AMPC
AUSTRALIAN MEAT PROCESSOR CORPORATION

Final Report

AI/ML Modular (and relocatable) Computer Vision and Sensing Cell

Project Code
2021-1273

Prepared by
MEQ Probe

Date Submitted
29/03/2022

Published by
AMPC

Date Published
29/03/2022

Contents

1.0 Executive Summary

1.1 Project Objectives

The purpose of this project is to develop a modular camera system that is capable of capturing multiple images simultaneously at chain speed in an abattoir. Being able to collect data in real time enables processors to make informed decisions on product consistency, and provides a strong foundation for which automated identification can be applied.

This project involved the design, selection and prototyping of the hardware for a modular camera system, including a centralised workstation and several cameras. This construction extended to an advanced design of a system capable of remaining within the boning room environment of the project partner Gundagai Meat Processors (GMP), for several months while collecting data across every day of production. Software development of a graphical user interface (GUI) occurred to ensure information could be displayed in real time to the user. The outlined objectives of the project are as follows:

- Remain in the abattoir for an extended period of time (meaning it is suitable for use in an abattoir)
- Take 1-6 photographs of primal's in lamb abattoir upon an operators command
- Save the photographs to a centralised workstation
- Display any required instructions/menu's to the user through a GUI

1.2 Hardware Selection and Design

There were three phases to the development of the hardware components of the modular camera system ultised for object detection. The project involved an initial system design and selection of hardware, followed by a prototyping phase that included several iterations of camera positioning and mounting systems to allow for optimal image collection, and a final, advanced hardware design that was installed and operational for several months continuously in an abattoir boning room.

Hardware design selection needed to consider the harsh conditions of a boning room, including the need to withstand cleaning with high pressure hot water and chemical solutions. Of greatest importance was the selection of computer workstation and cameras that were to be used. All components of the system were required to be IP67 rated, including the cameras. By selecting a camera designed for industrial use, able to capture high quality images and the ability to interchange camera lenses, it ensured that the final system would be suited to numerous applications with a processing plant. The computer workstation was selected for its powerful graphics processing unit and capability to capture data from several cameras at once. By housing all electronics inside a custom designed and built watertight stainless steel workstation with wheels, it ensured the system would be able to withstand abattoir conditions and remain flexible for data collection from multiple locations.

The hardware prototype was initially tested in a controlled environment before being tested in the boning room of GMP. Preliminary image capture occurred through single, unmounted cameras so that optimal camera placement

could be determined. Placement on the belt, including direction and angle, focal length, exposure and saturation of the image were all factors considered. Two suitable locations were determined for image collection, one a boning belt for primal detection of naked loins and naked racks, and the other a location for the collection of bagged rack images. As the system software developed, camera numbers increased from two, then four and then six operating at once. The mounting system for the cameras went through several interactions, from clamp attachment with adjustable heads to a fixed angle system with bolted anchor points. The initial design allowed for experimentation of belt positioning and camera angle, but were at risk of movement from accidental knocks and required regular recalibration. When optimal location was confirmed, attachment points were affixed to the side of the conveyor belts, and camera angles were fixed in place.

The final hardware design is modular in nature, with both the computer workstation and cameras able to be relocated between the two locations in the GMP boning room. The ability to interchange camera lenses ensured that the same camera bodies could be used across a range of positions. The only issue with the system related to the adjustable design of the camera mounting system, as it was susceptible to knocks during both production and overnight cleaning. The project moved towards a more fixed design to counteract this issue, once final camera positioning was determined. The adjustable camera head still allowed for the fine-tuning of image angle when required. The watertight design of the system withstood overnight cleaning, and at no stage in the project was there an intrusion of water into the system. The final hardware of the object detection camera system remained installed and operational in the GMP boning room with no issues and no camera recalibration for five months.

1.3 Software Development

Software development for the modular camera system required the selection of frameworks and algorithms and implementation of numerous software components, including a chain-speed data pipeline for machine learning, pre- and post-processing of images, and data recording; a GUI frontend with several distinct modes; a database to store and provide access to system-generated data; cloud synchronization utilities to enable efficient annotation of training images, heuristic error-detection systems to identify images for annotation and model training, neural network model versioning and configuration systems, simulation recording and playback utilities, and an overall software configuration system, along with various minor utilities for data processing, model format conversion etc. The completed prototype software system is able to process data from up to six cameras in real-time, allows a frontend user to monitor and override the system using a touchscreen-optimized GUI, and is able to record rich data about its operation and store images to local and cloud storage in response to manual or automated triggers. The overall software system has been shown to be reliable in operation within an abattoir environment for an extended period of time while producing accurate identifications of lamb primals at chain speed.

1.4 Project Outcomes

The final design of the modular camera system remained in GMP for five months, capable of collecting data every day of production with no complications. The system remained watertight through this period of operation, and the modular design allowed for the system to be moved easily between locations. Data was able to be captured simultaneously

across six cameras in real time, with images capable of being recorded manually by an operator or automatically by the system software. The development of a GUI was successful in displaying real time information to the user about the operation of the system, including video feeds and machine learning model outputs and allowed for user interaction through configuration options. All four objectives of the project were achieved successfully.

1.5 Recommendations

It is recommended that further research and development be undertaken to further develop these technologies which have direct and specific benefits to the red meat industry. It is of the opinion that there is immediate value in the following areas:

- Automatic Sorting: a machine learning/computer vision system could be developed to control mechanical systems such as belt diverters to enable automated sorting of meat cuts in certain situations.
- Supply Chain Traceability: with some expansion of scope, potentially including more cameras and processing nodes, the technologies used in this project could be applied to the problem of boning room traceability to track an individual carcase as it is transformed into primals and individual cuts.
- Quality Assurance: a machine learning/computer vision system could be developed to identify and count objects once boxed, and alert users to the presence of foreign objects

1.6 Conclusions

The project outcomes were successfully achieved. The final system design remained in the GMP boning room for an extended period, capable of capturing six images simultaneously and the centralised workstation was able to process and analyse these images at chain speed. The GUI displayed real time information and allowed for user interaction to configure the information displayed. The capabilities and flexibility of the system allow for it to be applied to various problems faced by processors, and will enable real-time decision making.

2.0 Introduction

In the red meat industry a consistent product is key. The problem is that there are a multitude of end-point products that can be delivered by a processor. Output from a single plant can be complex and diverse, with product specifications changing day-to-day to meet customer requirements. There is a certain level of skill required in identifying a piece of meat correctly and efficiently on the production line to ensure the product is meeting specifications, biosecurity and quality control measures. As the number of potential options for an object increases, so do the potential errors in classification. It might occur that identification tasks, such as packaging products and final checks for boxed meat, are often delegated to unskilled labour with the least experience in the red meat industry. This lack of skill and high rate of labour turnover often results in a high risk of human error occurring, disrupting work flow and having the potential to create quality and biosecurity issues for the processor. If products do not meet specifications, biosecurity requirements or are incorrectly packaged, there is the potential for reputational damage both domestically and internationally. If the wrong product is in the wrong box, the costs can be very high.

The aim of this project was to develop a modular camera system capable of detecting, identifying and counting individual objects as they pass through a predetermined point. By developing this system, different cuts of meat are able to be correctly identified and counted as they move through a point on a conveyor belt; if a miscount occurs or unidentified objects are detected, the system is capable of alerting the user to the issue as it occurs at chain speed, allowing for real time decisions to be made. This project was concerned with the design and development of system hardware, and the user experience through a graphical user interface (GUI). It aimed to develop a system that could be operational for long periods of time in an abattoir environment, have a centralised workstation and be capable of operating up to six cameras simultaneously in the capture of data. In the design, it was ensured that all electronic components were located within a watertight stainless steel box, mounted on an aluminium trolley for ease of relocation. The cameras were selected specifically for their capabilities in industrial applications and the ability to be modified to suit lighting and focal length requirements. Modular in design, the system was installed and operated at two separate locations within an abattoir boning room, collecting data across every production day for a period of five months.

This system has the capabilities and flexibility to be applied across a wide variety of use cases throughout processing plants of the red meat industry. The ability to capture images that are processed and analysed at chain speed will enable output to be used in real-time decision making. Unskilled labour requirements will be reduced, and a greater level of quality assurance and biosecurity guarantee will be achieved consistently and efficiently across the variety of end-point products manufactured by processors.

3.0 Project Objectives

The objective of this project is to develop a camera system containing 1-6 IP 67 cameras that is able to take images at chain speed in an abattoir that are of suitable quality for object detection. The following points are outcomes of this system:

- Remain in the abattoir for an extended period of time (meaning it is suitable for use in an abattoir)
- Take 1-6 photographs of primal's in lamb abattoir upon an operators command
- Save the photographs to a centralised workstation
- Display any required instructions/menu's to the user through a GUI

4.0 Methodology

4.1 Hardware Selection

The initial stage of the project required decisions on the suitability of hardware for a system that is required to operate for extended periods of time in the harsh environment of an abattoir, including being able to withstand daily cleaning with hot water and chemical solutions, while having the functionality to adapt to the requirements of specific processors environments and constraints. The first decision surrounded the selection of the major components of camera and

computer workstation to be used for the collection of data. All components of the system were required to be IP67 rated, including the cameras.

4.1.1 Camera Selection

Figure 1: Image of chosen camera, Baumer VCXG-32C.I

The selected camera needed to small enough to fit unobtrusively in the abattoir, be waterproof and knock-proof, have the capability to be adjusted to fine-tune the quality of images being collected, and the ability to be programmed in a way that allowed for multiple cameras to be operated simultaneously by independently designed software. With this criteria, the Baumer VCXG-32C.I camera was selected - it is designed to work in food manufacturing locations and are IP68 rated, deliver high quality images with the capability to stream over 30 frames per second, offered a programming interface which allows for seamless communication between devices and the ability to interchange camera lenses with varied focal lengths and capabilities and an external cover that is watertight. An example of this camera is seen in *Figure 1*.

4.1.2 Workstation Selection

The other major decision was the selection of the computer workstation. This equipment needed the capability to run machine learning algorithms, include a powerful graphics processing unit (GPU) and the capability to connect to, operate and store images collected from multiple cameras simultaneously. An Intel NUC Extreme fit these parameters and was selected, with the additional benefit of being smaller than a standard desktop computer. As this system, along with other electrical components required for operation were not waterproof, a stainless steel housing was required to contain this system when deployed in the abattoir. A casing was designed to be watertight and capable of withstanding overnight cleaning, with all external ports able to be sealed with screw-plugs and an industrial covered switched electrical socket. The workstation was mounted on a lightweight aluminium trolley to allow for easy relocation of the system to various camera installations across the boning room, as well as ease cleaning overnight (*see Figure 2*).

Figure 2: Stainless steel computer housing and trolley for easy relocation.

All cables and connectors were selected for the ability to transfer data quickly from camera to workstation, and the ability to be screw-locked in place to maintain a watertight system, and to reduce the risk of the connection being dislodged during production or cleaning.

A suitable monitor for use in the abattoir was selected, with touch screen capabilities, fully sealed and IP68 rated - similar to computer screens found in abattoirs. When the system is permanently installed, the monitor would be wall mounted to allow for easy system monitoring and manual override when necessary. However, for this project, it was decided the usage of a temporary monitor that could be taken in and connected to the system on an at-need basis was more suitable as the system is capable of remote access and does not require in-person control at all times. A washable mouse and waterproof keyboard were also acquired for in plant use when required.

4.2 Hardware Prototype

Once suitable hardware was determined, the prototype system was first tested in a controlled environment that allowed for data pathways to be developed before the system was deployed to be tested in an abattoir environment.

Initial testing utilised a single, unmounted camera to allow for flexibility in determining the optimal conditions and placement for image capture. Camera placement on the belt, including direction and angle of images captured, focal length of the camera lens, exposure and saturation were all considered during the initial phase of testing.

During this phase, suitable locations of camera installation in the abattoir were determined by finding positions that allowed for image capture of the target products clearly from multiple angles, and did not interfere with day-to-day duties of abattoir staff. Two locations were determined to trial the system, one located on a boning belt where images of naked lamb racks and naked lamb loins could be captured easily from multiple angles on either side of the belt. Another location, with similar attributes was determined for the collection of bagged lamb rack images.

At these locations, optimal camera positioning was determined through the use of handheld cameras and a repositionable mounting system. This involved a custom mounting system that had a clamp base to attach to the metal rails on the side of the belt that allowed for both the angle of the mount and the position of the camera to be readjusted as needed. The task of determining camera placement progressed alongside the development of the software system to handle two, then four, and finally six cameras operating simultaneously. During this phase of experimentation,

different focal lengths were tested to ensure all cameras were able to be calibrated to focus on a single detection zone on the boning belt. A range of lens lengths from 4mm to 16mm were trialed, with the final selection being a mix of 6mm and 8mm lenses. Two cameras which are angled to focus straight down onto the belt were initially trialled with 4 mm lenses, but the field of view was too tight to allow for optimal count calibration with the other cameras, as primals would pass through the detection zone too quickly to allow for agreement by the model. Fish-eye lenses were implemented to allow for greater peripheral capture which enabled for the model to correctly count primals.This comparison is seen in *Figure 3*. All lenses were covered by an external cover to ensure the system remained watertight and were not impacted by chemical solutions used in overnight cleaning.

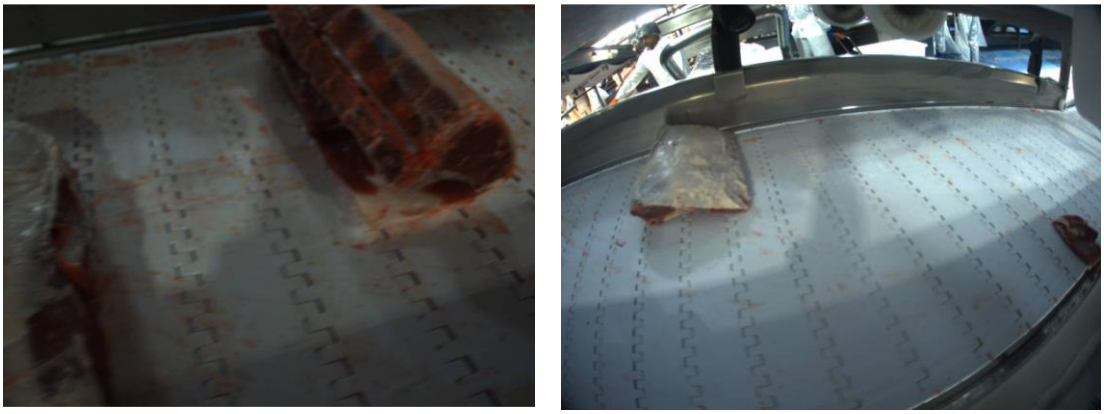

Figure 3: Comparison of images from 4mm lens (left) and fish-eye lens (right) from the same position

The camera mounts initially developed for the system were ideal for experimentation with belt positioning and camera angle, however, they were not suitable for long-term use in an abattoir environment. The clamp which held the mounting system to the side of the belt had the risk of moving when knocked accidentally during production and cleaning, causing the system to require recalibration on a weekly basis. The screws holding the mounts in place also loosened over time, which caused misalignment of the system and required readjustment. Once the system was operational with six cameras capturing data simultaneously, final placement was determined, a fixed mounting system was designed in-house. The bases for this system were manufactured in stainless steel and -welded to the side of the belt. Solid pillar mounts were then able to be screwed into place, reducing the risk of misalignment due to accidental knocks during production and cleaning. The camera was still attached to an adjustable head to allow for fine-tuning of the camera angle at the fixed location. Similar fixed attachment points were installed at the belt that images of bagged lamb racks were collected from to allow for the mounting system to be easily interchanged between locations. The two versions of the mounting system are shown in *Figure 4* below.

Figure 4: Version one camera mounting system (left); Version two camera mounting system with fixed position (right)

The cameras were positioned in a way that allowed for the capture of primals from all angles, which can be seen in *Figure 5*. Camera one and six are positioned facing down the belt, camera two and five are straight-on and camera three and four are facing up the belt. This positioning ensured that when objects were obscured from the view of one camera, it could still be detected by other cameras.

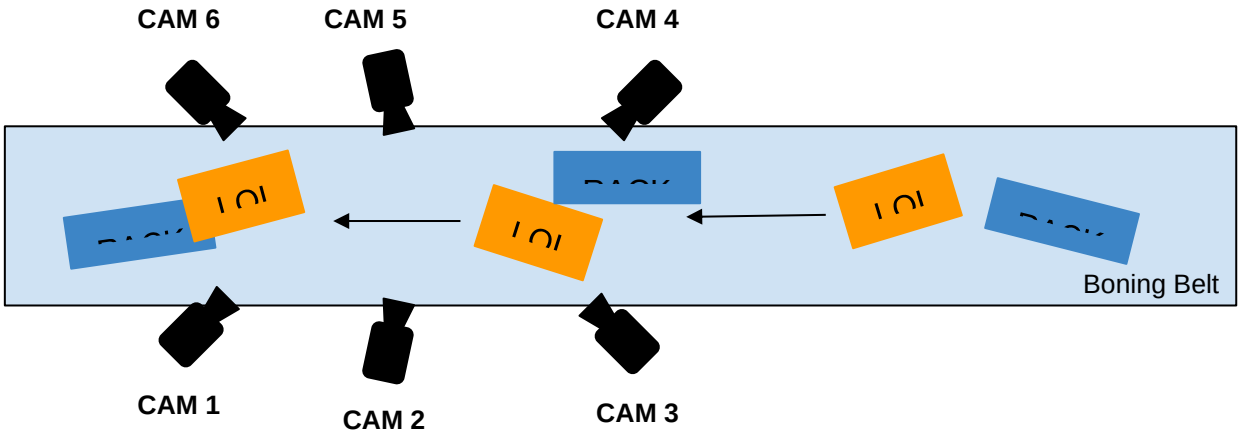

Figure 5: Positioning of the six cameras in relation to the conveyor belt and each other.

4.3 Software Prototyping

In this section, the system's software will be described, along with explanations of design decisions taken in the development of the software and the choices of software dependencies used in the implementation of the system.

4.3.1 Selection of software platform and libraries

Programming language

A primary decision in the design of any software engineering project is the choice of programming language or languages in which the software is to be written. In this case, the Python language was selected, for the following reasons:

- Python is widely used in scientific and numerical computing, particularly in machine learning contexts and in research and development projects; these uses are supported by a broad range of libraries, tools, machine learning code and supporting utilities.
- As a high-level language with expressive syntax, a flexible dynamic type system, and a comprehensive standard library, it is well-suited for projects requiring rapid prototyping and development along with ease of code modification.
- Python's support for object-oriented design enables code to be efficiently modularized and reused, enabling the software system to be highly flexible and configurable, while also offering many opportunities to create generic code that can be reused in different environments and contexts.
- There was existing strong familiarity with Python on the part of key team members, and within the organization generally.
- Python has out-of-the-box support for process-based parallelism, allowing software to be designed that uses multiple independent processes to increase performance by taking advantage of the multi-core design of modern processor hardware.
- Python's is open-source software with active development and a robust community, but its license (PSF License Agreement) is compatible with commercial use.

Numerical computation library

This project requires the handling of large array data, particularly in the form of images captured from cameras and the results of processing such images.

To carry out numerical operations on arrays, the NumPy library was used. NumPy is by far the most widely used numerical computation library for the Python language, with a vast number of higher-level image processing, machine learning and scientific computing libraries depending on its array-based functionality, and is the natural choice for a computer vision and machine learning project such as this one.

Additionally, NumPy's broad BSD-based license places no special restrictions on commercial use.

Image-processing / computer vision library

As part of the processing of image data in this project, it is necessary for the software to have access to a range of basic computer-vision operations such as contour approximation and calculation of contour statistics, determination of bounding rectangles, and drawing diagnostic and visualization data.

To this end, the OpenCV computer vision and image processing library was used, due to its fast performance, comprehensive Python bindings, broad range of supported operations, wide use in computer vision generally, active development, and existing familiarity with the library on the part of key team members. Additionally, OpenCV has a permissive Apache 2.0 license which is compatible with commercial use.

Machine learning inference library

In order to make real-time AI-based predictions using image data from the camera system, machine-learning code is required to carry out inference. For this purpose, the TensorFlow platform was used, for the following reasons:

- TensorFlow is an open source software project which is also backed by Google, which allows it to enjoy the robustness, user-responsiveness and community support associated with open source projects, while simultaneously receiving dedicated financial and development support from one of the world's largest technology companies.

- While TensorFlow supports a variety of machine-learning paradigms and algorithms, it has a strong focus on deep neural networks, a type of machine learning system which is highly suited to computer vision and image interpretation tasks such as object detection.

- A wide variety of pre-built machine learning systems and algorithms are able to be run out-of-the-box using TensorFlow, so the choice of TensorFlow as an inference library provides access to many 'off-the-shelf' solutions which can be adapted to the specific needs of the project while also leveraging the enormous data sets, cutting edge research, and extremely intensive computation that is often brought to bear in the creation of publicly available machine learning models and systems.

- TensorFlow is able to perform inference using a very broad range of machine learning algorithms and models using a unified interface, which means that software using TensorFlow for inference is able to be easily and rapidly adapted to support new requirements, environments, and data, which is important for the design of a modular AI system which is intended to support many practical applications, not all of which may be fully known at the time of initial design.

- TensorFlow has robust support for Python development, with comprehensive language-specific bindings available to Python developers.

- TensorFlow supports GPU-based computations using the CUDA platform, which allows certain processing operations to be carried out using graphics card hardware rather than on traditional processors, which greatly improves performance at inference time.

- Additionally, high-performance inference systems such as TensorRT can be used with TensorFlow, which can allow even higher performance by taking advantage of hardware-specific optimizations and pre-processing.
- TensorFlow's Apache 2.0 license does not present a barrier to commercial use.

4.3.2 Communication with cameras

To communicate with the Ethernet-connected cameras, the camera manufacturer, Baumer, provides a specialized Python library named neoAPI, which enables the software system to receive data from all cameras, and additionally offers a buffered mode in which the cameras can be directed to capture image data continuously and independently, which is streamed in the background into memory buffers and made available for the software system to use as soon as needed, rather than requiring the software system to issue image-capture commands for the cameras and then wait for the captured data to be loaded into memory.

4.3.3 Chain-speed data pipeline

To enable the system to capture image data, process images, perform machine learning inference and analysis, display information to and respond to commands from the user, and record data in real time at a speed sufficient to keep up with the pace of incoming primals at chain speed, a pipeline-based design was employed. The overall structure of the pipeline is presented in *Figure 6*.

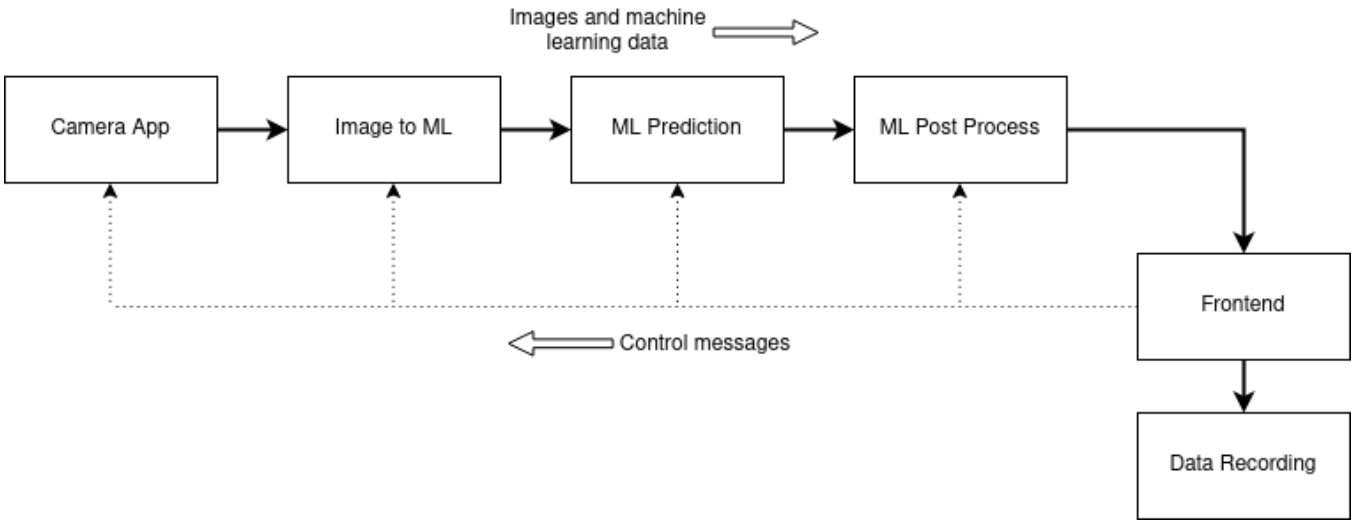

Figure 6: Diagram of software data pipeline. Each rectangle represents a separate stage of data processing carried out by an individual process; the solid arrows represent the forward flow of image and machine learning (ML) data, and the dotted arrows represent the backward flow of control messages from the user-facing frontend.

The data pipeline comprises several stages, each of which is designed to handle a single part of the overall processing of incoming data in a separate process. By dividing the processing requirements of the system into several processes, each can operate independently and simultaneously using separate processor cores, and then pass on its results to

the next stage in the pipeline for further processing, thereby taking fuller advantage of the processing capabilities of the hardware to increase overall performance.

Each process sends data forward to the next stage through a dedicated inter-process pipe, while control messages are communicated as needed from the frontend to upstream stages through an inter-process queue. The individual stages and their responsibilities can be summarized as follows:

Camera App

The responsibility of the Camera App stage is to connect with the camera hardware, configure camera settings, receive image data from the cameras, and pass that data to the Image to ML process.

Image to ML

The responsibility of the Image to ML stage is to pre-process images to be submitted to the core machine learning model by resizing, cropping, and converting to the appropriate numerical format for direct input to the neural network inference system.

ML Prediction

The responsibility of the ML Prediction stage is to receive the pre-processed image data and use it to perform inference on individual images using a machine learning model, to obtain predicted bounding boxes and classes of detected objects within each image.

It can be configured to use either standard CPU-based computation, higher-performance GPU-based computation using CUDA, or GPU-based computation using TensorRT which utilizes pre-optimized hardware-specific inference engines to achieve even higher performance at the expense of run-time flexibility.

ML Post Process

The responsibility of the ML Post Process stage is to process the results of machine learning inference in order to contextualize and enrich the data. Several types of processing are done in this step, depending on the system's current configuration:

- Tracking objects through time between individual frames for each individual camera.
- Matching objects between cameras at each specific moment in time.
- Heuristically detecting possible prediction errors and other events of interest and marking the relevant images for recording to longer-term storage so that they can be corrected or analyzed by human experts, so that the system or model can be improved.
- Calculating the system's best overall prediction for the classification of objects using the machine learning model's direct single-image predictions combined with derived temporal tracking and cross-camera matching predictions.

- Counting objects that pass through the view of the camera system using the system's best overall prediction of their classification.

Frontend

The responsibility of the frontend is to:

- Display video feeds from the cameras to the user, optionally annotated with various information summarizing the outputs of the machine learning model and derived predictions from the previous steps.
- Receive commands from the user which are communicated to the other processes in the pipeline in order to dynamically update certain configuration options.
- Alert the user to potentially-erroneous detections or classifications by marking them on the video feeds.
- Allow the user to manually mark images for longer-term storage so that they can later be corrected or analyzed to improve the system or model.

Data Recording

The responsibility of the data recording process is to receive data from the upstream steps in the pipeline, insert relevant information into the database, and additionally save images to disk which have been marked for recording either manually by the user at the Frontend or automatically by the system's heuristics at the ML Post Process stage.

4.3.4 Database storage of results and metadata

The system is able to store various prediction results, configuration states, image metadata and other information in a database. This allows data relevant to the system's behavior, performance and results to be comprehensively cross-referenced. This cross-referenced data is then available both for direct access by human experts, and for automatic processing and analysis by software utilities.

Database system

The database software used by the camera system is the MySQL relational database management system, and was chosen for the following reasons:

- The relational design of SQL-based database systems such as MySQL allows various types of data to be recorded in specialized tables and cross-referenced to provide a rich and comprehensive record of the system's operation.
- The database can be separated from the system's software, and then connected to by the software either locally within the system workstation, or across the network to a remote server, depending on the needs of

the specific application, and according to site-specific concerns such as bandwidth and Internet connection reliability.

- The database can safely be remotely accessed, queried, and, if necessary, updated by authorized users even during active system operation, due to its client-server design and support for atomic insertion.
- The database can easily be synchronized or backed up to a remote server or cloud service if required.
- MySQL is widely used, and is queried with the standard SQL language, making it compatible with a very broad variety of software and services, and able to be used easily by any experienced system administrator.
- MySQL is GPL-licensed open-source software which can be freely used in proprietary commercial applications and connected to by proprietary commercial software as long as no MySQL source code or modified version thereof is distributed without the GPL license (which this project does not entail).

Database schema design

The design of the database schema is intended to capture information of various types about system configuration, operation and results, and is organized into the following tables:

- *CameraDevice:* Records information about each physical camera, including its serial number and MAC address.
- *Layout:* Records information regarding changes in the physical camera setup.
- *Model:* Records configuration and architecture information about each individual machine learning model version used by the system.
- *Run:* Records information about each individual run session of the system, including static configuration data, model version used, and corresponding *Layout* table entry.
- *CameraSoftwareConfig:* Records information about the software configuration of each camera used in each run of the software, including gain, saturation, exposure time and other settings, and relates this information to the *Run* and *CameraDevice* tables.
- *Frame:* Records information about each frame captured by each camera during system operation, including its time of capture, image metadata such as resolution, and other relevant data. Each frame refers to the *CameraSoftwareConfig* table, allowing the frame data to be cross-referenced to the relevant run, layout, model version, camera hardware and camera software configuration information.
- *ObjectPrediction:* Records information about each object detected by the machine learning model, including classification, confidence, and bounding box coordinates. Refers to the *Frame* table, allowing each individual detection to be cross-referenced to all the information described above.

- *Image:* Records information about images recorded by the system for longer-term storage, the local file path to each image, and the remote cloud storage path to the image if it has been uploaded. Refers to the *Frame* table, allowing images to be matched with their corresponding frame data.

- *ExclusionBoundary:* Records information about the exclusion boundaries used by the system, which are configurable regions within each camera frame that designate the relevant area for machine learning analysis (for example, the part of the frame which contains a clear view of the belt on which the primals appear).

- *PredictionZone:* Records information about the prediction zones used by the system, which are configurable regions within each camera frame which designate the region in which the system will attempt to make a final classification for primals entering the region using all available information.

- *Simulation:* Records information about simulations recorded using the system software, which can be played back using the system software in a special simulation mode designed for evaluation, analysis and debugging purposes.

4.3.5 Cloud synchronization of images for model development

In addition to local storage on the system workstation of images marked for recording either manually or by the system's automatic heuristics, the software includes a subsystem which is able to sort and organize the recorded images for each day of operation and upload them to Azure cloud data storage. This provides the following advantages:

- While the data storage space on the system workstation is limited, cloud storage capacity is effectively unlimited, allowing a very large archive of images to be stored and retrieved across many months of operation.
- Software such as the Computer Vision Annotation Tool can be linked to the cloud storage container and accessed by expert annotators to produce labels for further machine learning model development.
- Storage of valuable model development data on Azure services provides a level of robustness, security and redundancy that makes data loss very unlikely.

4.3.6 User interface (UI)

Selection of GUI (Graphical User Interface) library

While there are myriad options available for displaying video feeds to and receiving input from the user in a Python project, including image-display and keyboard-control functionality built into image processing libraries such as OpenCV which are already dependencies of the project code, the decision was taken to use Qt, a dedicated general GUI processing library, to drive the system's GUI (through its PyQt5 Python bindings), as it offers the following advantages:

- *Separation of GUI event thread from data processing thread(s)*: Qt offers functionality to straightforwardly separate processing of GUI events from general UI-related data processing threads, and a comprehensive signalling system to allow the separate threads to communicate asynchronously, thereby allowing the UI

process to receive video data and format it for display without causing the GUI to become unresponsive to user input at any point.

- *Support for complex user interactions and rich UI elements*: While an image-processing library like OpenCV is able to display information to the user, receive mouse clicks, and monitor for keypresses, a general-purpose GUI system such as Qt can additionally be used to implement more complex user interactions involving interface elements such as buttons, sliders, labels etc. Particularly as the system is intended to support touch-screen interaction, the flexibility and control over UI design afforded by a dedicated GUI library allows for a much more user-friendly and intuitive experience which need not rely on peripherals such as keyboards which are typically unsuitable for use in an abattoir environment.

GUI design and implementation

The implementation of the GUI incorporates two threads: the main window thread, which draws and updates UI elements and handles user interactions, and the frame processing thread, which receives images and relevant data for each camera from the upstream pipeline components, annotates the received frames with information derived from the outputs of upstream processing (including alerts to the user when the model has low confidence in a prediction), and combines frames from multiple cameras in a grid layout for simultaneous display.

The frame processing thread sends each annotated and combined frame to the main window thread for display, and passes frame data to downstream pipeline components which handle data recording.

The user is able to use the controls provided by the UI to direct the main window thread to send signals to the frame processing thread in order to adjust which cameras are displayed, to instruct the frame processing thread to mark a frame for recording to disk by the downstream pipeline components, or to send control messages to upstream processing stages.

Because the UI event processing and frame processing take place in separate threads, the UI continues to be responsive to user input while frame data is being processed. *Figure 7* illustrates the primary components of the GUI implementation.

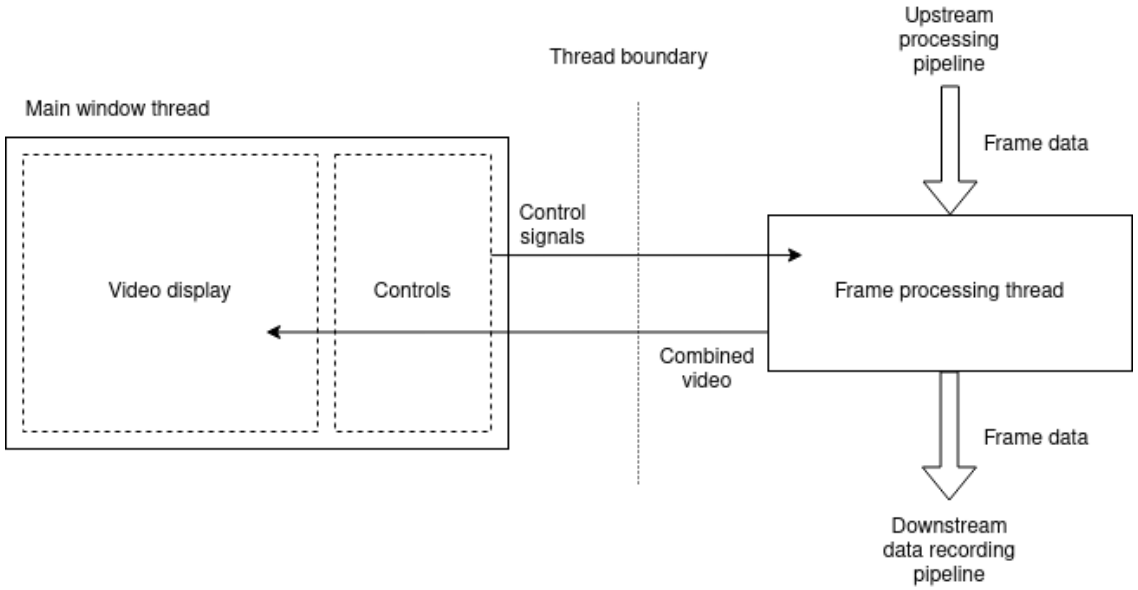

Figure 7: Diagram showing the primary components of the system's GUI frontend.

The design of the GUI allows for various specialized implementations to use the same camera selection and video processing source code, while presenting additional information or controls to the user depending on intended application.

Figures 8 and 9 show the system's frontend GUI in two different modes: the mode depicted in *Figure 8* is intended to be used by a human expert to monitor system performance, mark errors for recording and later analysis, and fine-tune model parameters. On the other hand, the mode depicted in *Figure 9* is intended to display to an end-user the objects currently detected by the system alongside the totals of each class of object that the system has counted in the current session, and allow the user to manually adjust the counts if the system makes an error.

In both modes, each individual camera feed can be selected or deselected, and the size and layout of the camera display is dynamically adjusted according to the number of cameras being viewed. Additionally, both modes are designed to be touchscreen-friendly, requiring no keyboard or mouse input from the user.

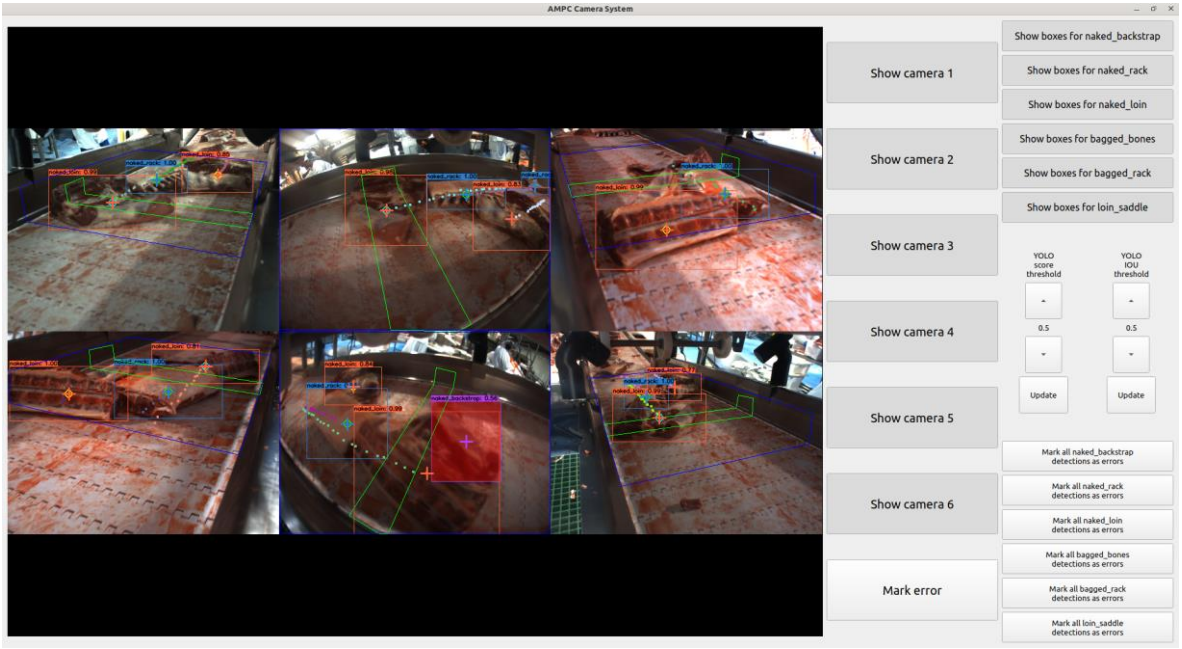

Figure 8: Screen capture of GUI frontend running in a mode designed for manual error recording. Note that one object's bounding box is highlighted in red, indicating that the corresponding detection was made with low confidence and may be in error.

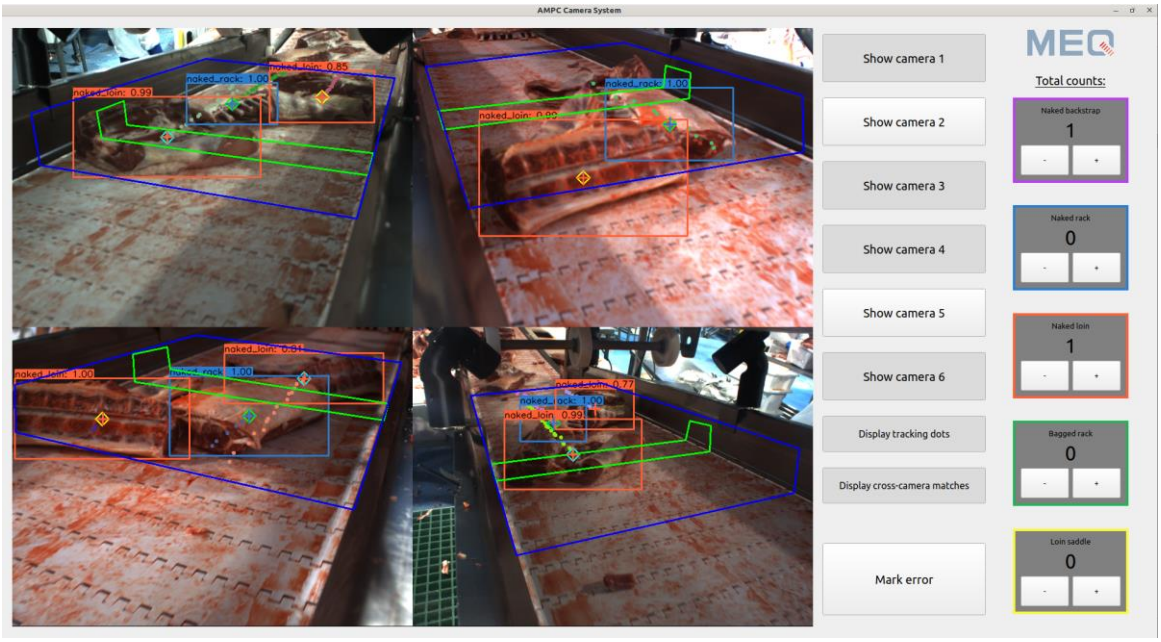

Figure 9: Screen capture of GUI frontend running in a mode designed to display the total counts of each class of

object that has passed by the system during the current session and allow the user to override the count.

Headless operation

In addition to the GUI frontend, the software is also designed to support 'headless' operation, in which the workstation does not have a monitor attached, and the software is automatically started when the system is powered on. This mode is most suitable for system operation and data acquisition without a user present, while still allowing the software to be remotely controlled and maintained using the remote administration methods detailed in the section below.

4.3.7 System configuration

Due to the complexity of the system, its research and development focus, and its modular and relocatable nature, a significant number of settings and parameters are made available for configuration. Some of these are statically configurable and are loaded when the software is initialized for a new session, while some are additionally dynamically configurable, and can be altered by the user during operation. Furthermore, the machine learning models used by the system are separately configured, allowing the model in use to be updated or changed without modification of the general system configuration.

Static software configuration

Static configuration is primarily controlled through a central YAML configuration file, in which key-value pairs are specified in order to switch various system elements on or off, provide numeric and categorical parameters to algorithms used within the system, supply information necessary to connect to the cameras depending on the current hardware configuration, and control the software in other ways. These settings are recorded in the system's database for each session, and default values are used for many configuration settings if not supplied.

An alternative configuration file can be specified at invocation time, allowing multiple separate sets of configuration settings to be maintained for different uses. For example, separate configuration files can be created for separate locations in which the modular system may be deployed, allowing the software to be easily reconfigured for different environments and applications when the hardware is relocated.

Additionally, some important configuration settings, such as the choice of UI mode, activation of simulation mode, or debugging output verbosity can be overridden as part of the invocation of the software using command-line parameters, so that several common usages of the software can be achieved without modification of the YAML configuration file.

Dynamic software configuration

In addition to the static configuration which is set for each run of the software, some configuration settings are also able to be modified dynamically while the system continues to operate. This allows the user to change, for example, model confidence thresholds, error detection parameters, or areas of interest within the camera frames in real-time, improving the convenience of system calibration during relocation or camera setup, and allowing the efficiency of data gathering for model development and improvement to be augmented in the presence of a human operator by enabling them to adjust data-gathering parameters according to the immediate situation.

Model configuration

Some of the configuration required by the software is directly related to whichever machine learning model is in use. In particular, model parameters such as input image array sizes, recognized classes and their index numbers, and various model-related thresholds may vary between models for different applications, and even between different versions of a model for a particular application as it is developed. For this reason, these parameters are handled separately: machine learning model files developed for use by the system are equipped with metadata containing the parameters necessary to run that model.

While each of the parameters can be overridden if needed using the system's main configuration file, in normal operation only the name or version number of the model needs to be specified, and then the relevant parameters are read from the corresponding metadata and stored in the system's database so that the software can automatically configure itself to optimally support the selected model.

As such, the machine learning model used by the system can be changed by simply specifying a different model version number or name, without requiring the manual reconfiguration of model parameters, thus enabling a more efficient and convenient process of iteration and model development, or redeployment of the system in different locations or to detect different types of objects.

4.3.8 Remote administration

Because the system is designed to operate for long periods of time in an abattoir environment without the regular presence of an in-person operator, it is important that the software can be remotely monitored, maintained and updated, especially during the process of prototyping and development, when rapid iteration and feedback can greatly increase the rate of progress.

To this end, the workstation is connected to MEQ's systems using ZeroTier, a secure virtual network utility, which allows authorized developers and administrators to access the system remotely using the Secure Shell Protocol, allowing both the software and machine learning models to be updated, maintained and configured using standard tools such as Git. It also allows authorized users to access the system's database remotely for analysis, debugging, or development purposes.

Furthermore, for GUI maintenance and development purposes, the system can also be accessed by authorized users through utilities such as x11vnc or RemotePC, which allow the remote user to interact with the system's graphical interface. In the case where no monitor is connected to the system workstation, the Xvfb virtual framebuffer utility is utilized to allow the GUI to nevertheless be accessed by remote users if necessary.

4.3.9 Simulation recording and playback

For purposes of software development, debugging, analysis and improvement, it is very useful to be able to reproduce the same conditions under different configurations, code changes, or versions of dependencies. Because the system is designed to process live camera data in an abattoir setting, it is not feasible in most cases to physically reproduce situations for software development purposes. Additionally, because the system operates in real-time, in some

circumstances the normal speed of operation may be too fast for developers to properly perform analysis or evaluation of the system's behavior.

To solve this problem, the software incorporates a simulation recording and playback system: intervals of input image data can be recorded to the system workstation's file system by the user in relevant situations, and then these files can be stored or transferred to other computers, and used with the software in a special simulation mode that replaces the subsystem that normally receives data from the cameras with a replacement subsystem that reads the recorded simulation data as though it were coming from the cameras.

In this way, the same situation can be 'played back' through the software pipeline in order to analyze the system's behavior, debug software code, and test improvements and modifications. Additionally, this system allows the simulation data to be fed into the system at lower than real-time speed, or even 'frame-by-frame' under direct user control, enabling deeper analysis of the running system than is possible under real-time operation.

4.4 Advanced Hardware Design

Using the knowledge and experience gained during the prototyping phase, an advanced hardware design was developed, taking in the considerations of modular design that allows for straightforward installation and relocation in an abattoir environment, suitability for overnight cleaning, and long-term reliability of the system.

4.4.1 Modular Components

The aim of the modular design was to ensure that the system could be adapted to numerous areas, and to ensure that installation and relocation was simple and straightforward. For this project, locations were pre-determined to allow for the capture of primals from multiple angles, while not interfering with day-to-day operations of staff. When final camera placement was determined, fixed anchor points for the camera mounting components were able to be installed ahead of time. As the cameras used across locations were the same model, and lenses were interchangeable, in-house manufactured mounts were able to be moved between locations. Adjustable camera heads allowed for the angles captured to be fine-tuned and fixed into place, and in the scenario where the system was installed for a long period of time at the boning belt, in a location were knocks to the cameras was inevitable, solid pillar mounts were designed as the final version of the mounting system. The main computer workstation was mounted on a trolley, to allow for ease of relocation in the boning room, with power accessible at both locations where the system was deployed. All cabling used was the same design, and a selection of different lengths allowed for the system to be connected back to the central workstation with no restrictions on reach. For a permanent installation, cable lengths can be easily customised to length. While for this project, a monitor was not permanently set-up, and it was decided to use a temporary monitor on an as-needed basis and accessed the system remotely most of the time, it is capable of being installed with a screen suitable for abattoir use which can be mounted near the system, or in a more central location for ease of monitoring. The components of the final design can be seen in *Figure 10*.

Figure 10: Components of the advanced hardware design. Clockwise from top left: Cameras mounted in place, computer system housing, camera on fixed angle mount, screen for real-time tracking.

Across the seven months the system has been installed in the abattoir, the minor issues that have arisen mostly come from the adjustable design of the initial camera mounting system, which is the reasoning behind why the project moved towards a more fixed design as the position of the cameras became more constant. The location of cameras on the primal boning belt was more exposed to knocks from staff during production and therefore required a more solid design to ensure cameras did not require constant re-calibration of the detection zone, while at the bagging belt collection point, cameras were able to be fitted with the adjustable mounts as there is less staff traffic during production, and the mounts of this design can withstand overnight cleaning. In the five months that the system was operational at the boning belt location using the custom designed fixed mounting system, no major issues have occurred.

4.4.2 Suitability for Overnight Cleaning

To ensure all the components were able to withstand overnight cleaning with hot water and chemical solutions, a watertight system was developed. All components are rated to IP67 level to protect against the ingress of water from

high-pressure sprays, and where pieces of the system joined together (ie. cables connecting to cameras and workstation), all connections were screw-locked to minimise the risk of water entering the system. An industrial grade electrical socket and cabling was also used. All electrical components of the main computer workstation are housed in a stainless steel box, which when sealed was watertight and had no leaks throughout the project. At no stage in the project did an issue arise due to the intrusion of water in the system. The modular system is therefore suitable for overnight cleaning in an abattoir environment.

4.4.3 Long-term Reliability

By designing a modular system that has a in-house designed camera mounting system, the project was able to be developed to ensure long-term reliability. By designing a system that is water-tight, and developed to ensure camera alignment was maintained through production hazards and high-pressure cleaning, the system has been able to operate with no issues and no need for recalibration for image collection in a five month period. The system has remained in place in an abattoir environment since the design was finalised, and has captured image data from every production day with no issues.

5.0 Project Outcomes

The outlined objectives of this project were achieved for this project of developing a AL/ML Modular (and relocatable) Computer Vision and Sensing Cell.

5.1 Remain in the abattoir for an extended period of time (meaning it is suitable for use in an abattoir)

The final design of the modular camera system and central workstation remained permanently in the abattoir for five months once the final camera mounting system was installed. The system has operated with no complications and has been capable of collecting images every day of production during this period. The system has remained watertight throughout operation, and during periods of system relocation no complications have arisen. The modular design allows for ease of moving and installation of components, with a simple recalibration of the detection zone the only requirement after a relocation.

5.2 Take 1-6 photographs primal’s in lamb abattoir upon an operators command

The modular camera system is able to capture video data in real-time from up to six cameras, and the software data pipeline includes a specialized component process for recording still images to data storage. The GUI frontend for the software is able to be used with a robust touch screen designed for industrial use, and includes functionality to manually or automatically mark images for recording.

5.3 Save the photographs to a centralised workstation

Images recorded either manually by an operator or automatically by the camera system software are, by default, saved to the local filesystem on the centralised workstation which operates the cameras. The system also incorporates synchronization software which is able to automatically upload captured images to secure cloud storage.

5.4 Display any required instructions/menu's to the user through a GUI

The camera system software includes a robust touchscreen GUI which is able to display real-time information to the user about the operation of the system, including video feeds and machine learning model outputs, and also includes various controls with which the user can interact with the system to capture images, configure certain options, or control the displayed information.

6.0 Discussion

There are no results to discuss for this project.

7.0 Conclusions / Recommendations

The outcomes of this project were successfully achieved. The system was able to remain in an abattoir for an extended period of time, and was capable of collecting data across every day of plant operation. The final design of the system was able to collect six images simultaneously as lamb primals passed a predetermined location on the boning belt, and the centralised workstation was able to process and analyse these images at chain speed. Interaction with the system is able to be performed by users through a GUI that displays real-time information and allows for the control of what information is displayed.

By being able to capture, process and analyse images at chain speed, this system will enable real-time decision making by processors in the red meat industry. The capabilities and flexibilities of the system allow for a broad range of applications to enable abattoirs to solve problems distinct to them. Implementing this system will contribute to a greater level of quality assurance and biosecurity measures, and reduce labour requirements in ensuring a consistent end point product is delivered.

8.0 Bibliography

Not applicable.

9.0 Appendices

Not applicable.