: Version Control in Final Repository

Appendix D: Slack

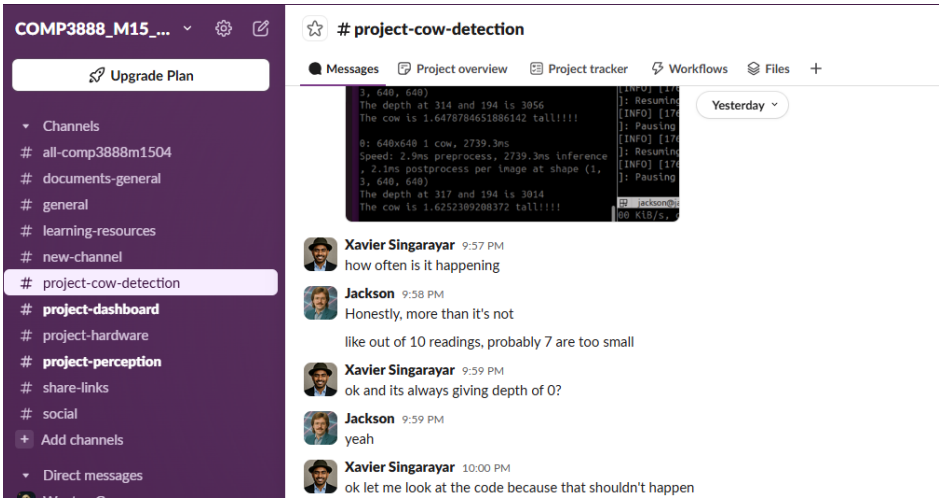


Figure 6: Slack Setup