# Real-time Quality Control for Offsite LGS Frame Manufacturing using Vision-based Deep Learning

by

George Nader

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

University of Alberta

# Abstract

Efficiency and accuracy are crucial in offsite panelized construction, including in the production of light-gauge steel frames. Maintaining consistent screw fastening is an ongoing challenge in the construction industry, and defective screw fastening can lead to quality problems in panelized construction projects. To address this issue, this research introduces a light-gauge steel framing machine that integrates a computer-vision system for precise quality control. This framework is developed and tested on real panelized construction projects. This innovative approach equips the light-gauge steel framing machine with visual perception capabilities, enabling real-time image capture and analysis of the framing process. By employing advanced imaging technology and YOLOv8n machine learning architecture, the computer-vision system provides immediate feedback to the machine's control system. This process facilitates precise decision-making regarding screw- fastening operations. The implementation of YOLOv8n on the light-gauge steel framing machine uses Python and the OpenCV library to process visual data in real -time, determining optimal methods to mitigate defects. The experimental results demonstrate that the computer-vision system substantially improves the integrity and precision of light-gauge steel frames. The results indicate a reduction in defects and rework, as well as significant enhancements to operational efficiency and material utilization. These enhancements underscore the promising potential of integrating real-time computer vision and artificial intelligence with the manufacturing processes of panelized construction, establishing a foundation for future innovations in autonomous systems within this industrial sector.

# Preface

This thesis is an original work by George Nader. No part of this thesis has been previously published.

## Acknowledgements

I am immensely grateful to a host of remarkable individuals who supported me throughout my journey to obtain my master's degree. Foremost, I extend my deepest appreciation to my supervisor, Dr. Mohamed Al-Hussein, and my co-supervisor, Dr. Ahmed Bouferguene for their patient mentorship, constant support, and continuous encouragement that were pivotal to my research.

I am also thankful to my colleague, Djamel Eddine Touil, for his significant contributions to our collaborative research, and his readiness to share his deep knowledge.

Lastly, I would like to express my profound gratitude to my family. My wife has been a pillar of love, support, and companionship. Similarly, I am thankful to my parents and brother for their unwavering encouragement.

Thank you all for being part of this rewarding journey.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| Abbreviation | Description |
|---|---|
| ADR | Automated defect recognition |
| AI | Artificial intelligence |
| AVI | Automated visual inspection |
| BIM | Building information modelling |
| CAM | Computer-aided manufacturing |
| CPS | Cyber-physical systems |
| CPPS | Cyber-physical production systems |
| CVS | Computer vision system |
| DBMS | Database management system |
| DDDM | Data-driven decision-making |
| DL | Deep learning |
| LGS | Light-gauge Steel |
| LGSFM | Light-gauge steel framing machine |
| LHS | Left-hand side |
| ML | Machine learning |
| OEE | Overall equipment effectiveness |
| OpenCV | Open computer vision |
| PdM | Predictive maintenance |
| PLC | Programmable logic controller |
| RHS | Right-hand side |
| RTQC | Real-time quality control |
| WFM | Wood framing machine |

# Chapter 1 . Introduction

## 1.1 Background and motivation

The demand for efficiency in offsite panelized construction has driven the adoption of manufacturing processes, particularly the manufacturing of light-gauge steel (LGS) frames. However, despite the continuous advancements in LGS-based panelized construction, challenges persist in consistently and reliably fastening screws. These inconsistencies often lead to quality issues, compromising the integrity of the prefabricated panels.

The light-gauge steel framing machine (LGSFM) showcased in Figure 1-1 is a unique prototype designed specifically for fabricating LGS frames within a factory setting. It represents an innovative approach to prefabricated panels, marking a significant advancement in manufacturing technology.

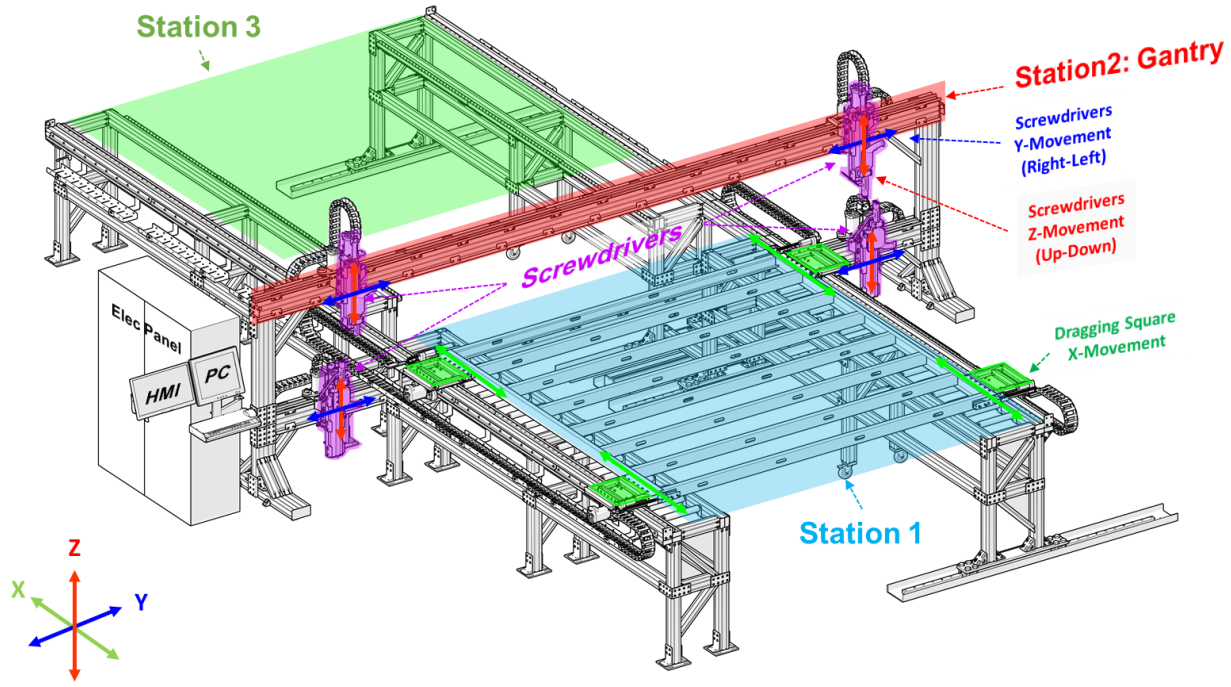



Figure 1-1. Overview of the LGSFM

The LGSFM comprises three main stations. Station 1 is where the operator oversees material feeding, including tracks, studs, and self-tapping screws that are loaded manually into the screwdrivers. Station 2 houses a gantry equipped with four SENCO DS242-AC screwdrivers

mounted on actuators, which are guided by stepper motors to facilitate the *z*-axis movement of the screwdrivers. Stepper motors also enable the movement of the screwdrivers along the *y*-axis of Station 2 (gantry). Station 3 is dedicated to the production of framed panels. Once the panels are completed, they are transferred to the next station of the prefabrication process.

The dragging squares contain electromagnets, and electromagnets and are responsible for capturing and dragging the panels along the *x*-axis. This allows the panels to move between Station 1 and Station 3 throughout the fabrication process. Moreover, the control system is centered around the Programmable Logic Controller (PLC), MODICON–M251 (Schneider Electric), which orchestrates the actions of stepper motors and other components like electromagnets and screwdrivers. Additionally, a Human–Machine Interface (HMI) offers user-friendly visualization and control over the LGSFM.

Figure 1-2 illustrates the framing steps for the current LGSFM. The machine's set-up phase involves three key steps. First, a recipe file, known as an RCP file, must be uploaded into the machine's PLC. This file is generated by based on the 3D model of the panel. It plays a vital role in determining the precise movements of the machine's dragging squares, ensuring the correct studs' positions according to the panel's design. The second step focuses on activating the PLC's code. This code, which integrates the instructions from the RCP file along with other essential functions such as timing and various motor controls, governs the machine's operations. Finally, the setup process includes using the "Vijeo" designer software by Schneider Electric to manage the HMI. This interface allows the operator to control and monitor the machine throughout the production process, providing direct interaction with the machine.

The next critical step is the machine calibration. This is done by tapping the "Calibration" button on the HMI. This command triggers the LGSFM to adjust its components (e.g., dragging squares, screwdrivers) to their precise positions. This step also confirms that all sensors and switches are functioning properly.

Once the setup and calibration are complete, the machine is ready to initiate fabrication of LGS panels. The operator begins with the material preparation procedures, such as organizing studs and tracks and ensuring the screwdrivers magazines are loaded with screws.

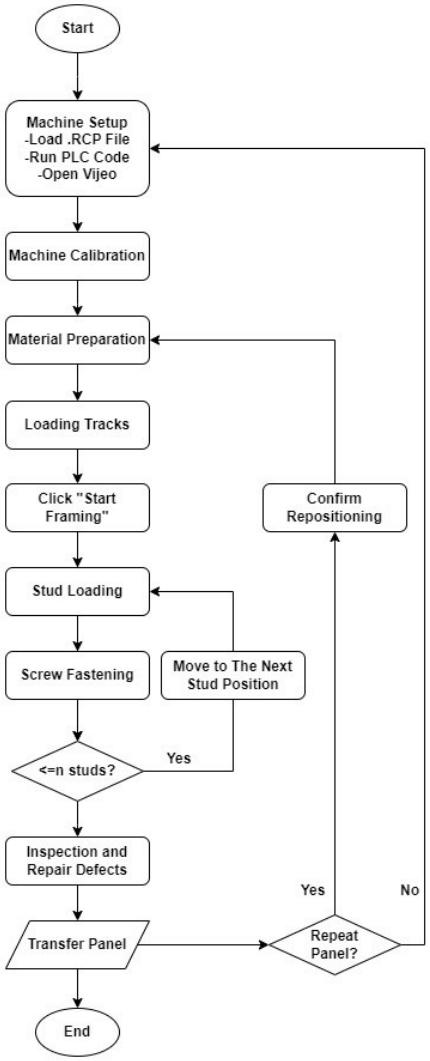



Figure 1-2. LGS framing process using the prototype LGSFM

Once the calibration is complete and the materials are prepared, the operator begins placing tracks on both sides of the machine. The tracks must be stacked onto the dragging squares, which hold them firmly in place as they move toward the insertion site on the stud. Once the tracks are placed, the operator moves on to the framing process. By pressing the “Start Framing” button on the HMI, the dragging squares slide along the *x*-axis, guiding the tracks to the exact location of the first stud. Once the stud is positioned, the operator presses the “Load” button on the machine’s HMI. This signals the screwdrivers located at each corner to begin the fastening process using the self-tapping screws. Once the screws are securely in place and the fastening is complete, the dragging squares automatically start shifting the panel to the next position for another stud to be added. This cycle

continues until all studs are securely fastened. Once the panel is fully assembled, it progresses to the next stage for more fabrication processes. To ensure the quality of the fabrication, a visual inspection is carried out. This step focuses on identifying and fixing any issues, including any screws that might not be properly fastened.

For subsequent panels of the same design, the process is repeatable with a simple press of the "Start Framing" button. If a new panel design is needed, then the process must restart with a new RCP file, illustrating the machine's adaptability to various panel configurations.

Despite the machine's adaptability, there is still a major issue in the framing process: the occurrence of defects during the fastening process. Specifically, the machine may fail to fully secure the screws and may require rework to rectify these issues. To address this challenge, this thesis introduces an artificial intelligence (AI)-based framework that integrates a computer-vision system (CVS) into the control system of the prototype LGSFM. This innovative approach equips the machine with visual capabilities, enabling real-time observation and analysis of the framing process. The CVS provides feedback to the control system, facilitating precise decision-making regarding screw-fastening operations.

This system enables the capture and processing of image data using Python codes in conjunction with the Open Computer Vision (OpenCV) library. The CVS assesses the real-time condition using a deep-learning model based on YOLOv8n architecture to detect the fastening of screws during the framing process. This system supports optimal screw-fastening positions, mitigating the occurrence of defects and enhancing the overall quality of LGS frames.

In summary, the integration of a CVS represents a transformative step in automated LGS framing for panelized construction. This approach not only addresses existing quality challenges, but also sets the stage for future advancements in autonomous panelized manufacturing systems. The research described herein explores the design, implementation, and performance of integrating CVS with the LGSFM's current control system. It provides insight into the practical applications of the integration by achieving a real-time quality control (RTQC) system based on machine learning (ML) techniques. The research described in this thesis investigates how minimizing defects and waiting time, reducing resource waste, and limiting rework can benefit the panelized construction industry, resulting in increased productivity of the manufacturing process.

## 1.2 Research objectives

There are three central objectives to this study:

1) Enhancing the quality of the prefabricated panels by implementing CVS, allowing real-time detection and correction of defects in screw fastening and ensuring consistent and reliable assembly of LGS framing components. This is vital for panelized construction projects.
2) Improving productivity for both machine and the operators by reducing interruptions due to rework resulting from defective screw fastening.
3) Enabling data-driven decision-making (DDDM) through the analysis of the generated data and measuring the impact of the CVS framework on the machine's performance.

Collectively, these objectives highlight the potential of integrating CVS technology into the automated LGSFM, enhancing quality and efficiency in panelized construction manufacturing.

## 1.3 Methodology

In this study, a structured, four-step methodology is adopted underpinned by three objectives, as illustrated in Figure 1-3. The first objective is to establish RTQC through the integration of CVS and ML. The second objective is to enhance the productivity of LGS frame manufacturing by minimizing defects, reducing rework, and decreasing the cycle time for panel production. The third objective is to develop a DDDM framework for the production of LGS frames.

Toward these objectives, this study examines quality control principles and relevant technologies. This involves a review of the literature on offsite panelized construction practice and on the utilization of CVS and ML technologies in the manufacture of prefabricated LGS frames.

Once the foundational knowledge is established, the first objective is addressed by developing an AI-driven CVS model for the process of the screw-fastening RTQC system within the LGSFM's existing control system. This AI-based quality control framework, capable of detecting process defects in real time, would mitigate defects during production. The successful implementation of this system can be expected to result in reductions in manufacturing defects, rework, and waiting time for both the machine and the operator. This fulfills the second objective, resulting in increased productivity of the manufacturing process.

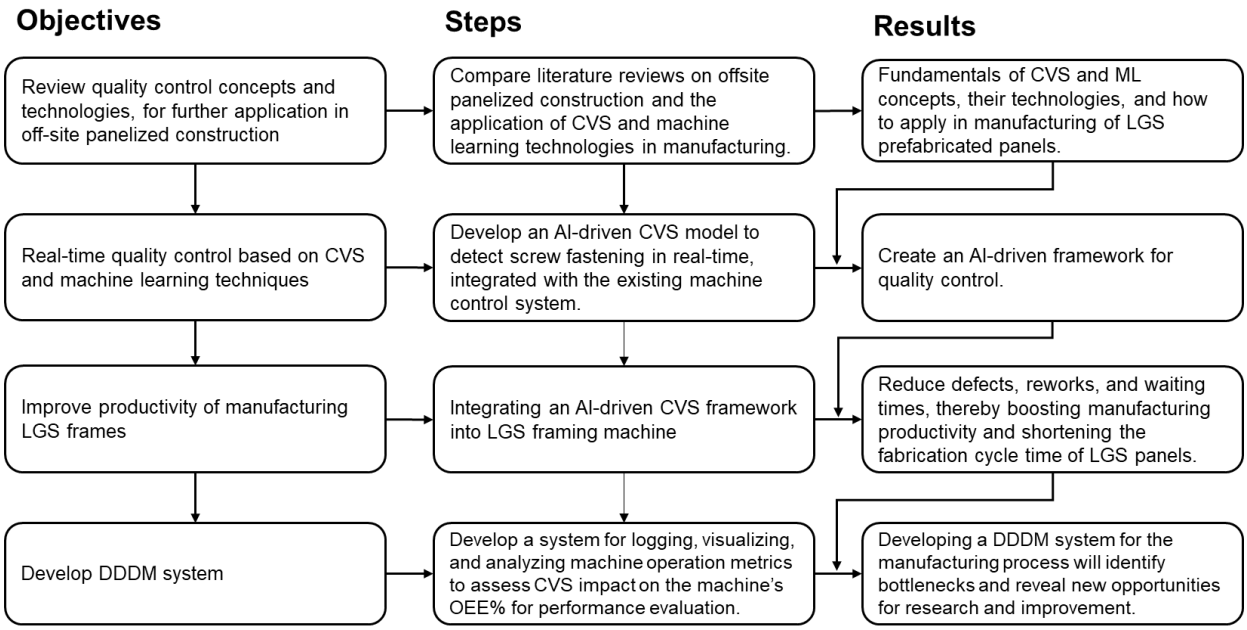



Figure 1-3. Overview of methodology

To facilitate data management and analysis, all relevant data is logged and documented within a structured system encompassing the machine's PLC, Python coding, MySQL as a database management system (DBMS), and Grafana for data visualization and analysis. This framework tracks critical metrics such as machine operating times, cycle times, waiting times, and defects, providing a valuable insight into machine performance via Overall Equipment Effectiveness (OEE) metrics. These metrics serve as the cornerstone of a DDDM framework for identifying process bottlenecks and exploring opportunities for future research and improvements of the LGS framing process.

## 1.4 Thesis organization

This thesis is organized into five chapters. Below is a brief overview of each chapter:

Chapter 1: Introduction

This chapter provides background information on the research, detailing the prototype LGSFM stations and parts, its operational method, and the step-by-step process of LGS framing. It identifies the current challenge involving the LGSFM, represented in the occurrence of defected screws during the manufacturing process. Chapter 1 also briefly outlines a potential solution to

this challenge through the application of CVS. Moreover, the chapter defines the research objectives and describes the methodology employed to fulfill these objectives.

Chapter 2: Literature Review

This chapter investigates the significant advantages of panelized construction. It focuses on LGS framing and compares its benefits to traditional materials such as wood or concrete. It explores current technologies in panelized construction, focusing on quality control challenges in manufacturing. It also highlights the potential of automated visual inspection (AVI) systems based on CVS to enhance quality and improve the framing process of the existing LGSFM, supported by AVI's proven success in various manufacturing sectors.

Chapter 3: CVS Framework and Implementation

This chapter outlines the CVS framework designed for this study, detailing the ML process from image data collection to training the YOLOv8n architecture. It also covers the integration of the CVS framework with the LGSFM's existing control system, followed by the experimental implementation and initial testing within LGS frame manufacturing. This chapter aims to evaluate initial improvements in performance and defect reduction, fulfilling the study's first objective.

Chapter 4: Impact of CVS Framework on the Machine's Performance

This chapter details the setup of the machine metrics collection through a PLC-Python-MySQL-Grafana framework. It includes a detailed overview of creating the machine database using MySQL as a database management system (DBMS), and a Grafana dashboard as a visualizing tool. Together, these form the basis of the LGSFM's DDDM system. The chapter explains how machine metrics are visualized through the Grafana dashboard. It also discusses the analysis of these metrics using OEE metrics to demonstrate improvements in the LGSFM's productivity and production cycle times after the integration of the CVS, thereby fulfilling the remaining objectives of this study.

Chapter 5: Conclusion

This chapter articulates the academic contributions and contributions to industry practice of this study, highlighting how the findings can be applied in real-world settings to improve the efficiency

and quality of LGS framing processes. Additionally, it discusses the limitations encountered during the research and proposes avenues for future work to expand on the work presented herein.

# Chapter 2 . Literature Review

## 2.1 LGS framing in panelized construction

The concept of offsite panelized construction is characterized by the prefabrication of structural components in a controlled factory environment prior to their delivery to the construction site for assembly. This method is detailed in the work of Gunawardena & Mendis (2022). They explored the design and construction facets of prefabricated building systems, highlighting the efficiency and sustainability of this approach. Their work showed the significant benefits of offsite panelized construction over traditional construction methods, including high-quality, customizable, and environmentally friendly building solutions, reduced production time (De Vincenzo et al., 2018), lower labour costs, and minimized waste (Darwish et al., 2020). Li et al. (2015) demonstrated that panelized construction not only speeds up the building schedule but also substantially reduces construction waste when compared to traditional stick-built methods. Central to this innovation are light-gauge steel (LGS) frames which have offered a robust and lightweight alternative to traditional wood framing. Reflecting on the economical landscape and environmental concerns that influence lumber availability and cost, LGS framing steps in as a practical substitute (Bateman, 1997).

Current modular constructions of high-rise buildings rely on either steel or concrete modules. Steel modules weigh 20%–35% less than their concrete counterparts (Liew et al., 2019), resulting in faster installation with bolted connections. However, concrete modules pose constraints due to heaviness, resulting in the need for more modules and connections (Thai et al., 2020). The significance of LGS lies in its versatility in various applications, such as single-family housing, terraced housing, and smaller apartment buildings. This system is also used for separating walls and infill walls within multi-storey buildings framed with steel or concrete. The inclusion of features such as fire resistance, lightweight construction, mobility, and suitability for buildings ranging from 2 to 6 storeys (Burstrand, 1998) have significantly contributed to the increasing prominence of light steel framing in panelized construction (Malik et al., 2019). It is also important to note that LGS offers the potential for recycling and reuse (Martins et al., 2013). The ability to recycle and reuse makes it an economical and sustainable alternative to traditional materials such as wood and concrete (Reza et al., 2022).

## 2.2 Core technologies in offsite panelized construction

Offsite panelized construction is considered a highly efficient method that harnesses advanced technologies throughout its manufacturing phase. Meanwhile, BIM revolutionized planning and optimization in construction, facilitating detailed planning and visualization of construction projects before the physical construction begins. In this manner, BIM allows for detailed 3D modelling and simulation. This enables designers and engineers to optimize the use of materials and predict potential issues early in the design phase using a software platform such as Autodesk Revit (Liu et al., 2017). Wei (2023) highlights advancements in BIM technology with a Revit add-on for automating the design of ribbed precast concrete panels, which simplifies the creation of detailed models and shop drawings. This enhancement significantly reduces manual drafting time and boosts the accuracy and efficiency of construction processes. Similarly, Abushwereb et al., (2019) developed another add-on for Autodesk Revit, designed to automate the design and drafting of wood-framed structures, significantly streamlining the production of detailed 3D models and shop drawings. The add-on reduced manual effort, ensured compliance with building codes, and enhanced the efficiency and accuracy of prefabricated panel production.

Moreover, the adoption of 3D printing and innovative material technologies has enabled the manufacturing of complex components and sustainable materials. One example is timber-glass composite panels, which enhance energy efficiency (Pequeno et al., 2009; Sanjayan & Nematollahi, 2019; Tay et al., 2017), increase construction speed, lower labour costs, and reduce waste (Lu et al., 2016).

Project management software is a technological approach to managing the complex logistics of modular construction. In their paper, Dos Santos et al., (2023) presented an online platform designed to enhance project management in sustainable construction. This platform supports the integration of design and legal frameworks and facilitates the management of orders and workflows.

On the other hand, many machines were developed through the adoption of Computer-aided Manufacturing (CAM), which uses computer software to automate manufacturing processes, relying heavily on high-precision numerical control machines to ensure accuracy and efficiency in producing panelized components. For example, two such machines were invented at the University of Alberta. The first machine is a wood framing machine (WFM) that automates the production of

wood frame panels, following a BIM 3D model derived from Revit of a wood frame panel. The other machine that has been developed is a prototype light-gauge steel framing machine (LGSFM), incorporating similar technology to that of WFM to manufacture LGS frames.

## 2.3 AVI systems and applications

Despite the advancements and the technologies used in offsite panelized construction, ensuring consistent quality remains a significant challenge, particularly due to issues with standardization (Lin et al., 2022). To enhance quality assurance in offsite panelized construction, practical and efficient methods like vision-based inspection techniques can be employed (Bae & Han, 2021). Because prefabricated panel manufacturing is particularly vulnerable to errors, technological solutions that tackle precision in offsite construction are of critical importance. Therefore, automated quality control has become crucial in manufacturing, enhancing efficiency and accuracy by providing a precise, real-time analysis of the manufacturing process (Lyu & Chen, 2009).

The integration of artificial intelligence (AI) and deep learning (DL) technologies have transformed the field by using convolutional neural networks (CNNs) to analyze visual data. This enables faster and more precise quality assessments exceeding the traditional human inspections in both speed and accuracy in many industrial applications, particularly with advancements in computer vision system (CVS) (Rahimi et al., 2021). While automated visual inspection (AVI) systems using DL algorithms have proven highly effective in detecting defects and maintaining high quality standards in various manufacturing settings, CVS have made notable advancements in off-site construction. According to Alsakka et al. (2023) CVS shows great potential in this area. Their review identifies key studies since 2018, such as progress monitoring, quality assurance, ergonomic analysis, process guidance, and safety management. These studies show that computer vision can effectively support construction projects by improving monitoring and productivity, saving time and effort, ensuring high-quality production, and ensuring worker safety.

## 2.4 Advantages of AVI and Challenges

Numerous studies have shown that implementing CVS can decrease inspection costs by reducing labour requirements and creating faster processing times. For example, Schulenburg (2018) demonstrated that automated defect recognition (ADR) system designed to operate 24/7 without human intervention, can significantly cut costs when compared to traditional inspection methods.

Additionally, a paper by Cinar et al. (2015) quantified the savings from reducing defective products reaching customers by identifying and removing defects early in the process. This prevented downstream costs, which in turn minimized returns and increased customer satisfaction, further adding to the financial benefits. Collectively, these studies highlight the considerable return on investment provided by automated CVS technologies, justifying the initial costs through substantial savings in operational expenditures. In addition to being accurate and cost-effective, another advantage of the integrated use of automated inspection systems with computer vision technology is data utilization. Particularly data-driven methods, which utilize machine learning (ML) techniques, and which are common in modern predictive maintenance (PdM) (Zhang et al., 2019). These methods are also being increasingly applied in real-time PdM systems, where historical failure data is unavailable (Bahar et al., 2023). ML techniques are commonly used in these systems, with a focus on failure forecast, defect detection, and predicting remaining useful life (Unal et al., 2021).

Despite their advantages, implementing automated inspection systems with computer vision technologies in manufacturing faces several challenges. These systems must ensure high precision and consistency under different operational conditions like variable lighting, dust exposure, and mechanical vibrations (Chung & Kim, 2006; Xi et al., 2017). Integrating modern technologies with older systems adds complexity, requiring a hybrid approach that merges hardware design and software development. This ensures the system stays compact, robust, and reliable, which is important in the industrial sector (Sayahi & Ismail, 2022). Finally, deploying these advanced technologies requires substantial data for effective model training. However, continuous advancements and improvements in these technologies are expanding the possibilities in automated quality control, leading to more intelligent and dependable manufacturing processes (Hütten et al., 2024).

As showcased previously, the application of AVI systems in various manufacturing sectors clearly demonstrates their effectiveness in enhancing precision and reducing costs. These systems not only streamline quality control but also reduce the need for manual oversight, thus, improving efficiency. This evidence strongly supports the adoption of AVI in offsite construction, where its implementation would lead to significant improvements in production accuracy and cost efficiency.

## 2.5 LGSFM – Case studies, recent innovations, and opportunities for improvements

Indeed, most LGS panel fabrication processes traditionally rely on manual labour, with limited automation or technological integration. This reliance on human-based methods can lead to inconsistencies in quality, production delays, and increased labour costs. In contrast, recent research has made significant strides in automating the fabrication of LGS panels. One such study was conducted by Malik et al., (2019), who introduced a prototype machine for fastening LGS framed wall-panels. The machine transferred manufacturing information from building information models to a Programmable Logic Controller (PLC). This system generates a collision-free tool, enhancing safety and efficiency and representing a ground-breaking departure from conventional practices. Thus, the integration of cyber-physical systems (CPS) in the fabrication of LGS panels not only underscores its uniqueness but also positions it as a transformative solution for modernizing the construction sector (Martinez et al., 2022).

However, a significant challenge surrounding the machine lies in ensuring compliance with building codes and regulations. Moreover, maintaining consistent quality control throughout the fabrication process, particularly in large-scale production, can be difficult. In light of this, ensuring early error detection and maintaining the quality of prefabricated panels during the manufacturing process has become a paramount concern. To address quality and defect mitigation challenges, several notable research initiatives have been launched. One such study by Martinez et al., (2019) proposed the implementation of a vision-based system aimed at monitoring the quality of steel frames during the pre-fabrication or assembly stages, prior to the fastening of screws. The system's objective is to verify that the soft assembled panel accurately matches its BIM representation. Building upon this foundation, another study done by Martinez et al., (2020) delved deeper into quality control by introducing an intelligent vision-based system. This system not only assesses the squareness of fabricated panels, but also detects defects in the fastened screws during the screw-fastening process—a critical step in the fabrication phase. Furthermore, other research focused on optimizing the intersection areas within LGS panels, which requires precise manufacturing and screw-fastening based on the frame's BIM model specifications (An et al., 2020). In parallel, Martinez et al., (2022) proposed a comprehensive framework integrating the Zero-Defect Manufacturing (ZDM) principle with a cyber-physical production system (CPPS).

This framework introduces a dual CPPS architecture, enabling both machine and product inspections, subsequent data analysis, and the implementation of ZDM strategies. By integrating findings from the research studies done by An et al., 2020 and Martinez et al., (2020, 2019), this framework enables continuous improvement cycles in the LGS manufacturing process. Supplying ongoing data on process quality to the framework optimization its operation, ensuring higher efficiency and product quality.

Despite these significant research efforts to enhance the quality of LGS frames, challenges persist in mitigating defects in the LGSFM. The key to resolving these issues is understanding the root causes of defects, particularly in the screw-fastening process. In the current LGSFM prototype, the fastening process is based on the calibration of screwdrivers to a precise position called "approach position" and is considered critical in screw-fastening operations. This position represents the ideal position for screw fastening. However, consistently achieving the ideal screw position is problematic due to several factors:

1) Alignment and levelling issues: Misalignment between Station 1 and Station 3, or uneven tables, can cause the required approach position to shift during the fastening process, resulting in a different position from the one set during calibration. This results in defects during manufacturing.
2) Material imperfections: The use of flexible or imperfect materials can deviate the screwdriver from the approach position, resulting in defects.
3) Mechanical tolerances: During the fastening process, actuators pushing the screwdrivers toward the tracks may cause movement along the *z*-axis. This opposite movement can create a gap, altering the approach position and lowering the fastening quality.

These factors result in variability in the approach position, making a fixed approach position inadequate for consistent quality. The solution lies in a real-time quality control (RTQC) system. This system not only detects defects, but also monitors and adjusts the approach position dynamically, ensuring optimal screw placement throughout the manufacturing process. Thus, real-time feedback and adjustments are crucial for achieving ideal results in LGS frame fabrication. Figure 2-1 illustrates the occurrences of defects during the fastening process.

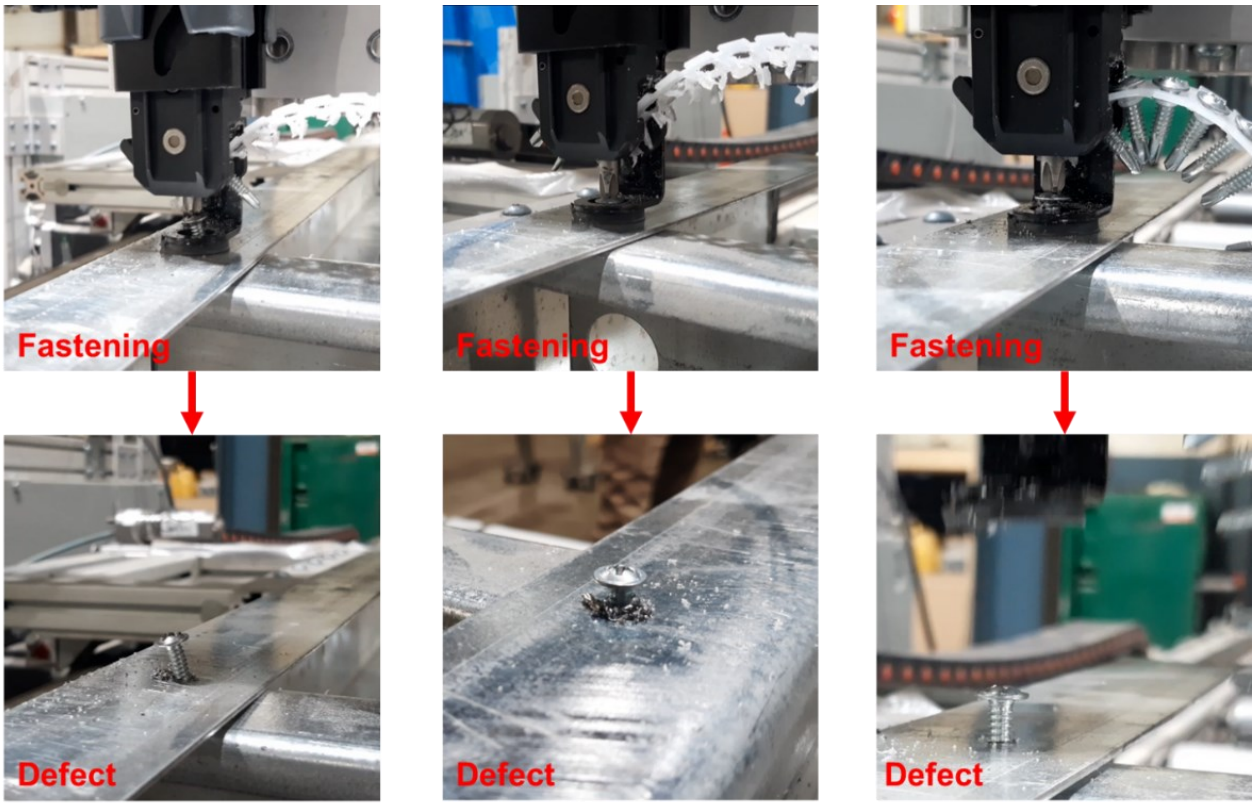



Figure 2-1. Defects occurrences during fastening process

To achieve RTQC, this study introduces an AVI system enhanced by an AI-based CVS. This system integrates AI via a pre-trained model within a Python environment, bringing AI capabilities into the control system of the manufacturing process.

This integration of CVS with the current control system of the LGSFM enables the machine to autonomously monitor and adjust itself (this is expanded upon later in this thesis). Integration significantly enhances accuracy and efficiency by replacing manual inspections with a real-time CVS system. This marks a critical advancement in RTQC. Additionally, the CVS not only speeds up the production line, increasing throughput and decreasing lead times, but also cuts labour costs by reducing the need for continuous human oversight and rework. Furthermore, the CVS forms part of the data collection framework employed in this study. The data collection framework, in turn, enables ongoing analysis and improvement of LGS framing practices, leading to smarter and more efficient production of prefabricated LGS wall panels.

# Chapter 3 . CVS Framework and Implementation

## 3.1 Framework description

The framework, illustrated in Figure 3-1, comprises four main components.

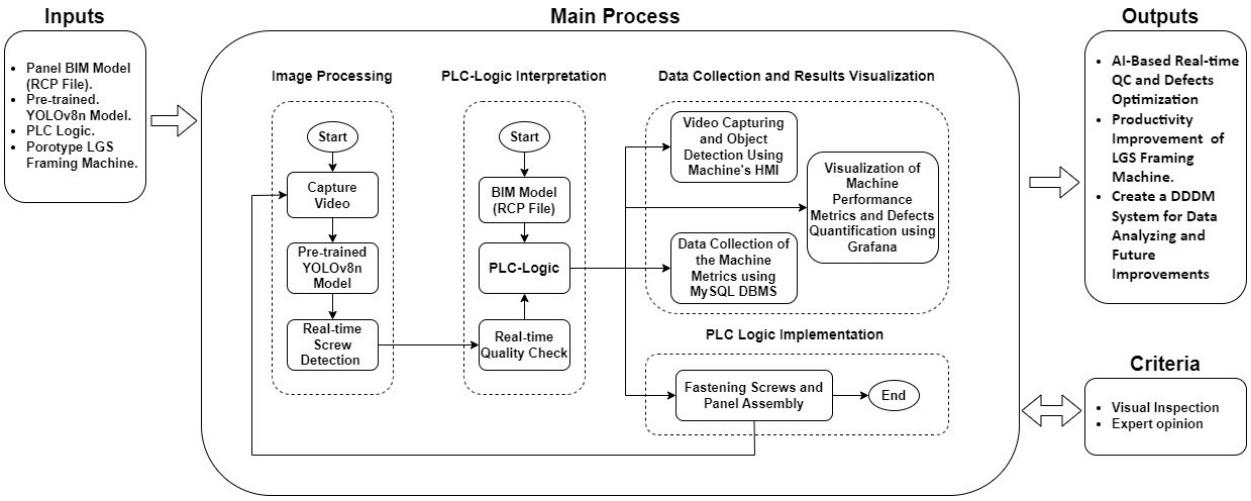



Figure 3-1. CVS framework overview

First, the inputs consist of essential elements such as the panel BIM model (an RCP file for the panel's 3D model), a pre-trained YOLOv8n model for object detection, and the Programmable Logic Controller (PLC) governing the light-gauge steel framing machine (LGSFM) and the manufacturing process. Second, the main process involves four key steps: image processing for identification of components and defects, interpretation of PLC logic to ensure compliance with predefined rules, real-time visualization of the fastening process and object detection through a human–machine interface (HMI) for monitoring and control, real-time visualization of machine performance metrics using Grafana as an online-based visualization tool, and real-time data collection of the machine metrics using MySQL as a database management system (DBMS). The fourth step is the implementation of the logic for screw-fastening and panel assembly. Third, the criteria for evaluation include a visual inspection of the manufacturing process and the final product, and insights from experts in the field. And finally, the outcome of this automated system aims to provide continuous improvement by enhancing productivity and quality control through AI-driven optimizations, establishing a data-driven decision-making (DDDM) framework for real-time monitoring and data analysis.

Initially, the PLC is pre-programmed with logic to control the movement of motors, actuators, and the screw-fastening process. However, the specific timing for initiating, dragging, and transitioning to screw-fastening is determined by the RCP file associated with the panel's 3D model. This RCP file, generated via a specialized RCP file generator, is uploaded to the PLC. By leveraging this file, the PLC can identify the precise locations of each stud, dictating the movement of the dragging squares and the timing for screw-fastening operations. Simultaneously, two cameras are linked to the machine's computer, tasked with capturing video using the Open Computer Vision (OpenCV) library through a custom trained YOLOv8n model to detect the screws during the screw-fastening operation. The cameras are operated within the PyCharm environment, executing Python codes for real-time detection and quality control. As an enhancement to the original PLC code, a "Quality-Check" step was added. This step allows the PLC to obtain distance measurements from the Python code in real time. The measurements specify the height of the screws and the necessary displacement of the screwdriver to fasten the screws correctly in their ideal position. This step facilitates adjustments if the screws fail to reach their intended locations during initial attempts. Upon successful placement of the screws, the "Quality-Check" process concludes the screwdriver's operation, returning them to their home position for continued framing. Subsequently, cameras confirm screw positions post-operation and label them accordingly. If screws are in their ideal position, they are labelled as "ideal". If they are not in their ideal position, they are labelled as "defect". This final step serves as a pivotal component of the proposed process's quality control, preceding subsequent screw-fastening operations.

Throughout these processes, the captured video, along with the status of screws, whether deemed "ideal" or "defect", provide a comprehensive overview of all screwing operations, facilitating real-time monitoring and quality assessment.

## 3.2 Machine learning process

This section focuses on supervised machine learning (ML), a type of ML where algorithms are trained using labelled examples. These examples consist of inputs paired with their corresponding desired outputs (Nasteski, 2017). This section explores scenarios where images are labelled as "keep_screwing", "defect", or "ideal".

In supervised ML, the algorithm learns from these labelled examples to make predictions on new, unlabelled data. By training on known inputs and outputs, the algorithm can generalize its learning

to classify or predict outputs for new inputs (Janiesch et al., 2021). During training, the algorithm receives input-output pairs and adjusts its model based on the differences between its predictions and the true outputs. This process continues iteratively until the model's performance reaches an acceptable level. Supervised learning can be applied to a wide variation of application, such as detection, prediction, and generation in scenarios where historical data can inform predictions about future events (Sharma et al., 2021). For example, historical image data can be used to train a model to predict the content of new images, enabling the model to perform tasks such as image classification and recognition.

As shown in Figure 3-2, the initial phase of supervised ML involves acquiring the data, especially when dealing with image data. This can entail obtaining images from various sources, such as capturing image data, where the images of the screws are collected for detection tasks. Such is the case for this study.

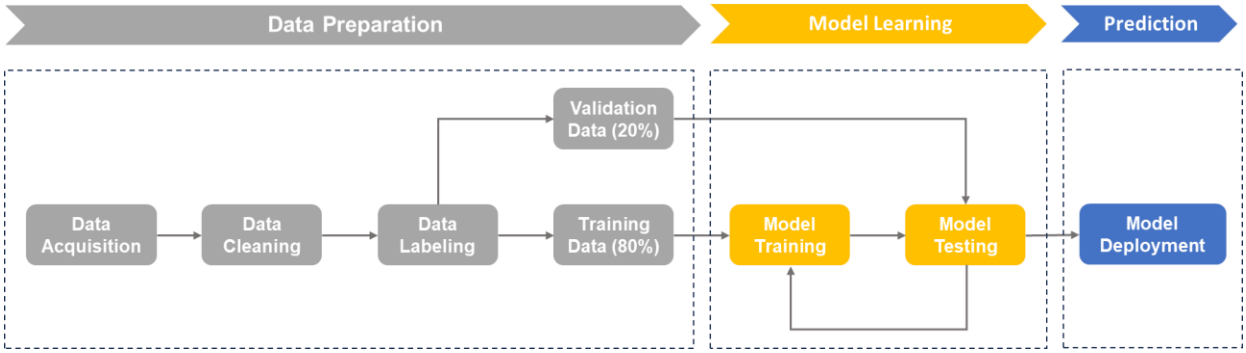



Figure 3-2. Supervised ML process

After acquiring the raw image dataset, the next essential step is data cleaning. This involves removing any unclear or low-quality images from the dataset, as these can adversely affect the learning process. Some images, such as those depicted in Figure 3-3, may not be suitable for training due to their poor quality, which could significantly increase the likelihood of false positives (Jiang et al., 2020).

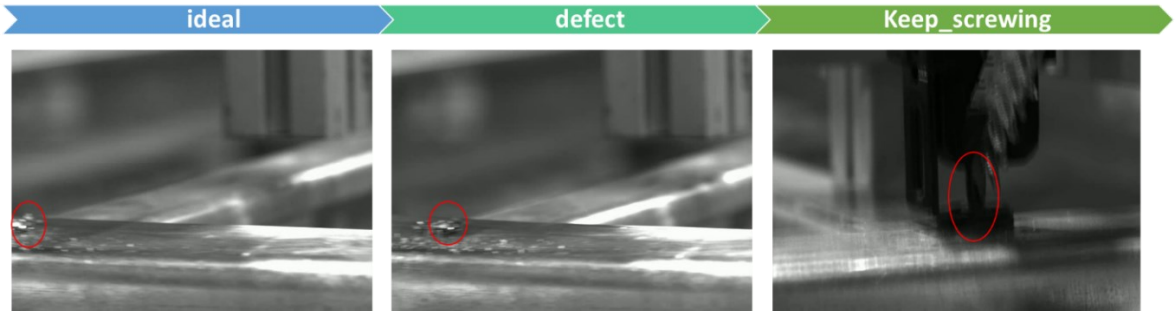



Figure 3-3. Examples of unclear image data

After data cleaning, a phase known as image labelling annotation) begins. This stage focuses on two fundamental aspects: first, the type of label must be decided. For this project, the label type is object detection. Second, the required number of label categories must be determined. Three distinct categories have been identified for this study: “keep_screwing”, “defect”, and “ideal" screws. Next, each image in the dataset is processed through the labelling phase using the labelling tool, Computer Vision Annotation Tool (Computer vision annotation tool, 2024), as discussed in more detail later in this thesis. This process ensures that the model learns to accurately recognize and differentiate between the specified categories in real-world scenarios.

After labelling the images in the dataset, it is crucial to divide the dataset into training and test sets. A typical distribution involves using approximately 80% of the labelled images for training, while the remaining 20% are reserved for validation (Ameli et al., 2024; Chabi Adjobo et al., 2023). This split ensures that the model is trained on a large dataset while also providing a separate subset for performance evaluation.

The training process begins by passing this data into an object detection algorithm. The YOLOv8n architecture was selected for this study. The training phase involves two main steps: training and validation. During the training phase, the model learns to detect and classify objects based on the labelled images by adjusting its parameters to minimize errors in predictions as it continuously improves through exposure to various scenarios within the training dataset. Following the training, the model’s accuracy and performance are assessed using the 20% of the dataset reserved for validation. This evaluation helps identify how well the model can generalize to new, unseen data, which is crucial for ensuring its reliability and effectiveness in real-world applications.

The evaluation metrics tested are as follows:

1) Precision: The proportion of true positive predictions among all positive predictions made.
2) Recall: The proportion of true positive predictions among all actual positives.
3) Average Precision (AP): The average of the precision values at different recall levels.
4) Mean Average Precision(mAP): Mean of average precision across classes (Wu & Dong, 2023).

Following evaluation, the model is ready to be deployed for predictions on new data outside the original dataset. This deployment phase enables the model to make predictions in real-world scenarios, such as classifying new images.

### 3.2.1 Data preparation

The data collection process of this study aimed to train two deep learning (DL) models for object detection, focusing on upper screwdrivers positioned on both the left-hand side (LHS) and the right-hand side (RHS) of the machine. To accurately capture the intricacies of screw positioning, two Basler aca640-90um cameras were used (Dierks, 2004). At full resolution, the cameras have a max frame rate of 100 frames per second, an image size of 640×480 pixels (width = 640 pixels, height = 480 pixels), and a pixel size of 9 µm. The cameras are supplied via two Universal Serial Bus (USB) 3.0 ports on the machine's computer. These two cameras were strategically fixed to provide comprehensive coverage of the target objects, as illustrated in Figure 3-4.

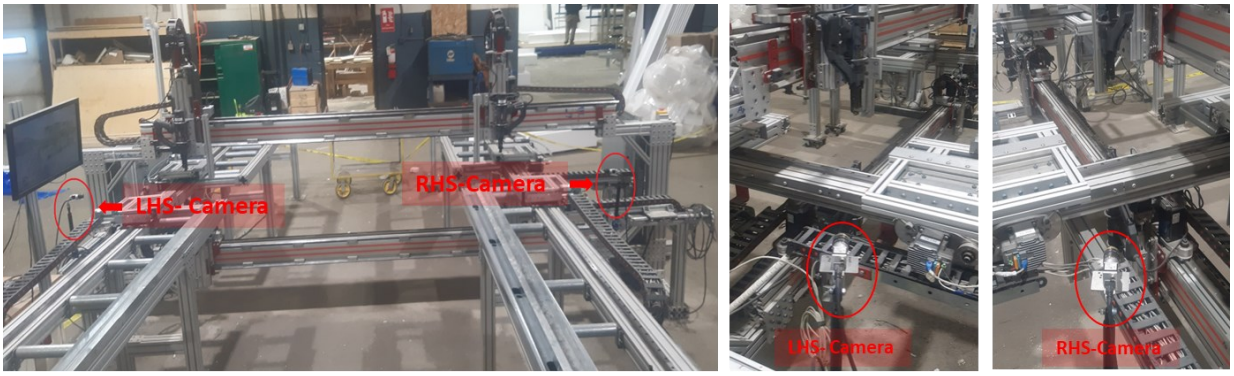



Figure 3-4. Camera positioning for both LHS and RHS

The data collection was detected in multiple stages. Initially, screw positions were manually simulated using a handheld screwdriver, capturing images using custom Python codes at various orientations for each thread of the screw, as shown in Figure 3-5. This approach yielded images for each side, forming the foundational dataset for subsequent model training.

Figure 3-5. Manual screw fastening and image capture

To further expand the dataset, a systematic approach was implemented into the data collection. This was done by constructing multiple panels to mimic real-world scenarios. Videos of the screwdriver manipulation process were captured for both left and right sides. Examples of frames from these videos are shown in Figure 3-6. The frames are extracted using Python codes, resulting in over 500 additional images from each side for training purposes.

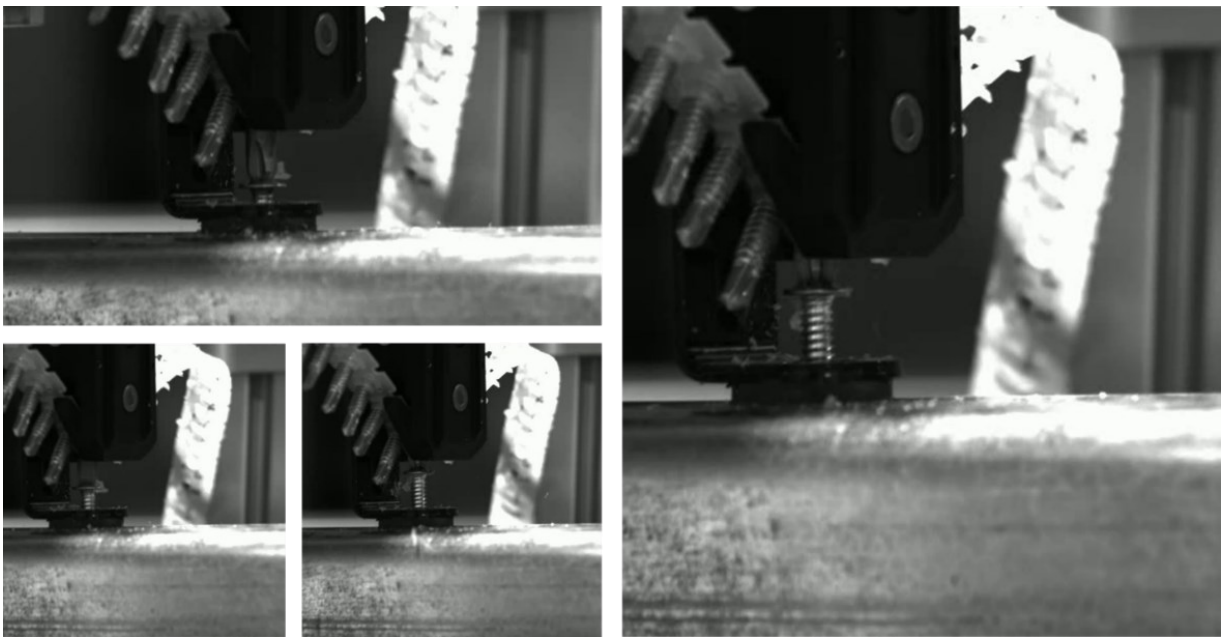


Figure 3-6. Images obtained through video captures

Data augmentation is typically implemented as part of the data preprocessing before feeding the images into the model for training. There are several common techniques for data augmentation:

“Translation”, which shifts the image horizontally or vertically; “Flipping”, which mirrors the image horizontally or vertically; “Shearing”, which skews the image along one of its axes; “Brightness” or “Adjusting”, which adjusts the brightness and contrast of the image; and “Noise Addition”, which adds random noise to the image. These techniques are displayed in Figure 3-7.

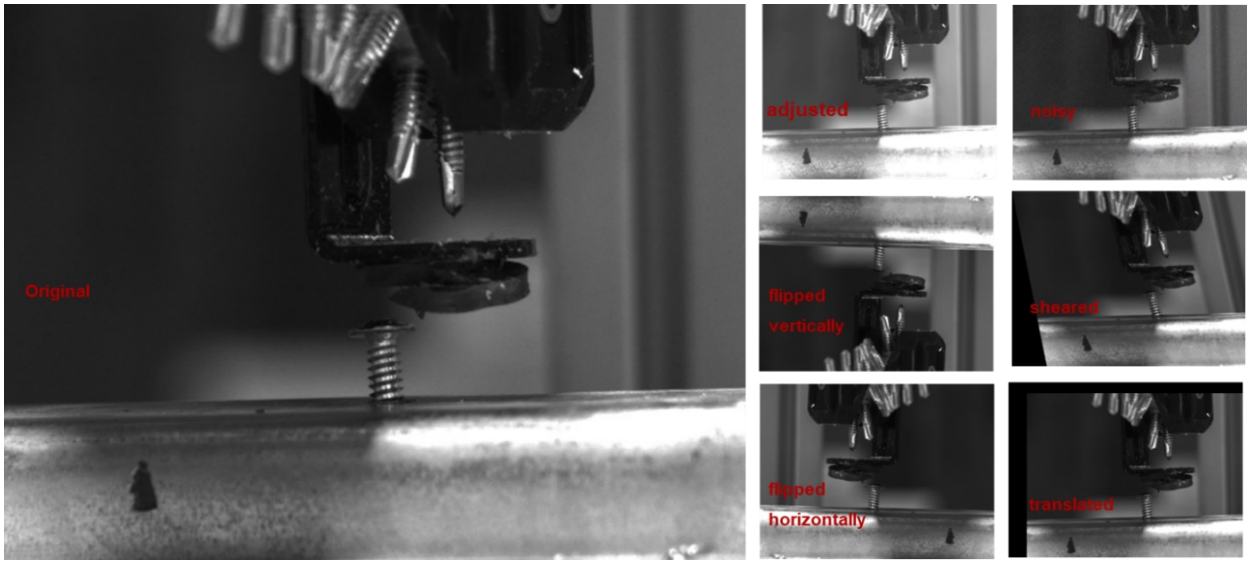



Figure 3-7. Image data augmentation

Python code was used to apply three specific augmentation techniques to 200 images. First, Gaussian noise was introduced to the images, ranging from a minimum of 0 to a maximum of “0.1×255”. Here, 0.1 (10%) represents the proportion of noise added, and 255 signifies the highest pixel intensity value for an 8-bit greyscale image. Second, the brightness of the images was adjusted by multiplying them with a factor between 0.5 and 1.5. Finally, all images in the dataset were horizontally flipped. Figure 3-8 shows the code that has been used for image data augmentation.

```
// Image Data Augmentation
# Define function to draw bounding box on image
    # Input: image, bounding box in YOLO format
    # Convert YOLO coordinates to pixel values
    # Draw rectangle (bounding box) on the image
    # Return image with drawn bounding box and bounding box coordinates
# Define function to augment images
    # Input: directories for images, labels, output images, and output labels
    # Create output directories if not present
    # Retrieve all image files from the images directory
    # For each image:
        # Load image and corresponding label
        # Validate label format
        # If valid:
            # Apply Gaussian noise to the image
            # Increase brightness of the image
            # Flip the image horizontally
            # Adjust bounding box for the flipped image
            # Save the augmented images and labels
        # Display success message for each processed image
# Set directories for images, labels, and outputs
# Execute the augmentation function with specified directories
```

Figure 3-8. Pseudocode for image data augmentation

These augmented images, stored as JPG files, were incorporated into the training process for each model, as outlined in Table 3-1.

### 3.2.2 YOLOv8n architecture

Several methods and frameworks have been developed in object detection and image processing, such as R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2017), and Mask R-CNN (He et al., 2017). The common ground among these models lies in their approach to object detection, which typically involves region-based CNNs and proposals to localize objects within images. Other methods, such as the Single-Shot MultiBox Detector (SSD) (Liu et al., 2016) share the same objective as YOLOv: achieving either highly efficient or real-time object detection. These advancements, including the RetinaNet model introduced by Lin et al., (2017) and EfficientDet by (Tan et al., 2020), have enriched the array of object detection

algorithms. Each technique offers distinct balances between speed, accuracy, and complexity, addressing various application requirements and computational limitations (Terven et al., 2023).

YOLOv8, a new version of a model for detecting objects and segmenting images, which was built upon the earlier success of YOLOv5 (Luo et al., 2023). The YOLOv8 algorithm has developed five distinct models, denoted as YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x, each with varying sub-module depths and widths of the neural network architecture. The model detection accuracy and model size have been improved in sequence, considering the high real-time requirements in actual light-gauge steel (LGS) framing environments. Therefore, YOLOv8n, which has the minimum parameters and model computation, was selected for this study.

YOLOv8n is a multilateral tool for image processing, offering capabilities in image classification, detection, segmentation, tracking, and posing. However, this study's application focuses on its detection capabilities. YOLOv8n, like other detection architectures, operates as a CNN, taking an image as input and producing bounding boxes, classes, and scores for detected objects. This process is shown in Figure 3-9.

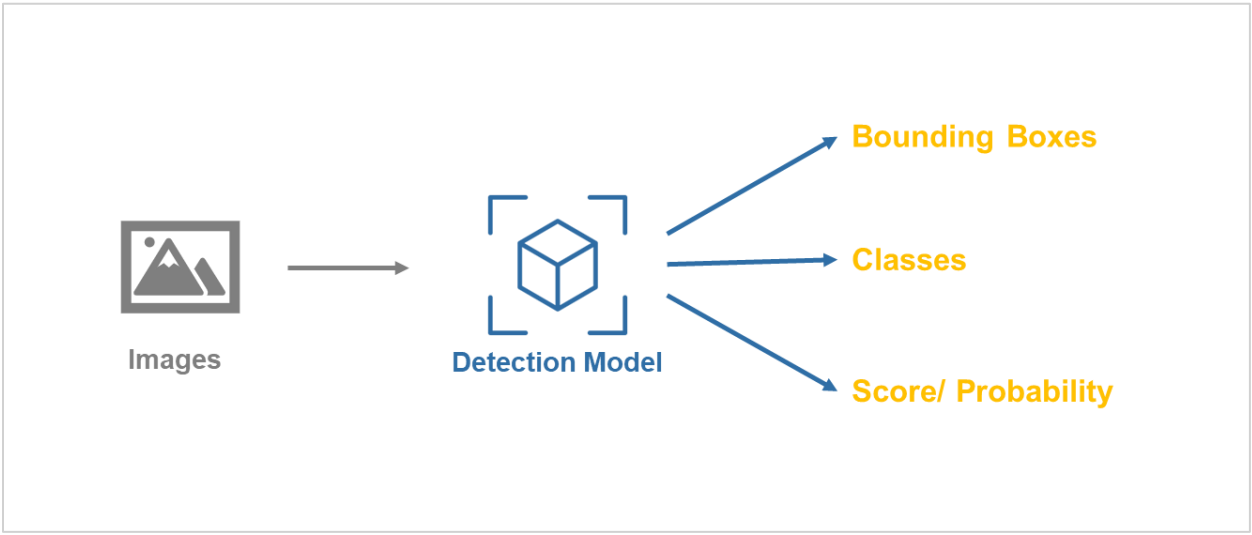



Figure 3-9. General object detection framework

The CNN used in YOLOv8n, referred to as the "Backbone", comprises layers designed for image classification, each capturing different features representing various aspects of input images. Within YOLOv8n, three layers positioned at different depths are responsible for generating feature maps.

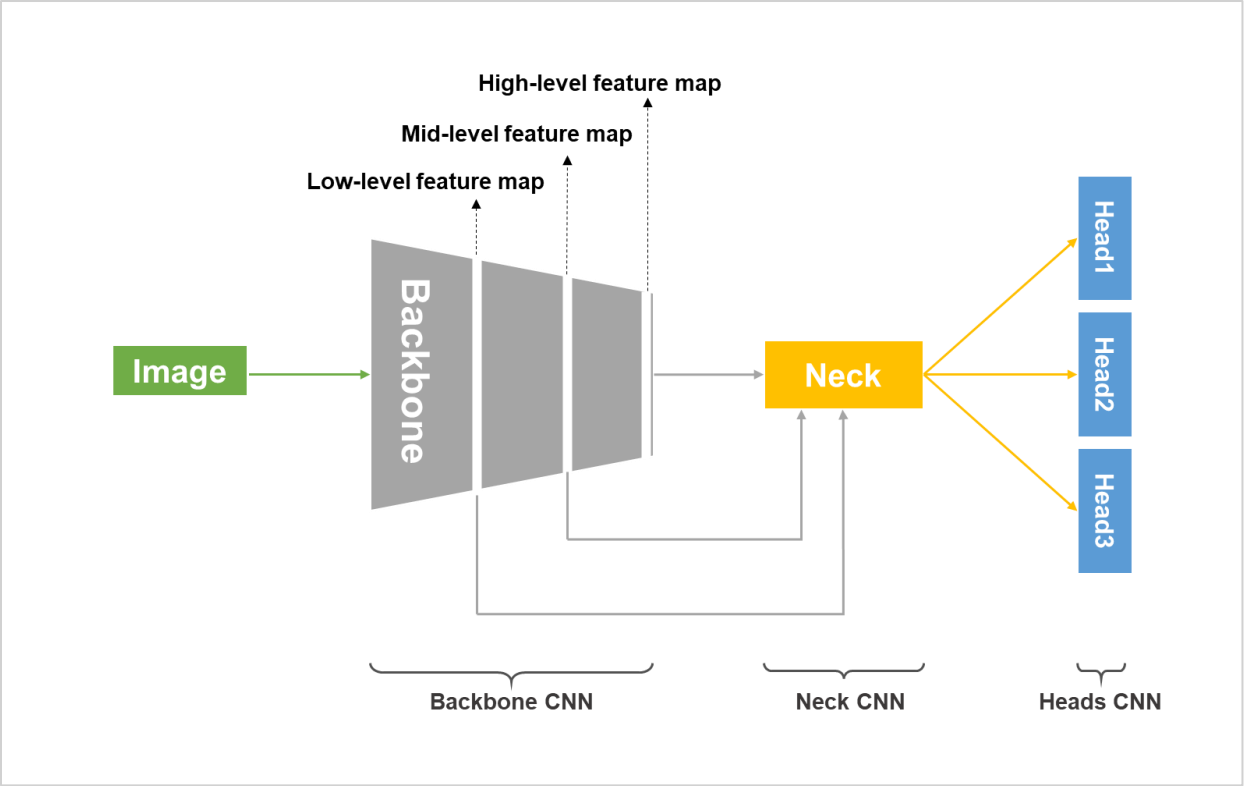



Figure 3-10. Simplified object detection architecture

As shown in Figure 3-10, the YOLOv8 architecture is structured to detect features from the low-level, mid-level, and high-level feature maps. The deeper layers detect larger objects, while shallower layers detect smaller ones, with features becoming more abstracted the deeper they are in the neural network. Each of these three layers is connected to its own CNN, known as "Heads" or "Detection Heads", resulting in three detection heads. Rather than directly feeding these feature maps into their respective detection heads, YOLOv8n incorporates another set of neural network layers called the "Neck" to refine the feature maps. This ensures more accurate and refined detection.

### 3.2.3 Image labelling and YOLOv8n model training on customized dataset

In the initial stage of model training for object detection begins by labelling the captured images. Following this, the images are divided into two sets: a larger portion (80% of images in the dataset) is allocated for training the model, while the remaining subset (20% of images in the dataset) is reserved for robust validation of the model's performance.

During the labelling stage, the dataset is annotated using the CVAT annotation tool (CVAT, 2024; Hansen et al., 2021), defining three distinct classes for data labelling. The annotations are shown in Figure 3-11.

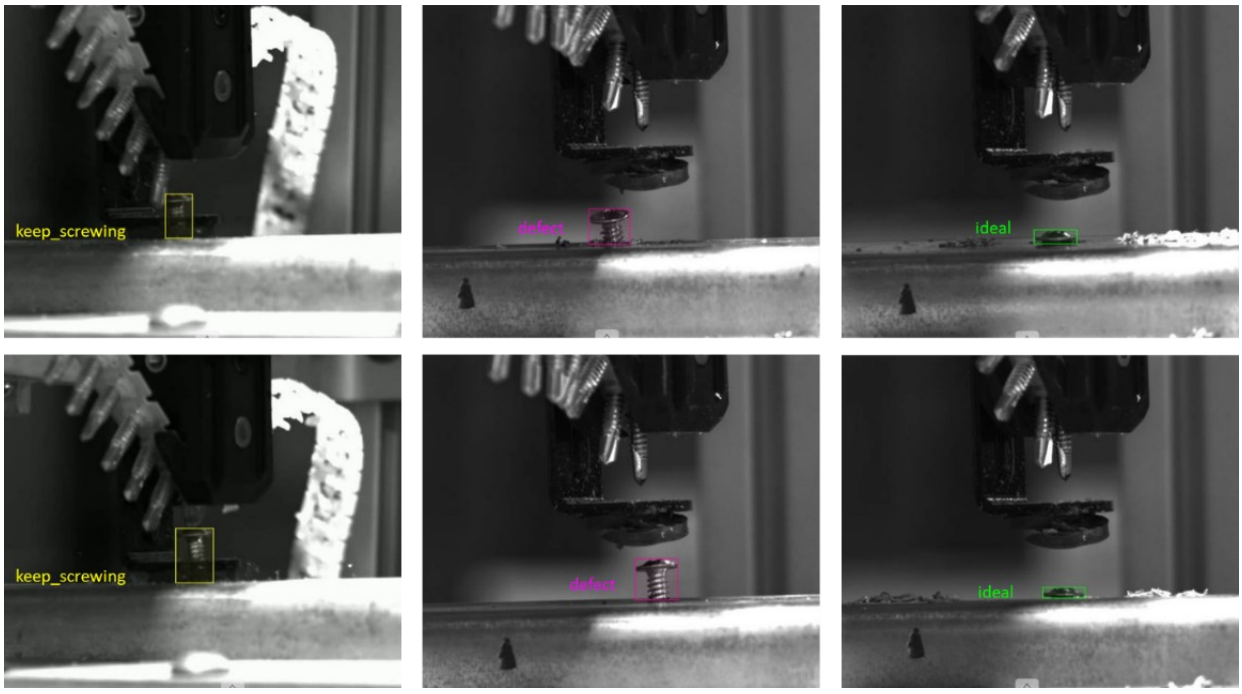



Figure 3-11. Image labelling using CVAT annotation tool

The “keep_screwing” class indicated instances where the screw needed continuous fastening, enabling real-time feedback to the PLC for precise screw-fastening operations. The “defect” class identifies any flaws or irregularities observed in the screw’s position, supporting defect detection. Finally, the “ideal” class represented the desired standard position of the screws, providing detection for ideally-fastened screws.

Once the annotation process is complete, the labelled images from CVAT are saved as text files. Each file contains coordinates in the YOLO format corresponding to different label classes. The files then progress to the training phase. An example of this annotation structure and label coordination is depicted in Figure 3-12.

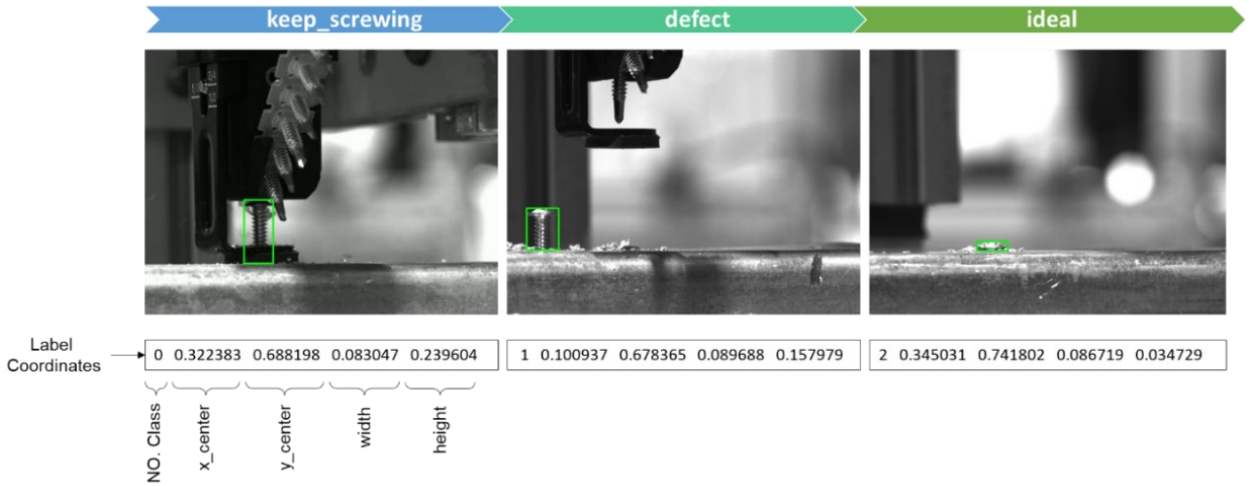



Figure 3-12. Label coordinates in YOLO format for different object classes

As mentioned before, prior to model training, the labelled dataset was partitioned into training and validation sets, allocating 80% of the images for model training and 20% for validation. Additionally, unlabelled images were included as a testing sample. This was done to assess the trained model's performance on unseen data, ensuring its reliability in real-world applications and guiding any necessary further improvements or adjustments. Figure 3-13 shows the dataset allocation.

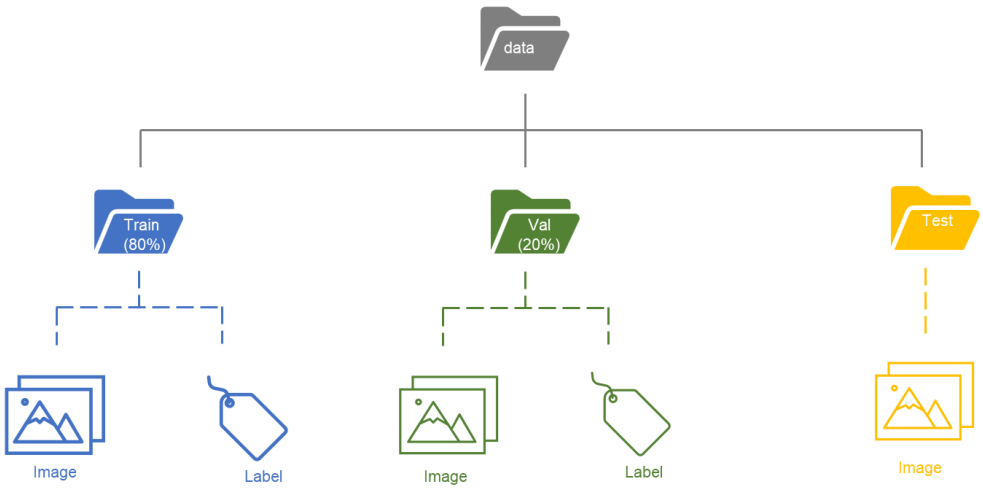



Figure 3-13. Training, validation, and testing dataset allocation

Table 3-1 shows the customized dataset for training, validation, and testing of the customized models for both the LHS and RHS models. Both models are simultaneously trained on the same

dataset, and on different classes specified in the YAML configuration file. Figure 3-14 shows the YAML configuration file for the LHS, while Figure 3-15 shows the YAML configuration file for the RHS.

```
path: '/content/gdrive/My Drive/My_Colab_yolov8n/data' # dataset root dir
train: images/train  # train images (relative to 'path')
val: images/val  # val images (relative to 'path')

names:
  0: keep_screwing_LHS
  1: defect_LHS
  2: ideal_LHS
```

Figure 3-14. YAML configuration file for LHS model training

```
path: '/content/gdrive/My Drive/My_Colab_yolov8n/data' # dataset root dir
train: images/train  # train images (relative to 'path')
val: images/val  # val images (relative to 'path')

names:
  0: keep_screwing_RHS
  1: defect_RHS
  2: ideal_RHS
```

Figure 3-15. YAML configuration file for RHS model training

The dataset comprised a total of 4,118 original images, of which 200 were randomly selected for each of the data augmentation processes of brightening, adding noise, and horizontal flipping. Additionally, 253 negative sample images were included, resulting in 4,971 total images in the dataset. Negative samples in ML are data instances that do not contain any of the target classes or features that a model is being trained to detect or recognize and are therefore labelled as empty. The inclusion of negative samples significantly boosts the performance and reliability of ML models by presenting a more comprehensive and realistic variety of scenarios during training (Liu et al., 2023).

The dataset was then divided into 4,146 training images and 825 validation images. An additional 350 unseen images were also used to test the model.

Table 3-1. Image dataset allocation for model training

| Image Type | Total | Total Dataset | Training | Validation | Testing |
|---|---|---|---|---|---|
| Original | 4,118 | | | | |
| Brightened | 200 | | | | |
| Noised | 200 | 4,971 | 4,146 | 825 | 350 |
| Flipped Horizontally | 200 | | | | |
| Negative samples | 253 | | | | |

One significant challenge was encountered throughout the training process, which was the production of filings produced during the fastening process. Examples of these filings are shown in Figure 3-16.

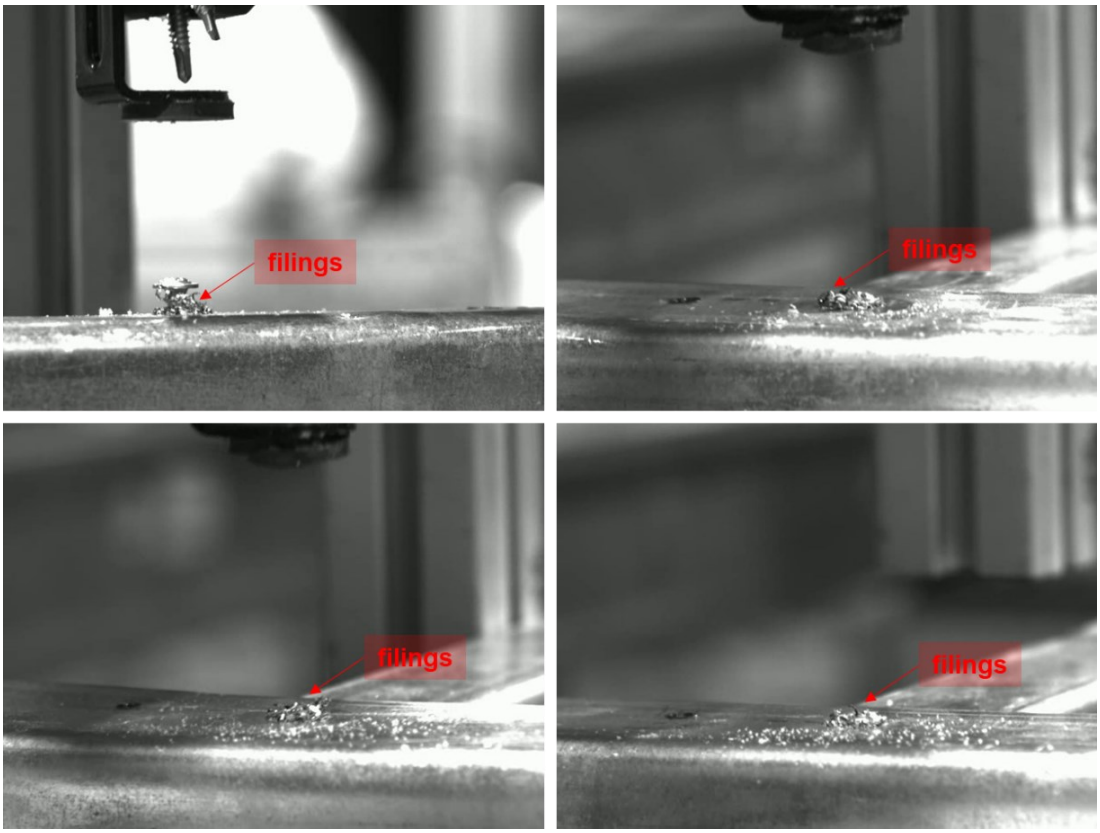



Figure 3-16. Filings produced during the fastening process

These filings accumulate around the screw and block the model's ability to accurately extract features from the detected element, especially the "ideal" class, as the screw could be completely covered by these filings. This obstruction can adversely affect the accuracy of the pre-trained model.

To address this issue, more images with filings were included, which helps the model generalize over obstructions. Precise boundaries were also annotated around the screw to teach the model to distinguish between screws and filings. Additionally, the training data was augmented to represent such scenarios, enabling the model to accurately recognize and classify screws, even in the presence of obstructions.

The model training phase was conducted using Google Colab, leveraging the powerful T4 GPU Hardware accelerator and Python-3.10.12 runtime environment. The YOLOv8n architecture was trained on the customized dataset.

```
// Object Detection Training Script in Google Colab
# Start
# Mount Google Drive
  - Access files through Google Colab's drive library at '/content/gdrive'
# Define data root directory
  - Set ROOT_DIR to path where data is stored on Google Drive
# Install Ultralytics for object detection
  - Run pip install for 'ultralytics'
# Import necessary libraries
  - Use 'os' for file path operations
  - Import 'YOLO' model from 'ultralytics'
# Load and train the YOLO model
  - Load pre-trained model from a specific file
  - Train model with configuration from a file in ROOT_DIR, for 200 epochs
# Set correct locale for Colab
  - Force locale to "UTF-8"
# Archive training results
  - Compress the training output directory into a zip file
# End
```

Figure 3-17. Pseudocode for object detection training in Google Colab

As depicted in Figure 3-17, the training procedure for a YOLOv8n object detection model utilizing the Ultralytics YOLO library begins with the importation of crucial libraries, such as "os", and the inclusion of the YOLO class from Ultralytics. Subsequently, a pre-trained YOLOv8 model is loaded from the file "yolov8n.pt". Following this, the training process is initiated by invoking the "train" method of the YOLO model. The training data directory is specified in a Google Drive account, pointing to a YAML configuration file (located in the same account). Additionally, the number of epochs for training is set to 200.

### 3.2.4 Training results and model evaluation

The confusion matrix displayed in Figure 3-18 serves as a fundamental evaluative tool for classification models, sorting predictions into true positives, false positives, true negatives, and false negatives. Figure 3-18 also outlines the calculations for precision and recall, which are essential accuracy metrics for these models. recall, precision, and Average Precision are the typical metrics used to evaluate performance in object detection tasks.

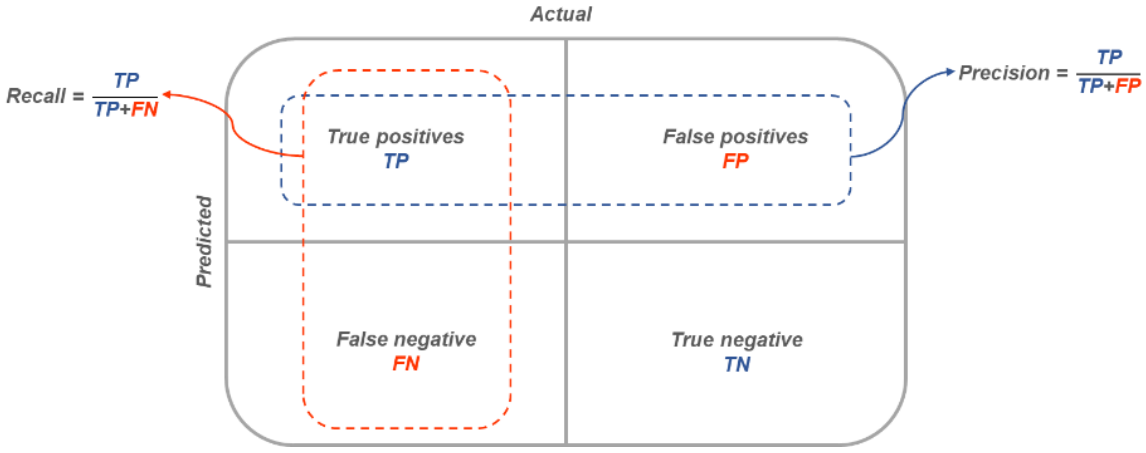



Figure 3-18. Calculation of precision and recall

**Precision and recall**

- **Precision** denotes the ratio of correctly identified objects to the total predicted objects, representing the percentage of correct predictions (Ma et al., 2019). Equation 3-1 shows the calculation for precision.

$$\text{Precision}=\frac{\sum \text{TP}}{\sum(\text{TP}+\text{FP})}\times 100\% \qquad 3\text{-}1$$

where:

TP represents correctly identified objects, and

FP denotes erroneously detected objects.

- **Recall** indicates the ratio of correctly identified targets to the total number of objects (i.e., how well the model performs at finding all the positives) (Ma et al., 2019). Recall is computed using Equation 3-2 provided.

$$\text{Recall}=\frac{\sum \text{TP}}{\sum(\text{TP}+\text{FN})}\times 100\% \qquad 3\text{-}2$$

where:

FN indicates undetected objects.

**Accuracy functions**

The accuracy of a model can be evaluated using mAP, which compares ground-truth bounding boxes with detected ones, providing a score where higher values indicate better detection accuracy. Two variations of mAP commonly used are mAP@0.5 and mAP@0.5:0.95. These metrics utilize Intersection over Union (IoU) and Average Precision (AP) to assess performance.

- **IoU** quantifies the overlap between two boundaries, i.e., how much the predicted boundary overlaps with the ground truth, as shown in Figure 3-19.

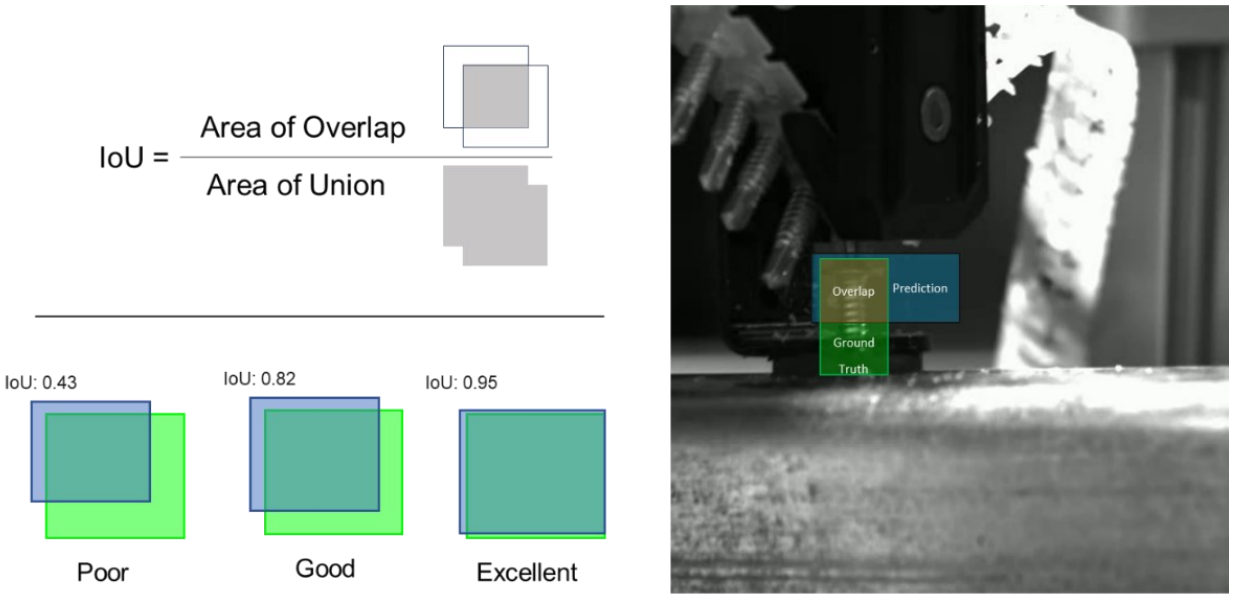



Figure 3-19. IoU

- **AP** is a metric used for object detection accuracy. It quantifies the area under the Precision-Recall (P-R) curve, with recall plotted on the *x*-axis and precision on the *y*-axis. It is calculated using Equation 3-3.

$$AP=\int_0^1 P_i(R_i)dR_i \qquad 3\text{-}3$$

where:
P is the precision value, and
R is the recall value.

- **mAP@0.5** is calculated when the IoU threshold is fixed at 0.5, and AP is computed for each category across all images, and then averaged over categories.
- **mAP@0.5:0.95** represents the average mAP across various IoU thresholds, ranging from 0.5 to 0.95 in increments of 0.05.

In this scenario, accurately detecting the "keep-screwing" class is crucial for determining the height of the bounding box, which is then transformed into a distance measurement. Therefore, the metric that best represents the performance of the trained model is precision.

Precision measures the proportion of true positive predictions for "keep-screwing" instances out of all positive predictions, including both "defect" and "ideal" instances. High precision is essential to ensure that the bounding boxes predicted around "keep-screwing" instances are accurate, as these boxes are used to determine the necessary movements for the screwdrivers.

While recall (which ensures that all instances of "keep-screwing" are captured) is also important, it is not the most critical factor for this scenario. Missing some "keep-screwing" instances could be acceptable.

The metrics mAP@0.5 and mAP 0.5:0.95, which consider the overall performance of the model across all classes and IoU thresholds, provide comprehensive insights. However, they may not directly reflect the accuracy of bounding box placements in this case.

Therefore, when selecting a trained model for custom data focused on accurately detecting "keep-screwing", precision should be prioritized. Nonetheless, it is a good practice to also evaluate the

model using mAP metrics to ensure it maintains robust performance across all classes and IoU thresholds.

Figure 3-20 and Figure 3-21 demonstrate the training results for the LHS and the RHS models.

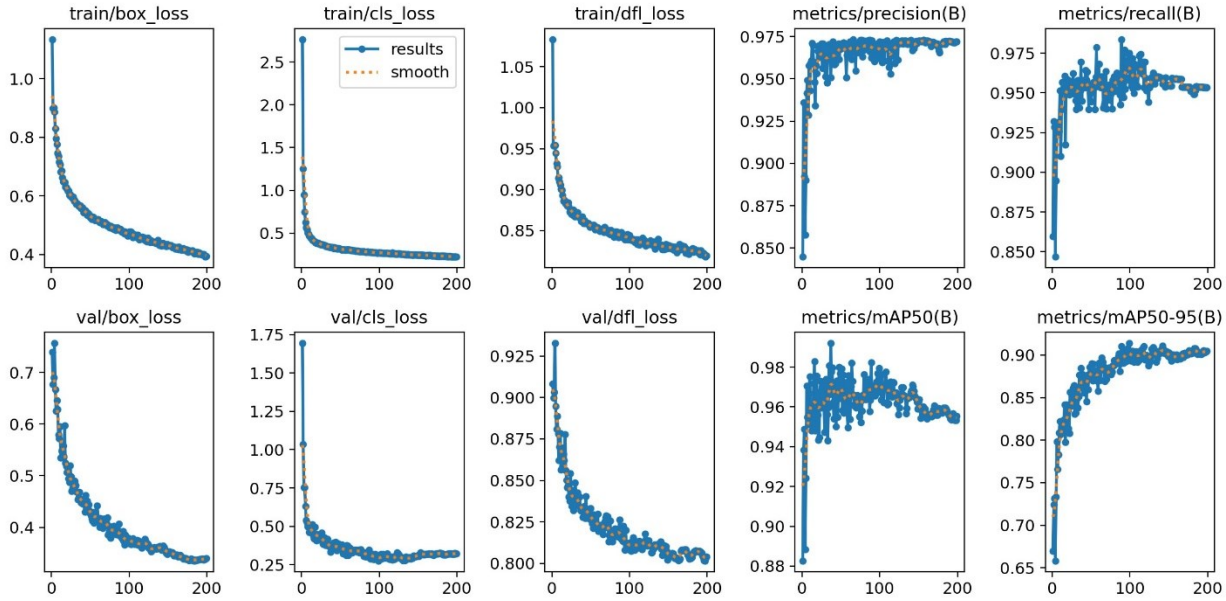



Figure 3-20. LHS model training results

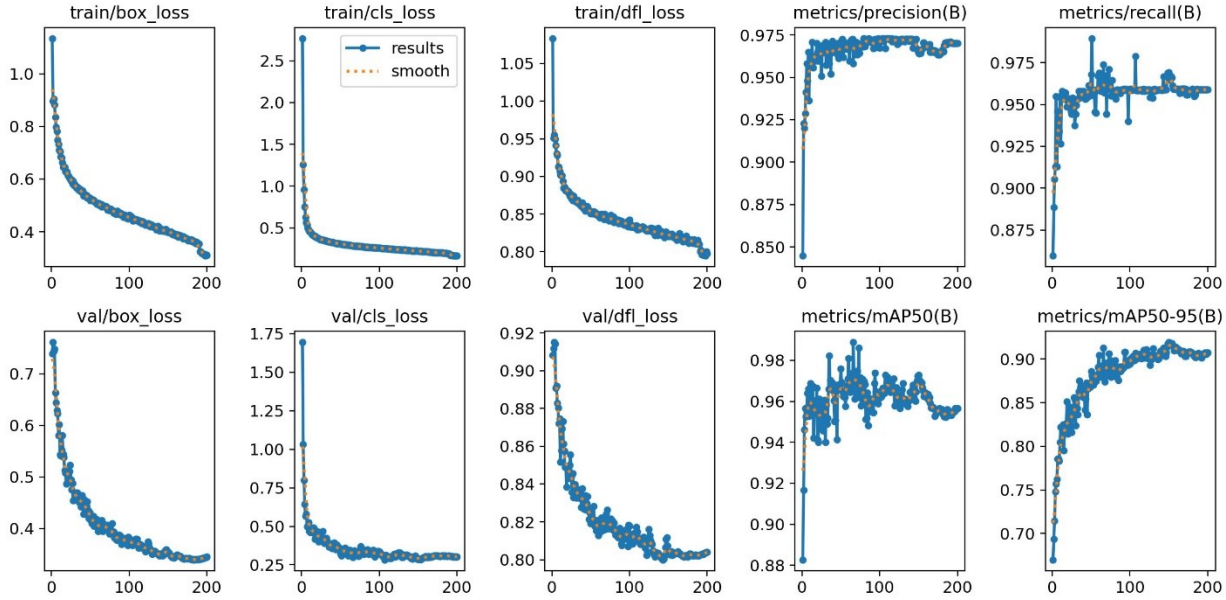



Figure 3-21. RHS model training results

Based on the findings depicted in Figure 3-20 and Figure 3-21, the YOLOv8n model has been effectively trained on a custom dataset to discern between three distinct classes for both LHS and RHS models.

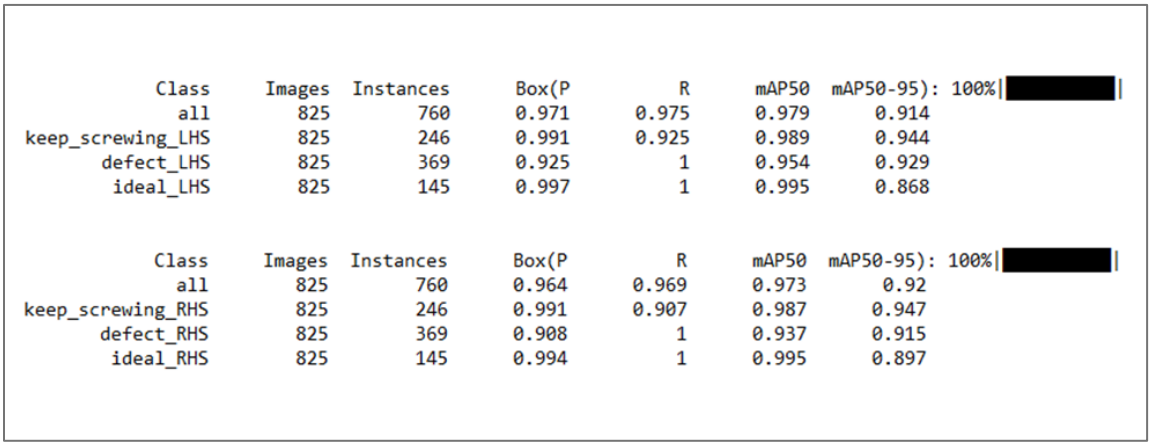



Figure 3-22. Training summary for LHS and RHS models

Figure 3-22 depicts the training summary and evaluation metrics for the "best" performing LHS and RHS YOLOv8n models, showing a slight variation in their performance metrics despite being trained on the same dataset. Both models demonstrate high precision and recall, indicating reliable performance for real-time applications.

The LHS model achieves slightly better precision, recall, and mean mAP scores compared to the RHS model, confirming its excellent detection accuracy. The analysis shows that both models perform consistently well in detecting "keep_screwing" and "ideal" instances, while LHS shows better precision in detecting "defect" instances (the LHS model achieved a score of 0.925, while the RHS model scored 0.908). These minor differences stem from factors such as random initialization, data augmentation, and batch order during training (Golatkar et al., 2019). Overall, both YOLOv8n models are highly effective for real-time applications, demonstrating reliable performance across the various object classes. Additionally, the streamlined design of YOLOv8n ensures fast inference times, which is ideal for real-time applications such as automated quality control. As noted in Figure 3-22, the classes are designated as "keep_screwing_LHS", "defect_LHS", and "ideal_LHS" for the LHS model, and "keep_screwing_RHS", "defect_RHS", and "ideal_RHS" for the RHS model. This classification is illustrated in Figure 3-23, which displays a portion of the training batch used for model training.

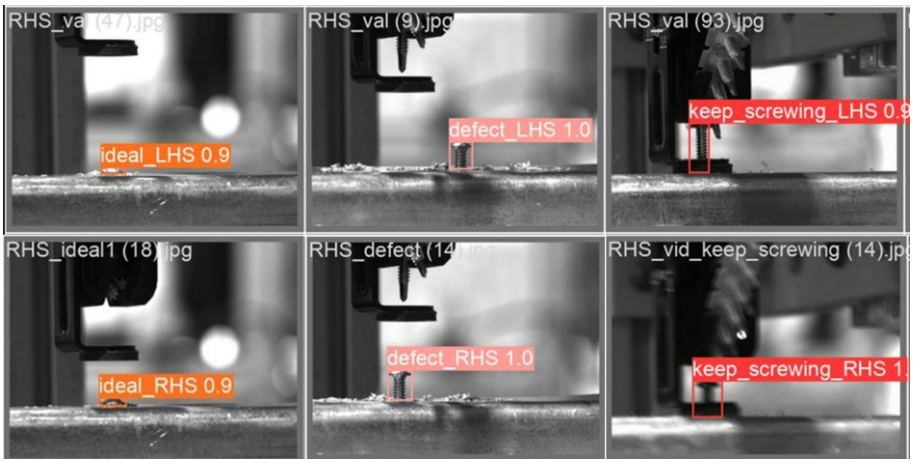



Figure 3-23. Different classes of LHS and RHS models

Most importantly, these metrics are based on the evaluation of the models on a specific dataset, implying that their real-world performance may vary depending on the similarity between the real dataset, and training datasets.

## 3.3 Model testing on unseen dataset

Testing of the models on unseen data is conducted following the training and validation phases. It uses images that the model has not previously encountered, similar to those it might face in real-world scenarios. Unlike the images used during training and validation, these images are unlabelled and are presented to the model only once. The purpose of this phase is to verify that the model delivers consistent and accurate performance on new data. This provides an assessment of the model's effectiveness and robustness, offering valuable insights of the model performance and ensuring its reliability in real-world practical applications.

The code that was used to test the models' performance is depicted in Figure 3-24. The Python code uses a pretrained YOLOv8n model to test it on unseen data images. This code starts with loading required libraries and then sets up the paths for the model, test dataset, and testing results. The code then checks whether a results folder exists and creates one if necessary. Next, the code loads the pretrained YOLOv8n model to initiate processing of the testing dataset, saving both the images and labels for detected objects as text files. Finally, the code prints where the results are saved.

```
# Import Necessary Modules
Load the os module for file and directory operations.
Load the YOLO class from the ultralytics library for object detection.

# Define File and Directory Paths
Specify the path to the trained model weights.
Specify the path to the dataset used for testing.
Specify the path for saving output results.

# Check and Create Output Directory
Ensure the specified output directory exists; if not, create it.

# Initialize YOLO Model
Instantiate the YOLO model using the weights from the specified path.

# Execute Object Detection
Perform object detection on images from the test dataset.
Save detection images and text files with detection details in the output directory.

# Output Results Path
Print the path to the directory where the detection results are stored.
```

Figure 3-24. Pseudocode for YOLOv8n model testing on unseen dataset

Figure 3-25 displays a portion of the test results, conducted on 350 unseen images for both LHS and RHS models.

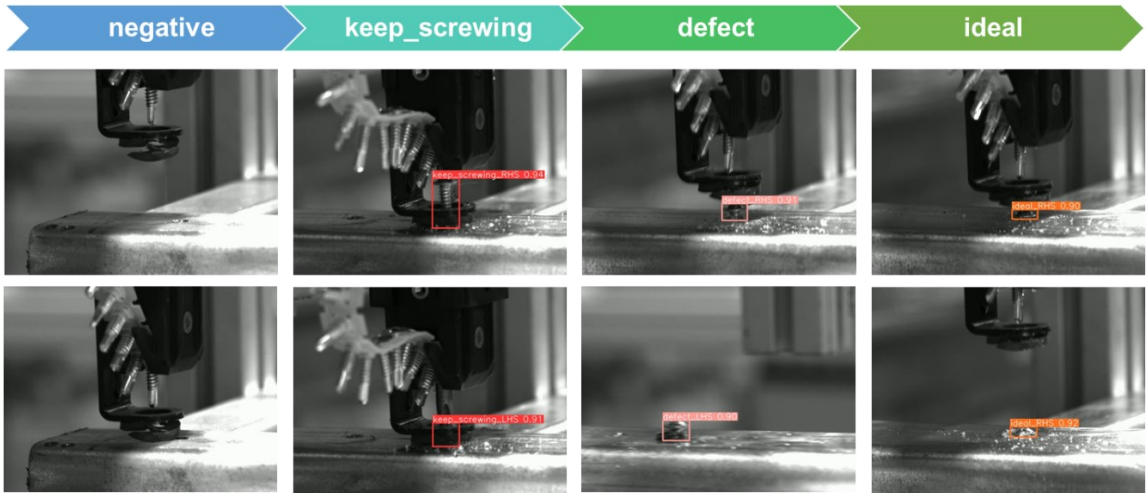



Figure 3-25. Test results of LHS and RHS models on unseen dataset

## 3.4 Camera calibration and positioning

Two cameras were placed (one on each side of the machine), approximately 80 cm from the fastening location, as shown in Figure 3-26.

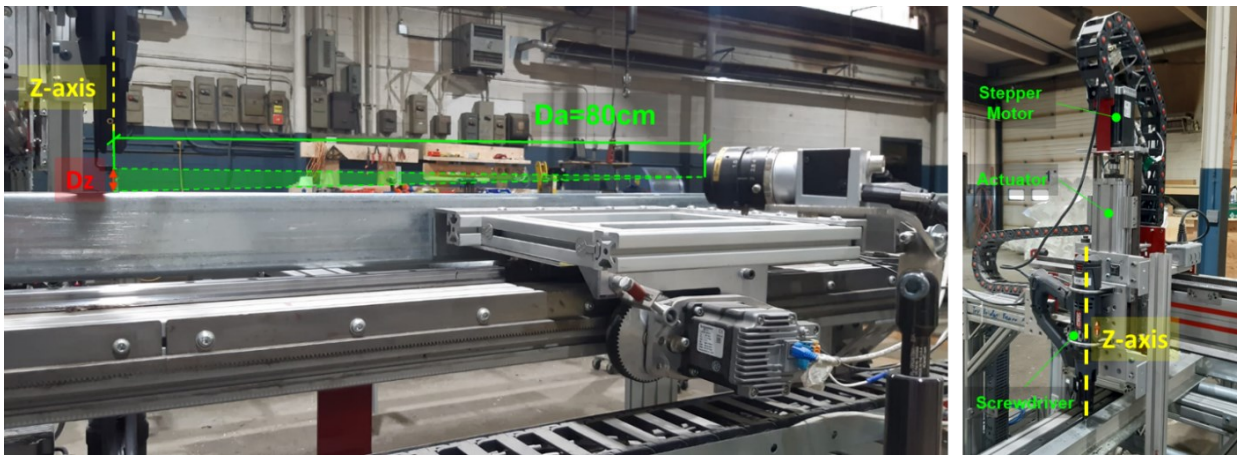



Figure 3-26. Camera configuration

Upon detecting an object using the custom trained YOLOv8n model, the bounding box of the detected element is displayed. From this, the height of the bounding box, which represents the real-time measured distance (Dz) along the *z*-axis, is determined. This distance, measured in pixels, corresponds to the distance required for the actuators to move and fasten the screws to the ideal position.

Table 3-2. Pixels-mm correlation for LHS and RHS cameras

| **LHS** | | **RHS** | |
|---|---|---|---|
| **$D_z$ (Pixels)** | **$D_z$(mm)** | **$D_z$ (Pixels)** | **$D_z$(mm)** |
| 89.8 | 14 | 83.2 | 14 |
| 86.5 | 13 | 79.5 | 13 |
| 79.1 | 12 | 76.6 | 12 |
| 69.1 | 11 | 69.2 | 11 |
| 62.6 | 10 | 61.7 | 10 |
| 54.7 | 9 | 54.1 | 9 |
| 46.9 | 8 | 46.8 | 8 |
| 40.3 | 7 | 42.1 | 7 |
| 34.7 | 6 | 36.8 | 6 |

In order to be transferred to the PLC, the pixel measurements must be converted to millimetres. This was done by conducting numerous measurements for both the LHS and RHS. The results were recorded in Table 3-2. Scatter charts were created for both tables using Microsoft Excel, illustrating the correlation between the height of the bounding box (in pixels) and the actual measurement of the screw (in millimetres) using a measuring tape.

Upon reviewing the scatter charts in Figure 3-27, it was observed that the correlation between height in pixels and the measured height in millimetres is nearly linear. Subsequently, trend lines were added to both scatter charts, allowing for the derivation of equations that represent these lines. Equation 3-4 represents the LHS camera, and Equation 3-5 represents the RHS camera.

$$y = 0.1364x + 1.458 \quad 3\text{-}4$$

$$y = 0.1593x + 0.2642 \quad 3\text{-}5$$

Here, $x$ represents the Dz (distance) value in pixels along the $z$-axis, and $y$ represents the Dz (distance) value in millimetres along the $z$-axis.

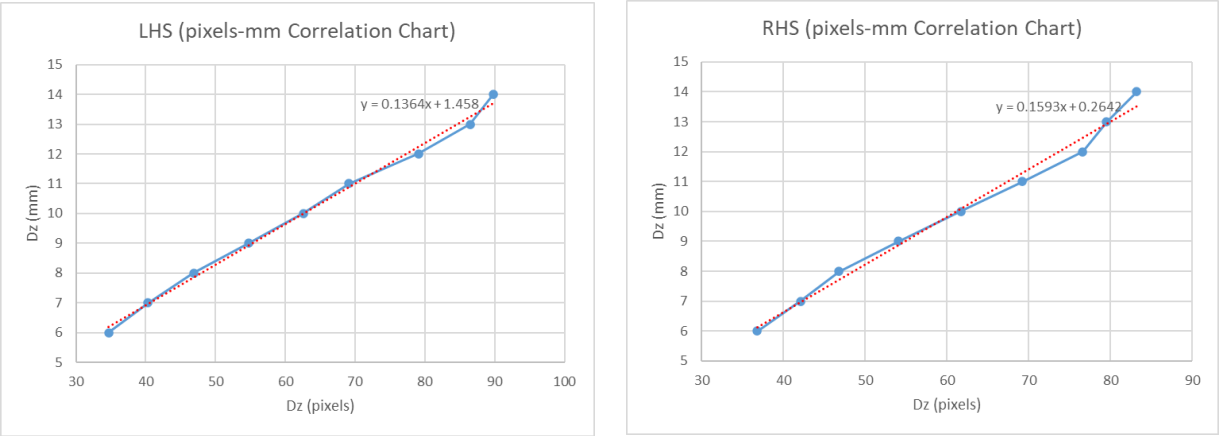



Figure 3-27. Pixels-mm correlation charts for LHS and RHS cameras

Equations 3-4 and 3-5 were then integrated into the Python code to convert and transmit the measured pixels ($x$) to millimetres ($y$ = Dz) as a final displacement value for LHS and RHS actuators, respectively.

To ensure the accuracy of converting pixels to millimetres, a real-world scenario was emulated, as illustrated in Figure 3-28 . In this simulation, measurements obtained from the code were compared with those from a physical measuring tape to verify if they match. This process was repeated with both the LHS and RHS cameras and used various screw heights.

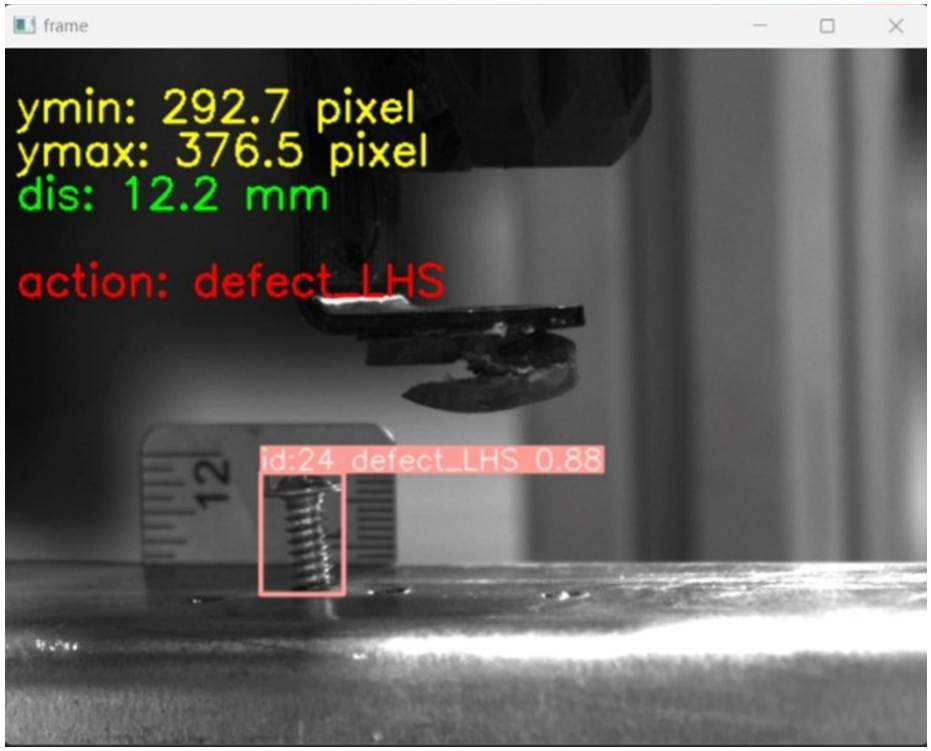



Figure 3-28. Validation of cameras measurements

Upon confirming the precise measurement captured by the cameras for the detected elements, the computer vision system (CVS) framework was validated and considered ready for integration with the control system of the LGSFM.

## 3.5 PLC code modifications and addition of quality-check steps

A new step named "Quality-Check" is added into the PLC logic to handle the measured values ($D_z$) obtained from Equations 3-4 and 3-5. These values are then converted into a corresponding number of steps for the stepper motors driving the actuators on both LHS and RHS. In the PLC logic, the $D_z$ values are adjusted to determine the number of steps required for the stepper motors, based on the specifications of the actuators holding the screwdrivers. According to these specifications, each 1 mm linear displacement necessitates 800 motor steps. Therefore, the equation used to calculate the number of steps required ($n_{steps}$) is given by:

$$n_{steps}=800\times D_z \qquad 3\text{-}6$$

Here, $n_{steps}$ in Equation 3-6 represents the real-time number of steps that must be transferred from the PLC to the motor's driver. This approach allows for precise tracking of the number of real-time steps needed for the motor to achieve the desired displacement.

Figure 3-29 shows the modifications in the original PLC code, clarifying the additional “Quality-Check” steps on both the LHS and RHS.

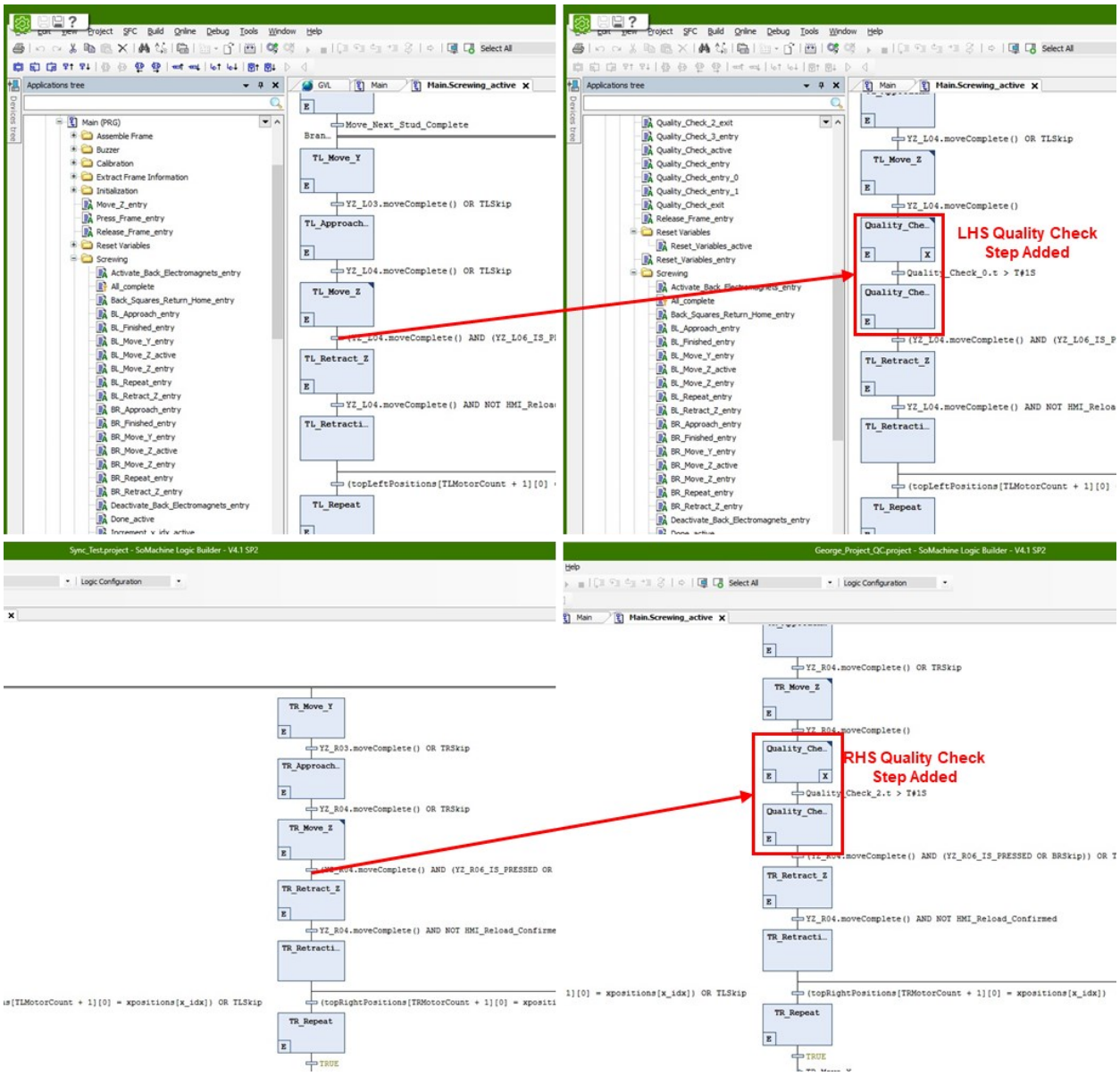



Figure 3-29. PLC code modifications and addition of quality-check steps

## 3.6 PLC–Python communication framework

The PC employs Python and the YOLOv8n ML model to interact with the machine’s PLC over Modbus protocol, facilitating the exchange of control signals and data. This PLC governs the whole control system of the LGSFM, thus integrating real-time monitoring and automated quality control into the machine’s setup. The framework underlying this process is depicted in Figure 3-30.

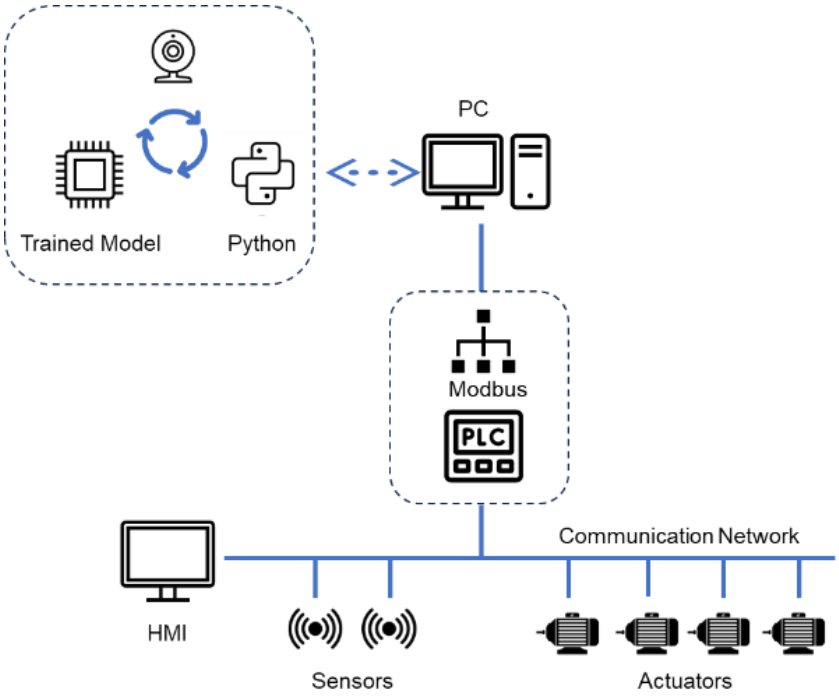



Figure 3-30. PLC-Python communication framework

The communication between the PLC and Python code via a PyCharm environment is settled entirely by Python code. This code, shown in Figure 3-31, has two features. First, it continuously reads frames from the two cameras, performs object detection and tracking, and calculates the real-time distance of detected objects. Second, it facilitates communication with the PLC via the Modbus TCP/IP protocol while incorporating computer vision capabilities. This code imports all the necessary libraries, including “pymodbus” library for Modbus communication, “os” library for environment variable manipulation, OpenCV “cv2” library for video capture and processing, and “ultralytics.YOLO” for object detection and tracking using a YOLO model. After setting the environment, the two custom YOLO models (LHS and RHS models) are loaded. Modbus TCP/IP connection parameters, such as the PLC’s IP address and Port number, are defined, along with the PLC’s holding register addresses. The code establishes a connection with the Modbus server and initializes a video capture object to read frames from the camera.

```
# Import libraries
IMPORT necessary libraries

# Set environment variable
SET environment variable for compatibility

# Load YOLO model
LOAD YOLO model from specified path

# Setup Modbus client
DEFINE Modbus server IP, port, and slave address
CONNECT to Modbus server

# Initialize video capture
INITIALIZE camera for capturing video

# Process frames from the video
WHILE there are frames to read
    CAPTURE frame from video
    DETECT objects using YOLO model
    IF objects are detected
        CALCULATE object dimensions
        TRANSLATE dimensions to Modbus values
        WRITE values to Modbus registers
        DISPLAY results on the frame
    SHOW the processed frame
    IF 'q' key is pressed
        BREAK the loop

# Cleanup
RELEASE camera
CLOSE Modbus connection
```

Figure 3-31. Pseudocode for Python-PLC communication

Within the main loop of the code, frames are continuously processed for object detection and tracking. Meanwhile, the detected objects' distance on the machine's *z*-axis is calculated and written to specific Modbus holding registers. This achieves real-time interaction between computer vision tasks and the automated fastening process through Modbus' communication with the PLC.

In this setup, the screws have a maximum length of 20 mm, corresponding to the maximum distance the screwdriver must move during fastening. When converting pixel measurements to millimetres, non-integer values are frequently encountered. In this case, the value is typically rounded to one decimal place. To facilitate communication with the PLC, these values are multiplied by 10 in Python to ensure they are sent as integers. Consequently, the maximum value sent to the PLC is 200 (20mm×10).

For the machine's PLC MODICON M251, it is important to note that each data type is addressed differently in the memory. Each data type, whether bit (%MX), byte (%MB), word (%MW), or double word (%MD), has specific considerations and sizes, as shown in Figure 3-32 (Schneider Electric, 2019).

| Double Words | Words | Bytes | Bits | | |
|---|---|---|---|---|---|
| %MD0 | %MW0 | %MB0 | %MX0.7 | ... | %MX0.0 |
| | | %MB1 | %MX1.7 | ... | %MX1.0 |
| | %MW1 | %MB2 | %MX2.7 | ... | %MX2.0 |
| | | %MB3 | %MX3.7 | ... | %MX3.0 |
| %MD1 | %MW2 | %MB4 | %MX4.7 | ... | %MX4.0 |
| | | %MB5 | %MX5.7 | ... | %MX5.0 |
| | %MW3 | %MB6 | %MX6.7 | ... | %MX6.0 |
| | | %MB7 | %MX7.7 | ... | %MX7.0 |
| %MD2 | %MW4 | %MB8 | %MX8.7 | ... | %MX8.0 |
| | | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... |
| | | ... | ... | ... | ... |

Figure 3-32. Memory addressing table for PLC-MODICON M251

Bits carry binary logic (0 or 1), while a byte comprises 8 bits. Similarly, a word encompasses two bytes, and a double word comprises two words. Figure 3-33 shows the hierarchical structure of the memory addresses for the PLC-MODICON M251.

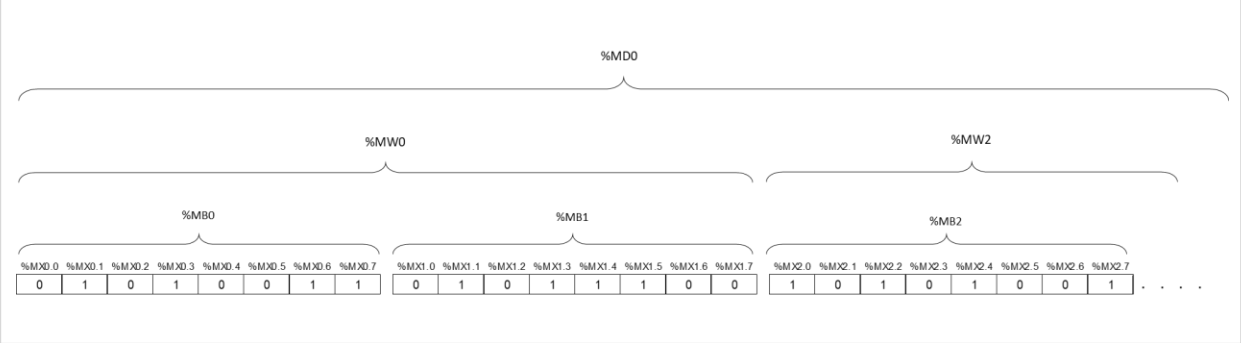



Figure 3-33. Hierarchical structure of PLC-MODICON M251 memory addresses

The determination of the appropriate memory address type for storing the values depends on the range of values that must be transmitted. Considering the maximum value of 200, which fits comfortably within the byte's capacity, each value can be sent using a single byte. The byte's capacity, calculated as the sum of powers of 2 from 0 to 7, shown in Equation 3-7, provides a maximum value of 255 well above the required 200.

$$1\times2^0+1\times2^1+1\times2^2+1\times2^3+1\times2^4+1\times2^5+1\times2^6+1\times2^7=255 \quad 3\text{-}7$$

The application requires four byte-type registers to be stores in the PLC in order to capture values sent from Python code. Each register is dedicated to its specific function within the PLC's code. Specifically, two registers are allocated to retrieve class-related values ("keep_screwing", "defect" and "ideal") for both LHS and RHS. The remaining two registers are designated for reading values representing the distance along the *z*-axis for the LHS and RHS screwdrivers.

It is assumed that, when the model detects the "keep_screwing" class, a value of "0" is transmitted to the PLC. For the "defect" class, a value of "1" is sent, while, for the "ideal" class, a value of "2" is sent. The distance values indicating displacement along the *z*-axis for the LHS and RHS range from 0 to 200. This communication protocol ensures the efficient transmission of screwdriver movement data and detection outcomes to the PLC.

## 3.7 Experiment implementation

In the experiment, five panels are constructed from LGS frames, each comprising two tracks and five studs fastened together using steel-to-steel self-tapping screws. A 3D model of a sample panel is depicted in Figure 3-34.

Figure 3-34. 3D model of sample panel

The studs used are cold-formed steel molded into a C-shape with a lip return. They are 14-gauge studs measuring 43 mm × 95 mm. The tracks are U-shaped and lack a lip return, allowing studs to fit inside them. (Tracks are typically used to cap the top and bottom of a steel stud wall.) In this experiment, 14-gauge tracks, measuring 53 mm × 98 mm, are used. The screws are self-tapping, measuring 4.5 mm × 20 mm, and are specified for steel-to-steel connections compatible with SENCO DS242-AC screwdrivers. These materials are shown in Figure 3-35.

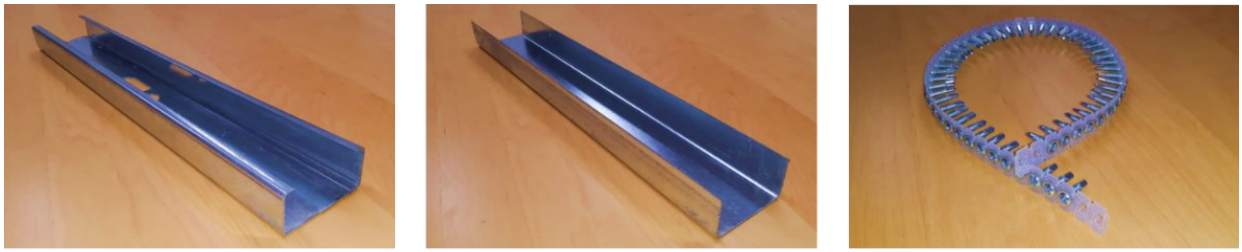



Figure 3-35. Experimental materials

During the initial experiment, the machine operates with its original code (i.e., without any CVS integration). In the second experiment, the proposed framework is implemented, integrating a CVS into the machine's original code. The purpose of the comparison between the two experiments is to demonstrate the enhancement in the quality of screw fastening and the mitigation of screw defects during the manufacturing process of LGS frames.

The process begins with the creation of a 3D model of the panel using Revit. An RCP file is then generated using an RCP file generator. An RCP file, often referred to as a recipe file, is a type of file used in industrial automation, particularly in PLCs. RCP files are typically designed to be compatible with specific PLC models, such as MODICON M251, and they can be created using programming or configuration tools. These files typically contain parameters that define how a specific process should be executed or controlled by the PLC. Figure 3-36 shows the RCP file generator that was used in the application.

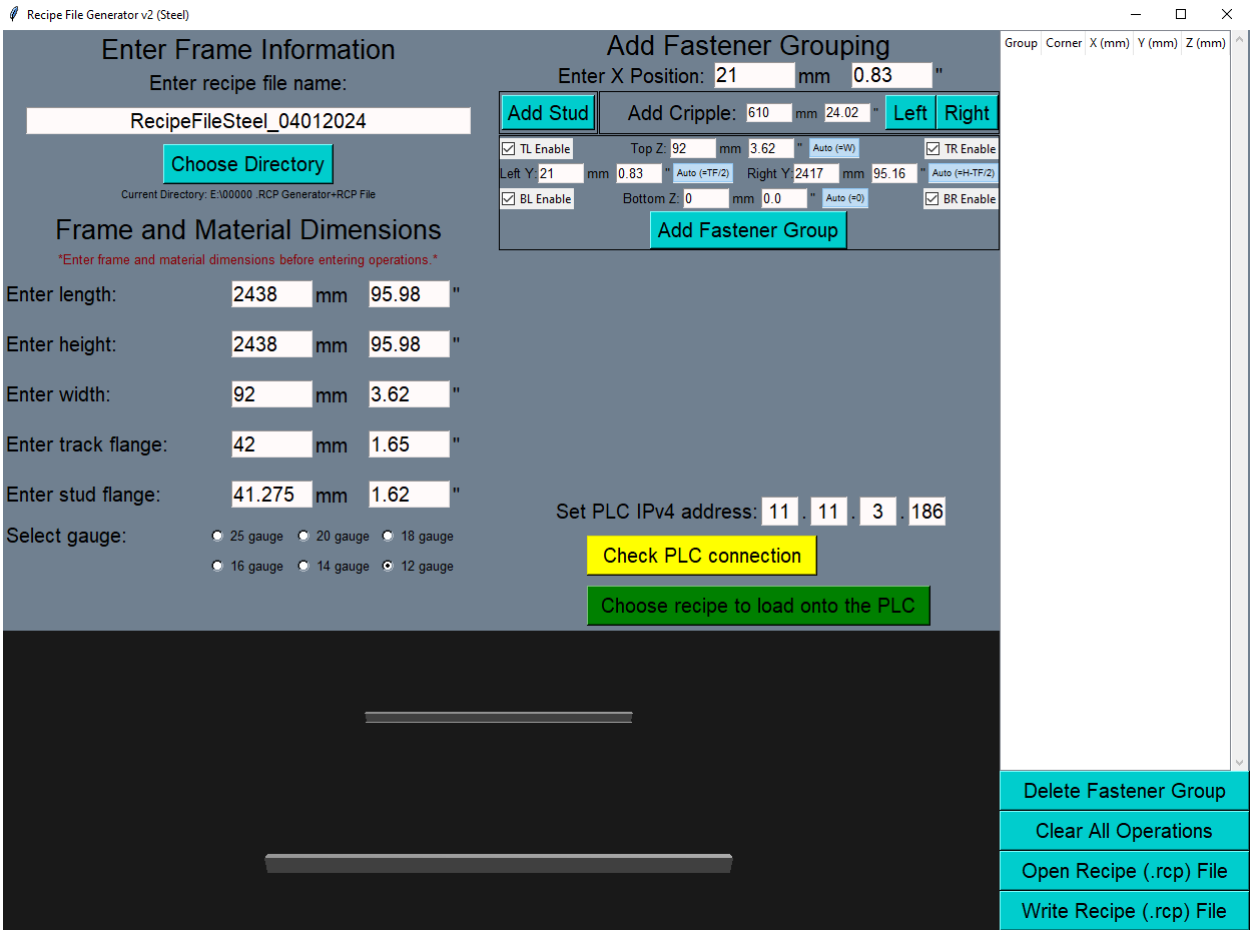



Figure 3-36. The RCP file generator for LGSFM

The RCP file in this application specifies a sequence of steps that the PLC should follow to carry out a particular process. These steps, illustrated in Figure 3-37, include the following parameters:

- Main.frame_length_x: Specifies the length of the frame in the *x*-axis in millimetres.
- Main.frame_height_y: Specifies the height of the frame in the *y*-axis millimetres.
- Main.frame_width_z: Specifies the width of the frame in the *z*-axis in millimetres.

These parameters define the dimensions of the frame involved in the process. Other parameters defined in the RCP file are as follows.

- entrycount: Indicates the number of entries or operations in the RCP file.
- VIS_File_Name: Indicates the RCP file name.
- stringArray: Defines coordinates and dimensions for different points in the process. Each line represents a point, and each point has *x*-, *y*-, and *z*-coordinates, along with a label.

These labels are TL, TR, BL, and BR, which denote Top-Left, Top-Right, Bottom-Left, and Bottom-Right, respectively.

- xpositions: This section defines *x*-positions of the studs in the panel structure.

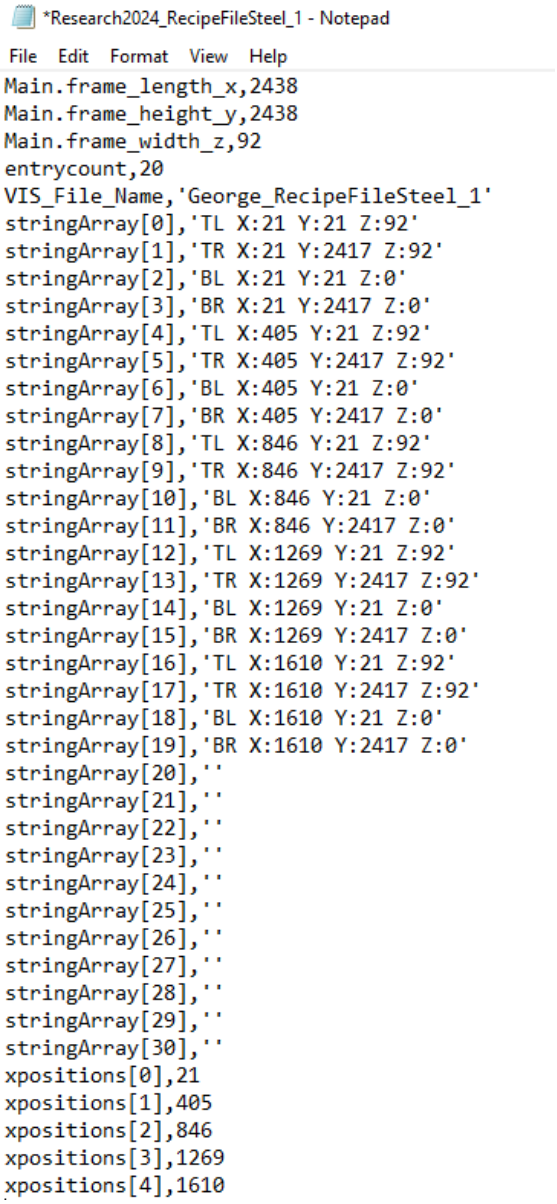



Figure 3-37. Structure of RCP file

Overall, the RCP file configures a framing process for the LGSFM to control the positioning of various components.

After uploading the RCP file to the machine's PLC, two Basler aca640-90um cameras (LHS and RHS) are prepared for the machine. These cameras are operated using Python code and utilized in

conjunction with two trained models to detect screws during the manufacturing process. The experimental results are stored in real time to a dedicated MySQL database. The results and machine metrics are then displayed on a dashboard developed in Grafana, serving as an online monitoring tool. The detailed setup of the database and the dashboard configuration are discussed in Chapter 4.

### 3.7.1 Experiment 1: Original control system without CVS integration

This experimental step was implemented based on the original control system of the LGSFM. The PLC oversees the entire process using the PLC code and RCP file derived from the panel's 3D model. The fastening process, involving inserting self-tapping screws into the frame, follows a fixed number of steps that are dictated by the PLC and transferred to the stepper motor drivers. Figure 3-38 illustrates the current configuration of the LGSFM.

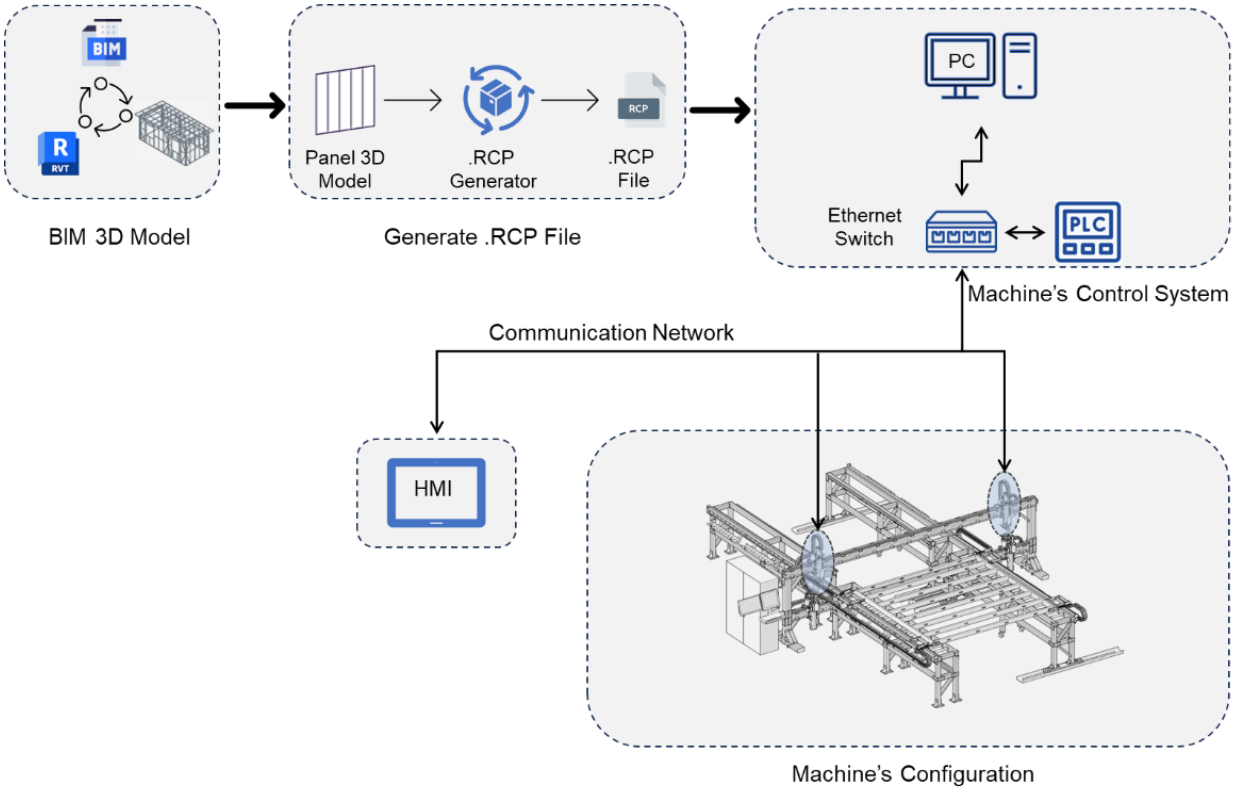



Figure 3-38. System configuration without CVS framework

However, as previously outlined in Chapter 2, the causes of defects include misalignment between stations or unleveled tables, which makes the static calibrated "approach" position ineffective.

Additionally, the use of bendable or flawed materials can lead to deviations in the screwdriver's intended position. Mechanical tolerances in actuators during the fastening process may also produce *z*-axis movements, resulting in gaps that affect fastening quality. Without correction or feedback signals to change the "approach" position dynamically, the machine becomes vulnerable to producing defects, as shown in Figure 3-39.

These defects highlight the need for a feedback system integrated into the PLC, such as a CVS. This system would offer real-time feedback to the PLC, indicating the necessary displacement in millimetres for the actuators holding the screwdrivers. The PLC would then convert these measurements into precise number of steps for the stepper motors controlling the actuators. This process ensures accurate and dynamic adjustments in real-time to achieve the optimal "approach" screw positions.

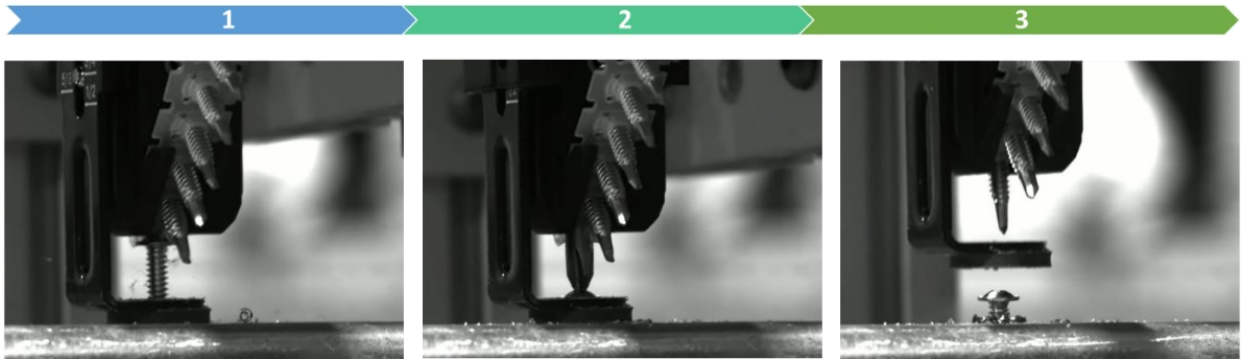



Figure 3-39. Experiment implementation based on current system configuration

### 3.7.2 Experiment 2: Proposed CVS framework

In this experimental setup, two cameras were positioned at the upper left and upper right corners of the fastening area to effectively capture images of the screw-fastening process. Employing Python code within the PyCharm environment, seamless communication was established between the cameras and the PLC. This integration facilitates real-time monitoring and control of the screw-fastening procedure. The system configuration, the integration between the machine's control system, and the proposed CVS are depicted in Figure 3-40, illustrating the setup implemented for efficient monitoring and management of the manufacturing process.

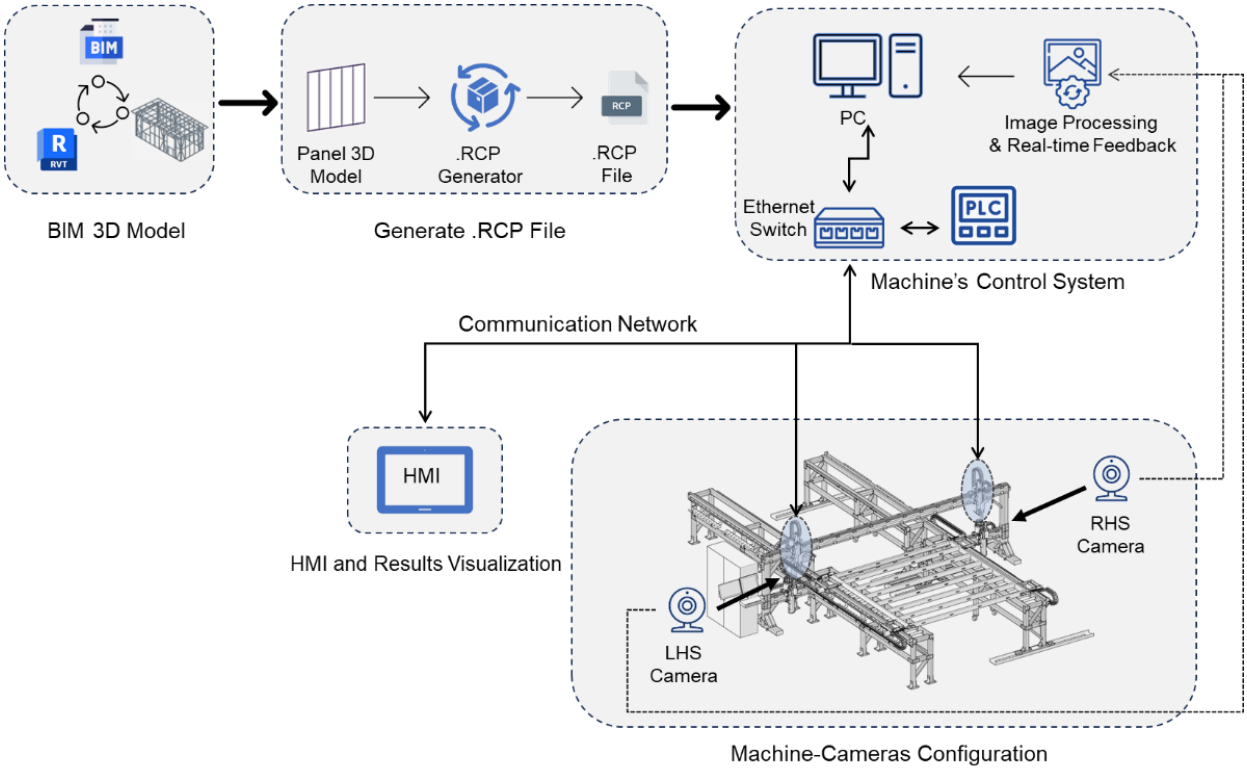



Figure 3-40. System configuration using CVS framework

Figure 3-41 illustrates the screw-fastening process, depicting how real-time screw detection occurs during fastening. This includes capturing the distance the screwdriver moves along the *z*-axis to achieve the ideal screw status.

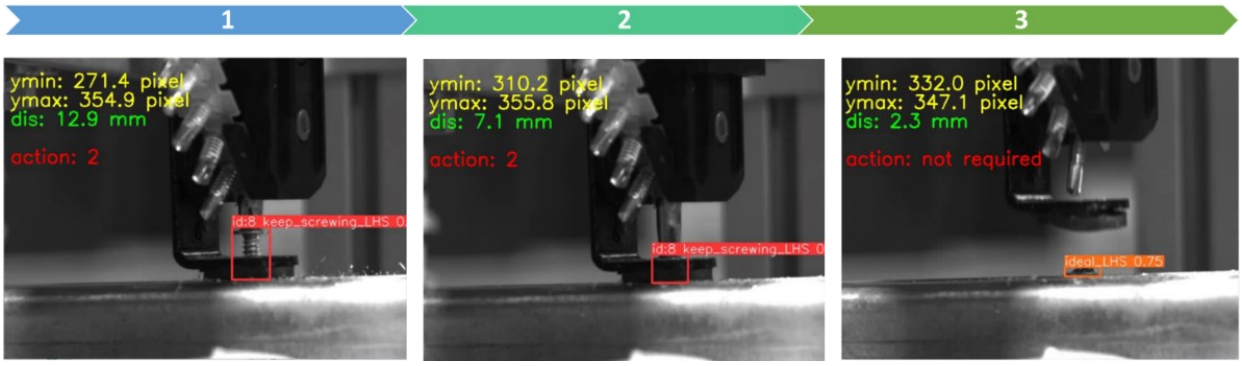



Figure 3-41. Experiment implementation using proposed CVS framework

Additionally, as shown in Figure 3-42, an enhanced HMI was added by incorporating a feature that displays the status of each screw on both the left and right sides throughout the manufacturing process. This additional feature serves as a visual representation, ensuring operators can easily

track the real-time status of screws as they progress through the fastening process, facilitating immediate quality control measures.

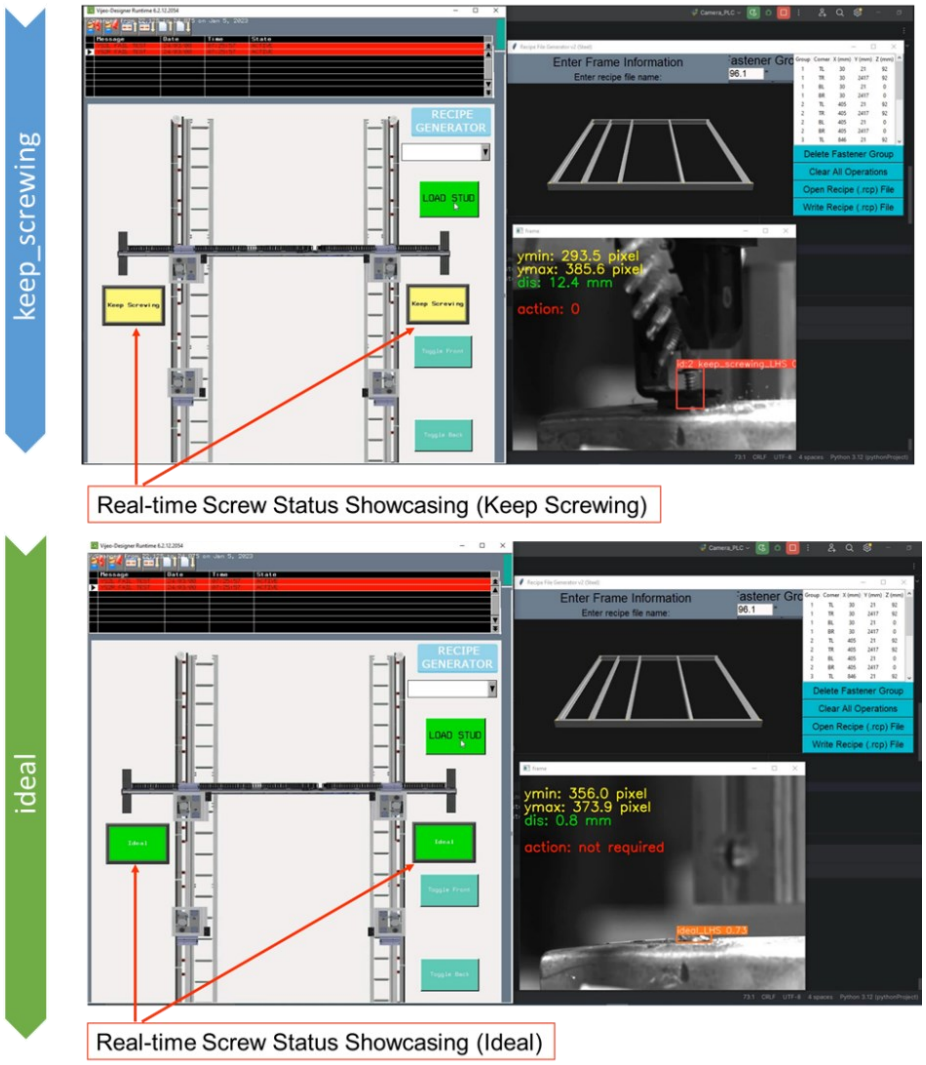



Figure 3-42. Real-time screw-fastening status visualization using machine's HMI

### 3.7.3 Results and discussion

Five experiments were conducted to construct the same LGS panel comprising two tracks and five studs, as previously detailed. The initial experiment adhered to the original system configuration without incorporating the proposed CVS framework, while the second experiment integrated the CVS framework with the existing control system.

Table 3-3 compares the results of the two experiments, revealing a notable difference between the presence and absence of the CVS. The table details the status of the screws on the LHS and RHS

across five different panels, assessing the effectiveness of CVS in detecting screw-related defects. “1” indicates a defect, and “0” indicates no defect. The “Defects Rate” indicates the percentage of defective screws relative to fastening operations, of which this experiment had 20. However, since the CVS controls only the top-left and top-right screwdrivers, only the 10 fastening operations on the upper side of the panel are counted.

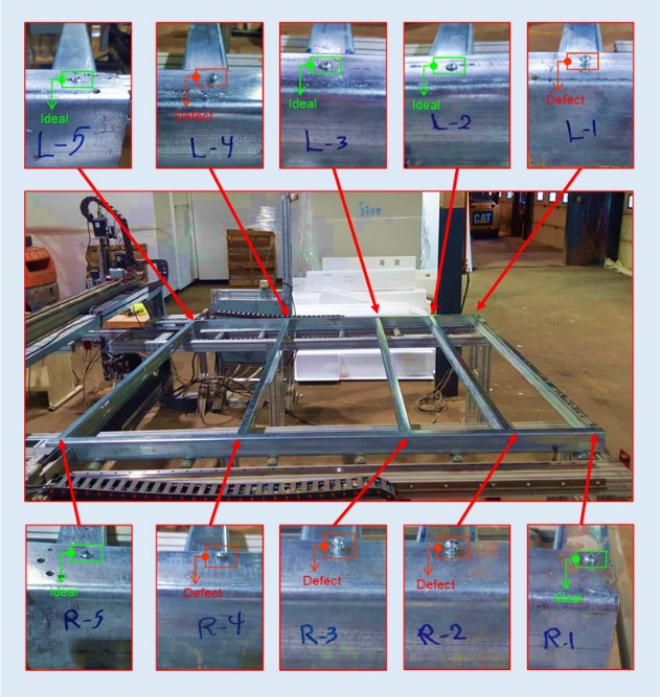
Table 3-3. Experiment results

| NO. Panel | Framework | LHS (Screw Status) | | | | | RHS (Screw Status) | | | | | Tot NO. Defects | Defects Rate % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | S1 | S2 | S3 | S4 | S5 | S1 | S2 | S3 | S4 | S5 | | |
| Panel 1 | Without CVS | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 20% |
| | With CVS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| Panel 2 | Without CVS | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 3 | 30% |
| | With CVS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| Panel 3 | Without CVS | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 20% |
| | With CVS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| Panel 4 | Without CVS | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 6 | 60% |
| | With CVS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| Panel 5 | Without CVS | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 6 | 60% |
| | With CVS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |

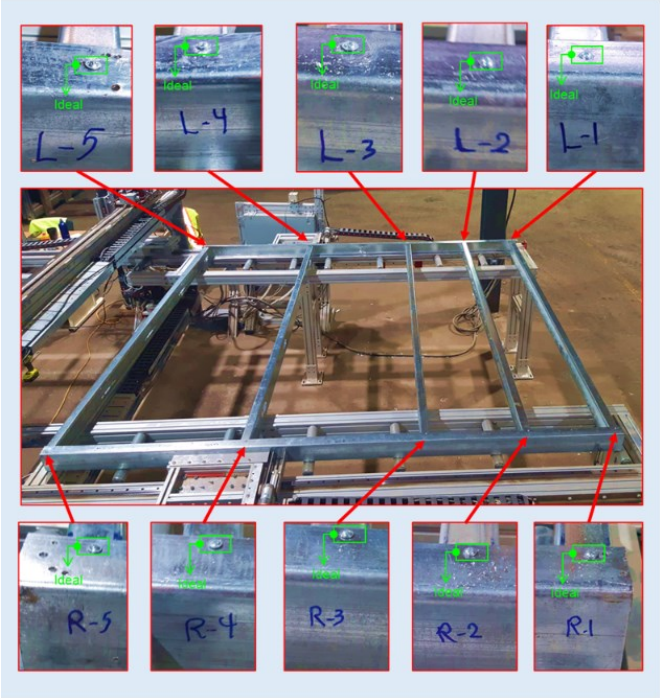
A summary of the results of the experiments depicted in Table 3-3 are listed bellow.

- Panel 1: Without CVS, there were 2 total defects (20% defects rate). With CVS, this was reduced to 0 defects.
- Panel 2: Without CVS, there were 3 total defects (30% defects rate). With CVS, this was reduced to 0 defects.
- Panel 3: Without CVS, there were 2 total defects (20% defects rate). With CVS, this was reduced to 0 defects.
- Panel 4: Without CVS, there were 6 total defects (60% defects rate). With CVS, this was reduced to 0 defects.
- Panel 5: Without CVS, there were 6 total defects (60% defects rate). With CVS, this was reduced to 0 defects.

The results show that the CVS works effectively in eliminating defects across the tested panels, resulting in a significant improvement in the quality control, as shown in Figure 3-43.




**Without CVS Framework**




**With CVS Framework**

Figure 3-43. Experimental results and comparison

# Chapter 4 . Impact of CVS Framework on the Machine's Performance

The effectiveness of steel framing machines undergoes a notable transformation with the integration of a computer vision system (CVS) framework. At the same time, various metrics, including the widely used Overall Equipment Effectiveness (OEE), serve as crucial indicators in manufacturing, production, and operations management for evaluating different process aspects and can effectively measure the impact of CVS over machine performance. This chapter explores an approach for real-time data collection and analysis from the machine using a Python code. The data and metrics are stored in a dedicated database within the MySQL Workbench, and the results are visually represented using the Grafana visualization tool. This allows for a visual evaluation of how the proposed CVS framework enhances efficiency and precision in the steel framing process through reliance on performance metrics such as OEE.

## 4.1 OEE metrics

OEE is a key performance indicator for both equipment and machinery, as well as a performance index of the production floor. This means that OEE can be calculated from the multiplication of the three main elements: Availability (A%), Performance (P%), and Quality (Q%), expressed as a percentage (Muchiri & Pintelon, 2008). Availability assesses the machine's uptime, performance gauges its operational speed, and quality evaluates the defect rate of the items produced. This integrated approach provides a comprehensive evaluation of a machine's efficiency.

The goal of calculating OEE is to increase its overall value, as well as identify areas for improvement. The percentage of OEE is calculated as follows:

$$\text{OEE } \% = A\% \times P\% \times Q\% \tag{4-1}$$

OEE can be affected by six significant losses, as depicted in Figure 4-1 and categorized as follows: (1) availability losses, i.e., breakdowns and changeovers, (2) performance losses, i.e., minor stops and speed reductions, and (3) quality losses, i.e., defects and start-up losses (Chikwendu et al., 2020)

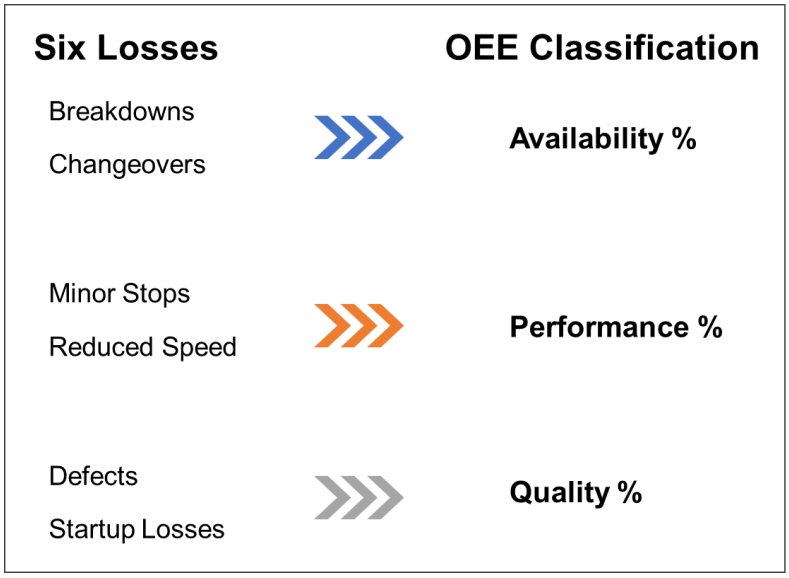



Figure 4-1. OEE measurement factors and corresponding six major losses

The availability in OEE measures the proportion of time that a machine or system is available to operate, compared to the total scheduled time in which a machine should be operating. Availability is calculated using the following formula:

$$A\% = \frac{\text{Operating Time}}{\text{Loading Time}} \times 100 = \frac{\text{Loading Time-Down Time}}{\text{Loading Time}} \times 100 \qquad 4\text{-}2$$

where:

- Operating Time: The amount of time the machine was operational and available for production, excluding any downtime.
- Loading Time: The total time that the machine was supposed to operate, including both operating and downtime. This duration is the period during which the equipment was expected to be available for production (Loading time is also referred to as scheduled time).
- Down Time: The total amount of time that the machine was not available for production due to reasons such as maintenance, repairs, setup, and adjustments.

The performance in OEE measures how well a machine or process performs compared to its maximum potential speed under optimal conditions. Performance is calculated using the following equation:

$$P\% = \frac{\text{Productive Time}}{\text{Operating Time}} \times 100 = \frac{\text{Ideal Cycle Time per unit} \times \text{Actual Output Units}}{\text{Operating Time}} \times 100 \qquad 4\text{-}3$$

where:

- Ideal Cycle Time: The shortest time in which one unit can be ideally produced under perfect conditions.
- Actual Output Units: The total number of units produced during the Operating Time.

The quality in OEE measures the proportion of right-first-time parts units produced out of the total units produced. This metric highlights the effectiveness of the production process in delivering products that meet quality standards.

$$Q\% = \frac{\text{Total Production Units - Defects Count Units}}{\text{Total Production Units}} \times 100 \qquad 4\text{-}4$$

where:

- Total Production Units: The total quantity of units produced, including both acceptable and defective units.
- Defects Count Units: The number of units produced that are not up to quality standards and are therefore defective or require rework.

In summary, availability, performance, and quality help improve the effectiveness of the machine. This is due to the ability to target the smallest calculated numbers in order to understand the reasons behind their lower values. However, the research goal is to minimize defects and reduce rework, and as such it has to do primarily with performance and quality aspects. Availability is not directly influenced, as losses are solely attributable to breakdowns and changeovers. Ultimately, these three elements have a significant effect on the OEE value.

To calculate the OEE values and gauge the performance of the CVS implementation, timers were set up in the machine's Programmable Logic Controller (PLC) to record the durations of different processing steps within the manufacturing sequence. The process flow, outlined in Figure 4-2, shows a sequential order of operations with time estimations for each step.

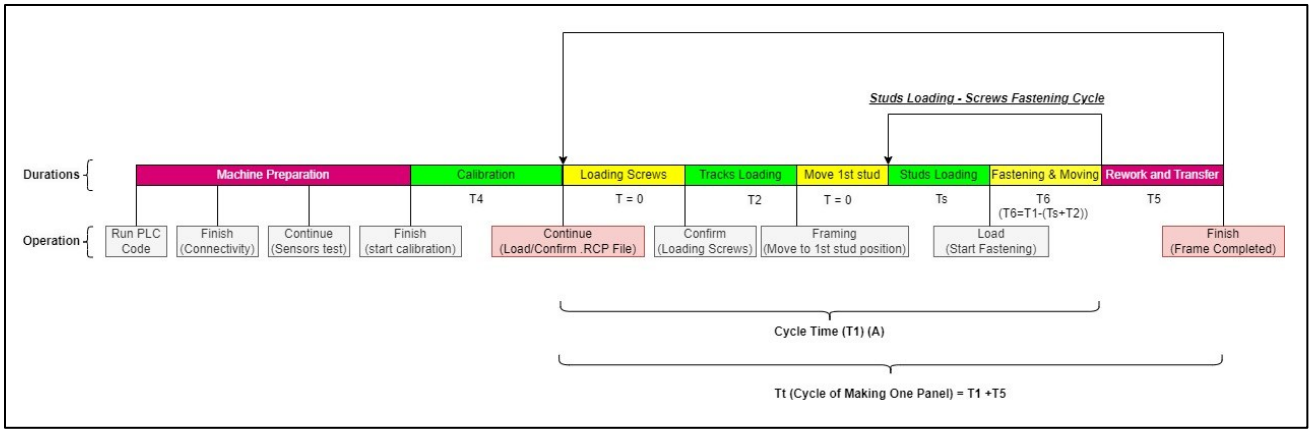



Figure 4-2. Operations breakdown and cycles times of LGS framing process

Here is a breakdown of the stages as per the flowchart:

- Machine Preparation: This phase involves two steps: running the PLC code, and testing sensors. Running the PLC code must be completed before proceeding to the next step. Completion is indicated by “Finish (Connectivity)”. Sensor testing must also be carried out before calibration starts. Machine setup and sensor checks are performed only once, after the machine starts up and all systems are functioning, causing their duration to be negligible with minimal impact.
- Calibration (T4): This phase includes a timer to record the duration needed for the machine calibration process. The recorded duration is labelled with a T4 and included in the PLC code, which indicates the time required for the calibration phase.
- Loading Screws: This stage accounts for the time to load screws into the screwdrivers. Since screws are pre-loaded in this scenario, it is associated a 0-s duration with no associated timer.
- Tracks Loading (T2): This stage uses a timer (T2) to measure the time for loading tracks. Ideally, this takes 15 s; any excess time is logged as waiting time on a separate timer ($T_{wt}$) in the PLC code.
- Move to First Stud Position: As the first stud is always positioned at the start of the panel, this step requires no movement or timer, and is considered to have a 0-s duration.
- Studs Loading (Ts): This step, ideally taking 10 s per stud, uses a timer (Ts) to record any additional time as machine waiting time ($T_{ws}$).

- Machine Running Time (Fastening & Moving) (T6): A comprehensive timer (T6) in the PLC code captures all operation times, including screw fastening and moving to subsequent stud positions. It is calculated using the following equation:

$$T6 = T1 - (Ts + T2) \quad 4\text{-}5$$

- Rework and Transfer (T5): This is the final phase of making the panel. It is equipped with a timer (T5) and measures the duration needed for repairing defect and transferring the panel to the next production stage.
- Cycle Time (T1): Defined as the accumulated time to produce the panels during the machine operating time, excluding rework and transfer time, this is recorded with a timer (T1). Thus, the average cycle time (t1) to produce one panel can be calculated as follows.

$$t1 = \frac{T1}{\text{Actual Outputs (units)}} \quad 4\text{-}6$$

- Total Cycle Time ($T_t$): This is the overall time to complete the panel during the machine operating time, including rework time. It is recorded with a timer ($T_t$) and calculated as follows:

$$T_t = T1 + T5 \quad 4\text{-}7$$

These timers help compute the OEE%, A%, P%, and Q% using the formulas below.

$$\text{Loading Time} = T1+T4+T5 \quad 4\text{-}8$$

$$\text{Operating Time} = T1+T5 \quad 4\text{-}9$$

$$\text{Down Time} = \text{Loading Time} - \text{Operating Time} = T4 \quad 4\text{-}10$$

Assuming no rework in ideal cycle time (T5 = 0), and that tracks and studs load within their ideal times (Ts<10sec, T2<15sec). Thus,

$$\text{Ideal cycle time per panel} = 10\times\text{No. Studs per panel} + 15 + \frac{T6}{\text{Actual Outputs (units)}} \quad 4\text{-}11$$

The number of studs per panel can be easily retrieved as the "entrycount" parameter in the RCP file, which, in this case, is 20. Since each stud has 4 fastening operations, then:

$$\text{No. Studs per panel} = \frac{\text{Total Fastening Operations}}{4} \quad 4\text{-}12$$

Additionally, two counters were integrated into the PLC's code. One counter tracks the number of screws labelled "defect" on the left-hand side (LHS) of the machine, while the other counter tracks the number of screws labelled "defect" on the right-hand side (RHS). These counters were set into the PLC's code, gathering information from the CVS about any detected defects. Thus:

$$\text{Total Defects} = \text{LHS\_Defects\_QTY} + \text{RHS\_Defects\_QTY} \quad 4\text{-}13$$

The total number of fastening operations is determined by the number of studs per panel. This information can be directly obtained from the RCP file under the label "entrycount", as illustrated in Figure 3-37. It can also be extracted using PLC code. For this experiment, with only 5 studs, there are 20 fastening operations.

Based on Equations 4-8, 4-9, 4-10, 4-11, 4-12, and 4-13, Equations 4-2, 4-3, and 4-4 can be modified to produce the following equations.

$$A\% = \frac{T1+T5}{T1+T4+T5} \times 100 \quad 4\text{-}14$$

$$P\% = \frac{\text{Ideal cycle time per panel} \times \text{Actual Output Units}}{T1+T5} \times 100 \quad 4\text{-}15$$

$$Q\% = \frac{\text{Total Fastening Operations} - \text{Total Defects}}{\text{Total Fastening Operations}} \times 100 \quad 4\text{-}16$$

Using the values derived from Equations 4-14, 4-15, and 4-16, the OEE% of the machine is calculated. This allows for the evaluation of the machine's performance before and after the implementation of the proposed CVS framework.

The calculated values are stored in a MySQL database. Moreover, the same equations are used to create the queries for the Grafana dashboard visuals, as discussed in later sections of this chapter.

## 4.2 Collection of metrics and visualization of results

The assessment of various performance metrics related to the impact of the proposed CVS on the light-gauge steel (LGS) framing process has underscored the critical necessity for integrating a visualization approach within a comprehensive data-driven decision-making (DDDM) framework.

This integrated system relies on four fundamental pillars: the machine's PLC, a database management system (DBMS), an online visualization tool, and a form of communication software. The PLC-MODICON M251 serves as the central hub for all necessary operations and durations required for calculating the performance metrics. MySQL is used as the DBMS to securely store all relevant data. For visualization, Grafana is chosen to display the metrics pulled from the database. To ensure smooth communication between the PLC, MySQL, and Grafana, a Python code has been developed to manage the data flow among these components. This comprehensive framework, illustrated in Figure 4-3, facilitates efficient data collection, storage, visualization, and communication. This integration enhances informed decision-making within the LGS framing process.

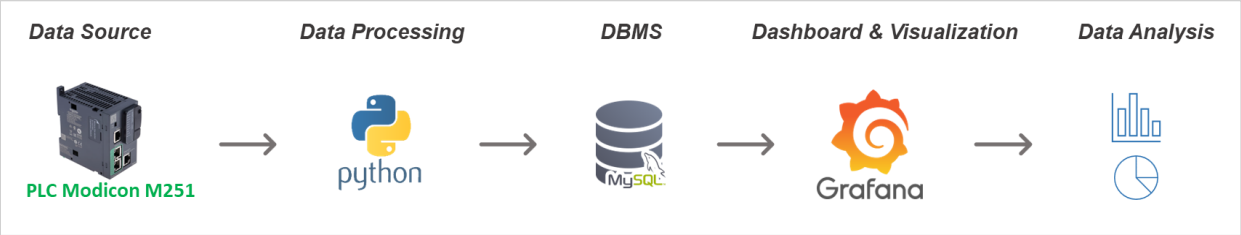



Figure 4-3. Framework for data logging, visualization of metrics, and analysis

### 4.2.1 Creation of MySQL database

DBMSs are essential software tools for managing databases and ensuring data integrity and security. There are various types of systems, including relational DBMSs (such as MySQL and PostgreSQL) and non-relational or NoSQL DBMSs (such as MongoDB and Cassandra). Among these, MySQL is favoured for storing data from the PLCs of the light-gauge steel framing machine (LGSFM). Because it is open-source, MySQL offers cost advantages by eliminating licensing fees. The database captures machine state data, which is then visualized through Grafana.

To establish the database, MySQL Workbench was installed on the computer, selecting windows-based version 8.0.36.

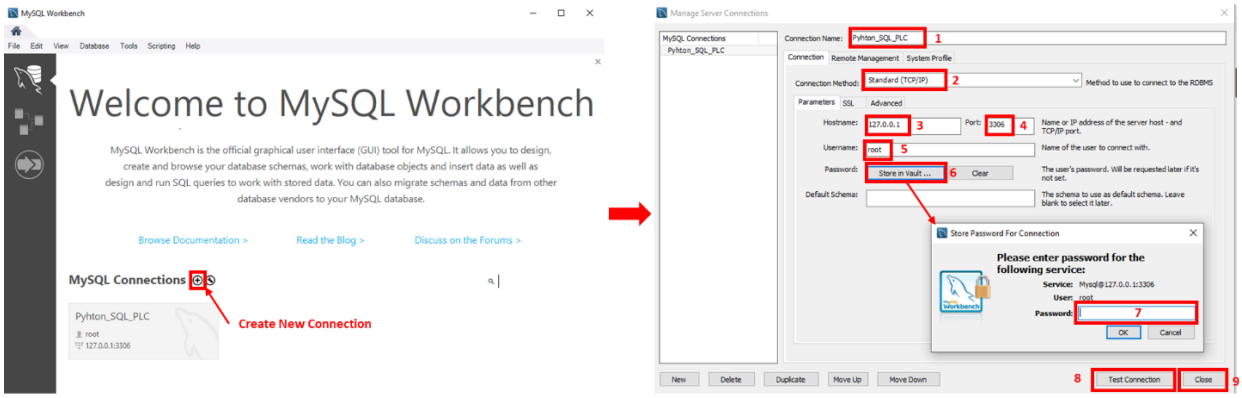



Figure 4-4. MySQL new connection configuration

As depicted in Figure 4-4, initiating a new connection begins with selecting the (+) icon, which prompts the "Manage Server Connections" window to appear. Here, the connection settings are configured. In step 1, a name is assigned to the connection. Step 2 involves specifying the Connection Method, which, in this case, is TCP/IP, since communication with the PLC occurs via its Ethernet port. Steps 3, 4, and 5 entail setting the "hostname", "port number", and "username", respectively. Step 6 involves setting a password for the connection by clicking the "Store in Vault …" button to input a desired password in step 7. Step 8 involves testing the connection by clicking the "Test Connection" button. The test having been passed, as shown in Figure 4-5, the "Manage Server Connections" window is closed, and the process of creating the database begins.

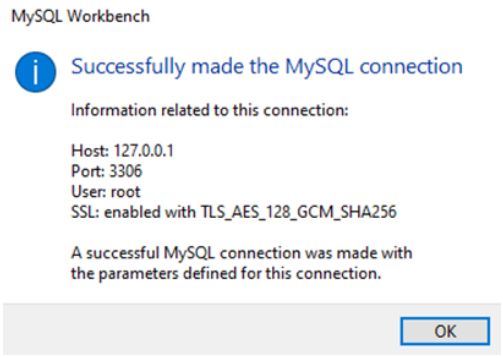



Figure 4-5. Message for a successful MySQL connection

The parameters configured during this step later serve as the login credentials when establishing communication between Python and the MySQL database.

Once logged in, the interface appears as depicted in Figure 4-6. Here, the first element encountered is the query command window, where query commands are inputted to perform various actions,

such as creating databases, tables within databases, or columns within tables. Additionally, the query execution button appears. This button allows the queries inputted to the command window to be executed, among other essential functions.

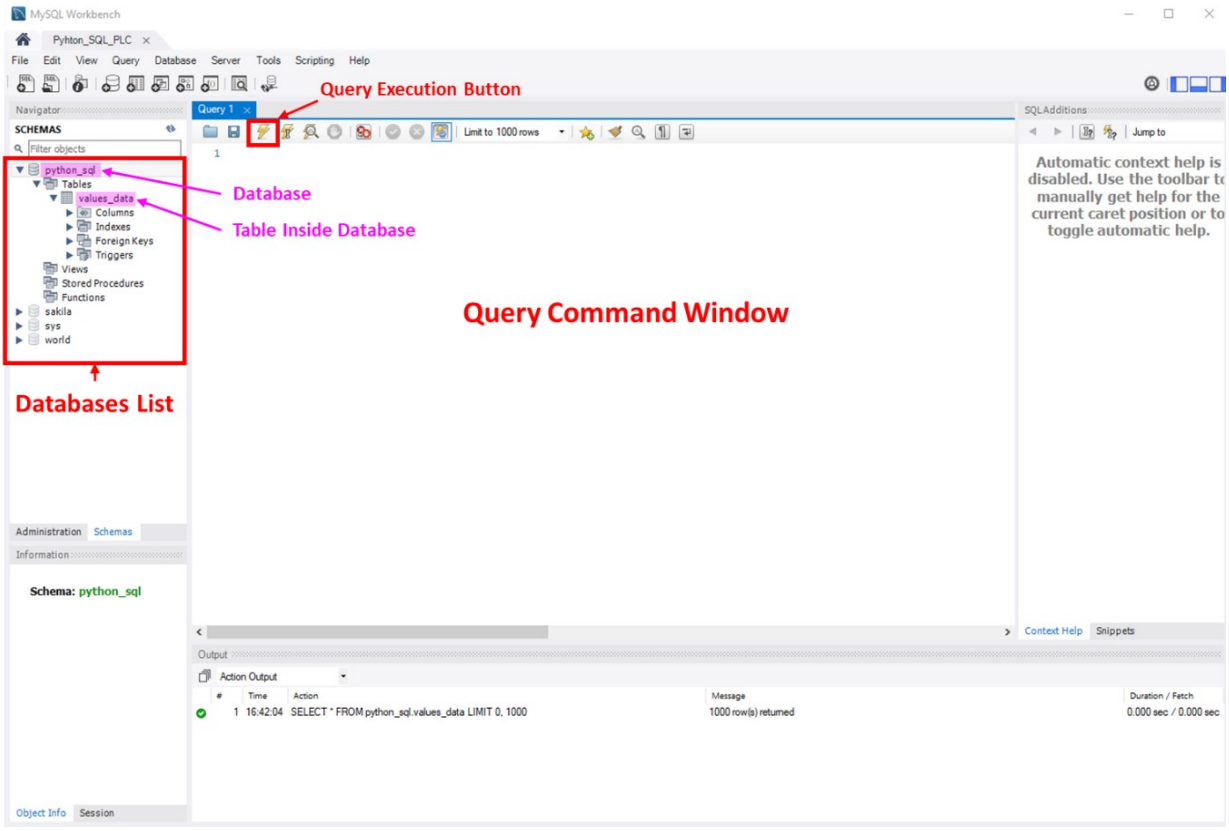



Figure 4-6. MySQL workbench interface

Numerous tables are established within the database to store different types of data, including,

- LHS defect quantity (lhs defects_qty)
- RHS defect quantity (rhs_defects_qty)
- Cycle time (t1_cycle_time)
- Tracks loading time (t2_tracks_loading_time)
- Calibration Time (t4_calibration_time)
- Machine running time (t6_machine_running_time)
- Studs loading time (ts_studs_loading_time)
- Studs waiting timed (tws_studs_waiting_time)
- Tracks waiting time (twt_tracks_waiting_time)

And each table of these tables has the following columns:

- ID: This is the primary key for the table, with unique values to identify each record.
- Panel1 to Panel10: These columns represent the record related to ten different panels.
- CurrentTimeStamp: This column stores the date and time when the record was created or last updated.

Figure 4-7 displays an example SQL query for creating the table “lhs_defects_qty” in the MySQL database “python_sql”. This table includes columns from “Panel1” to “Panel10” as floats, “ID” as an auto-incrementing primary key, and “CurrentTimeStamp” as a timestamp column that initially sets to the current timestamp and updates with each record change.

```
CREATE DATABASE IF NOT EXISTS python_sql;
USE python_sql;

CREATE TABLE lhs_defects_qty (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Panel1 FLOAT,
    Panel2 FLOAT,
    Panel3 FLOAT,
    Panel4 FLOAT,
    Panel5 FLOAT,
    Panel6 FLOAT,
    Panel7 FLOAT,
    Panel8 FLOAT,
    Panel9 FLOAT,
    Panel10 FLOAT,
    CurrentTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Figure 4-7. SQL Query for creating a table with timestamp in MySQL database

To create the table, this query must be run in the query command window of the MySQL Workbench environment. To set up rest of the tables, replace the table’s name “lhs_defects_qty” with the names of the other tables.

Figure 4-8 shows the final database with the created tables to collect the measurements form the machine's PLC.

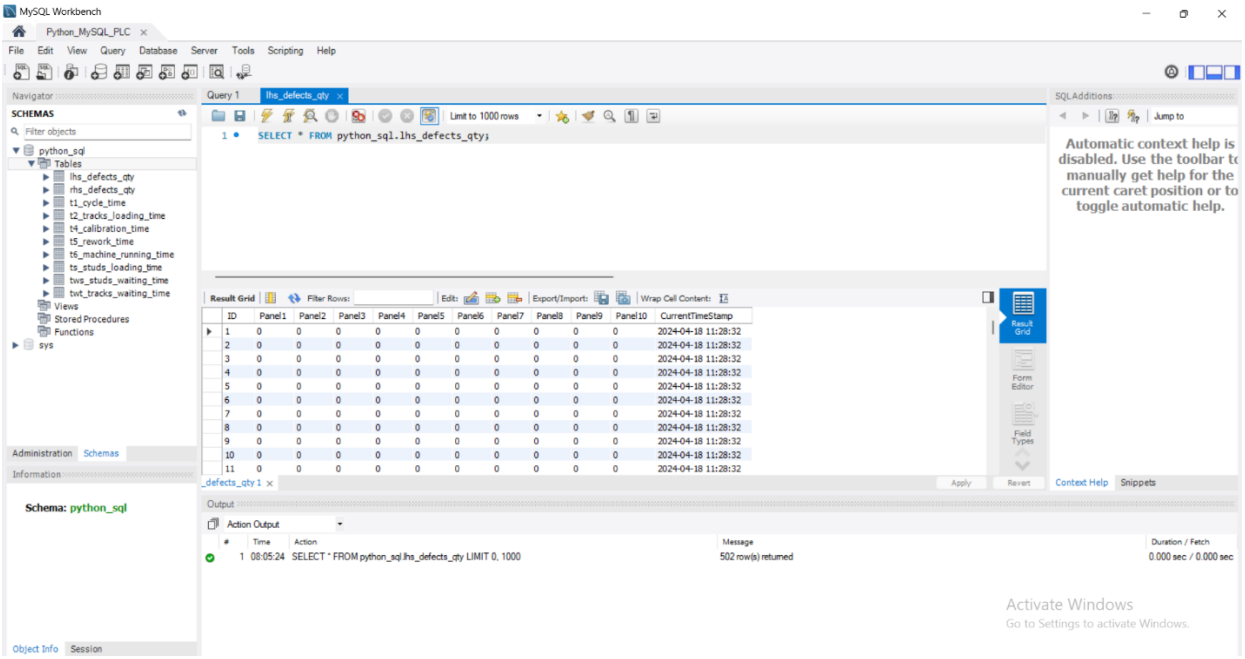



Figure 4-8. Overview of database containing machine metrics

### 4.2.2 PLC–MySQL database communication for real-time data logging

The communication between the PLC and a MySQL database was established using the Python code depicted in Figure 4-9 as a typical application of integrating PLC data with a relational DBMS for data logging.

The code begins by establishing a connection over Modbus TCP/IP protocol, which is a widely adopted industrial communication protocol, leveraging the specified host IP address and port number. Upon successful connection with the PLC, the code attempts to connect to a MySQL database using the provided credentials of the host name, the username of the database, the password, and the name of the database. In this case, the database is named "python-sql".

```
Procedure Read_PLC_and_Store_to_Database
    Initialize Modbus server and database settings
    Establish a connection to the Modbus server
    If connection successful
        Print "Connected to PLC"
        Establish a connection to the MySQL database
        Repeat indefinitely:
            For each mapping of PLC register addresses to database tables
                Read 10 registers from the current PLC address
                If read successful
                    Format the read values for SQL insertion
                    Insert values into the corresponding database table
                    Print success message with details
                Else
                    Print error message for read failure
            Sleep for 1 second
        End Repeat
    Else
        Print "Failed to connect to PLC"
    Handle interruptions and errors
    Close connections and clean up resources
End Procedure
```

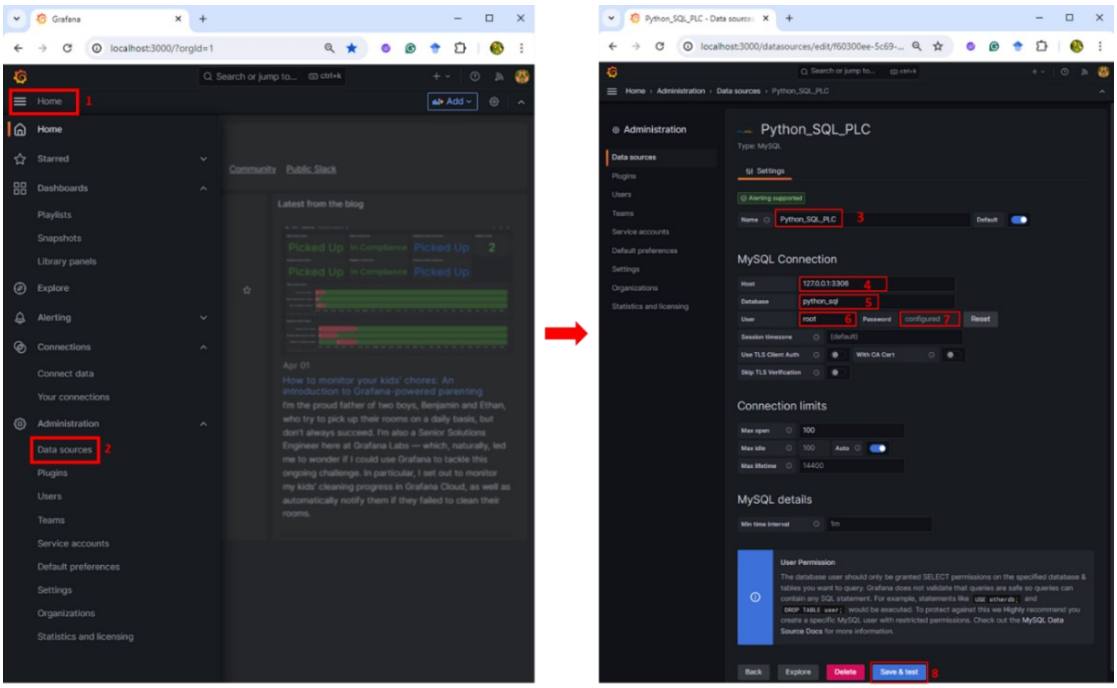
Figure 4-9. Pseudocode for storing PLC data into MySQL database

Once both connections are established, the code enters a real-time continuous read of a pre-defined set of holding registers from the PLC. These registers typically contain variables and relevant performance evaluation metrics of the manufacturing process. The values retrieved from these registers are then timestamped and inserted into a designated table within the “python-sql” database, allowing for real-time data logging and subsequent analysis. The process repeats at 1-second intervals, ensuring ongoing data acquisition. This approach exemplifies a basic yet powerful integration between industrial control systems and information technology systems, enabling enhanced DDDM processes.

### 4.2.3 Creating a Grafana visualization dashboard

Grafana, an open-source platform for data visualization, enables the creation of visually appealing dashboards showcasing various aspects of machine data. This platform functions by using queries from the MySQL database to retrieve information about the machine in real time. The query results are made more user-friendly by improving their readability. The Grafana-enterprise-10.4.1 (a windows-based version that is running as the "Grafana" service is used in this scenario because it provides a customisable dashboard layout that is simple to edit and read. Grafana Enterprise version 10.4.1 having been installed to the system, it can be accessed via "localhost:3000" using the default login credentials. (These credentials can be updated later for security purposes. Once logged in, the dashboard can be crafted according to specific needs and preferences.)

Prior to initiating the dashboard creation process, it is essential to set up the data sources. This is done by navigating to the "Home" menu and selecting the "Data Sources" option, as illustrated in Figure 4-10-a. This leads to the data sources configuration page, depicted in Figure 4-10-b, where data sources are defined and configured accordingly.




(a) (b)

Figure 4-10. Configuring communication between Grafana and MySQL database

First, any name for the connection must be chosen. In this case, the same name that was previously used for the "Connection Name" in MySQL, "pyhton_sql_plc", is selected. Next, the "Host" name

is inputted as “127.0.0.1:3306”, where “127.0.0.1” signifies the host chosen in MySQL, followed by the port number “3306”. For the “Database” field, the name of the database created in MySQL, “python_sql” is entered. The username specified on this page is “root”, consistent with the username chosen in MySQL Workbench. This is followed by the corresponding password for the MySQL connection, ensuring Grafana’s access to the database. Finally, the “Save & test” button is clicked to verify that the connection between Grafana and the “python_sql” database was successful. Successful establishment of communication is confirmed upon receiving the notification “Database Connection OK”, as depicted in Figure 4-11 This notification signifies that Grafana now has the capability to display all relevant data from the database.

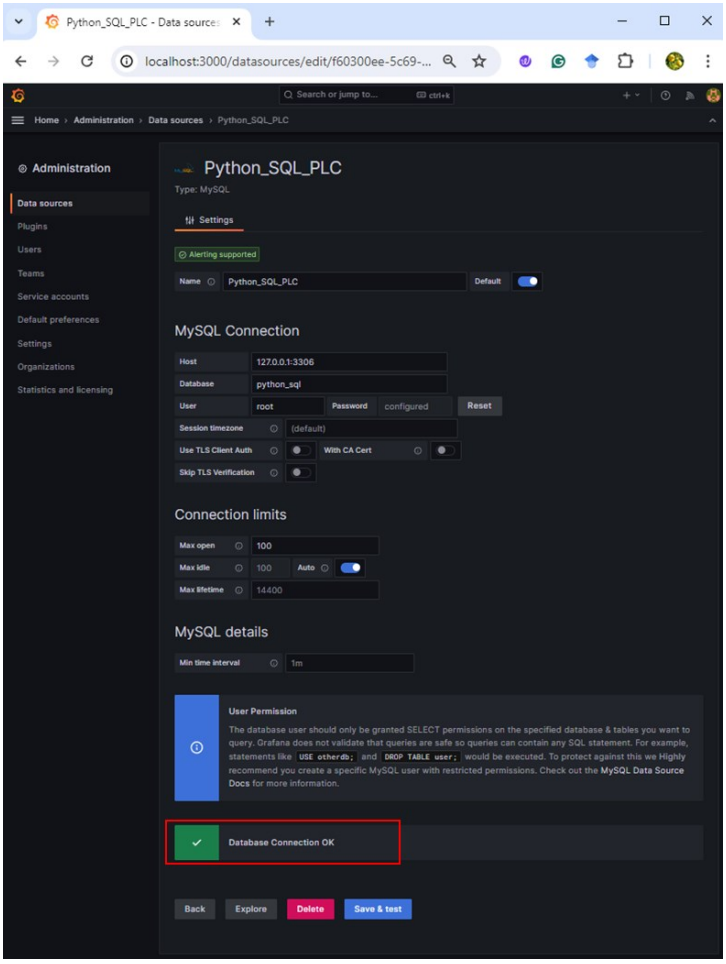



Figure 4-11. Grafana-MySQL successful connection notification

Once the data sources are configured, the creation of the visualization dashboard begins (see Figure 4-12). The creation process begins with navigating to the “Dashboards” menu, selecting “New”,

and establishing a “New Folder”. This folder is where the dashboard can be saved and organized in accordance with the project’s requirements. Subsequently, selecting “New Dashboard” initiates the crafting process, enabling visualization of the desired performance metrics.

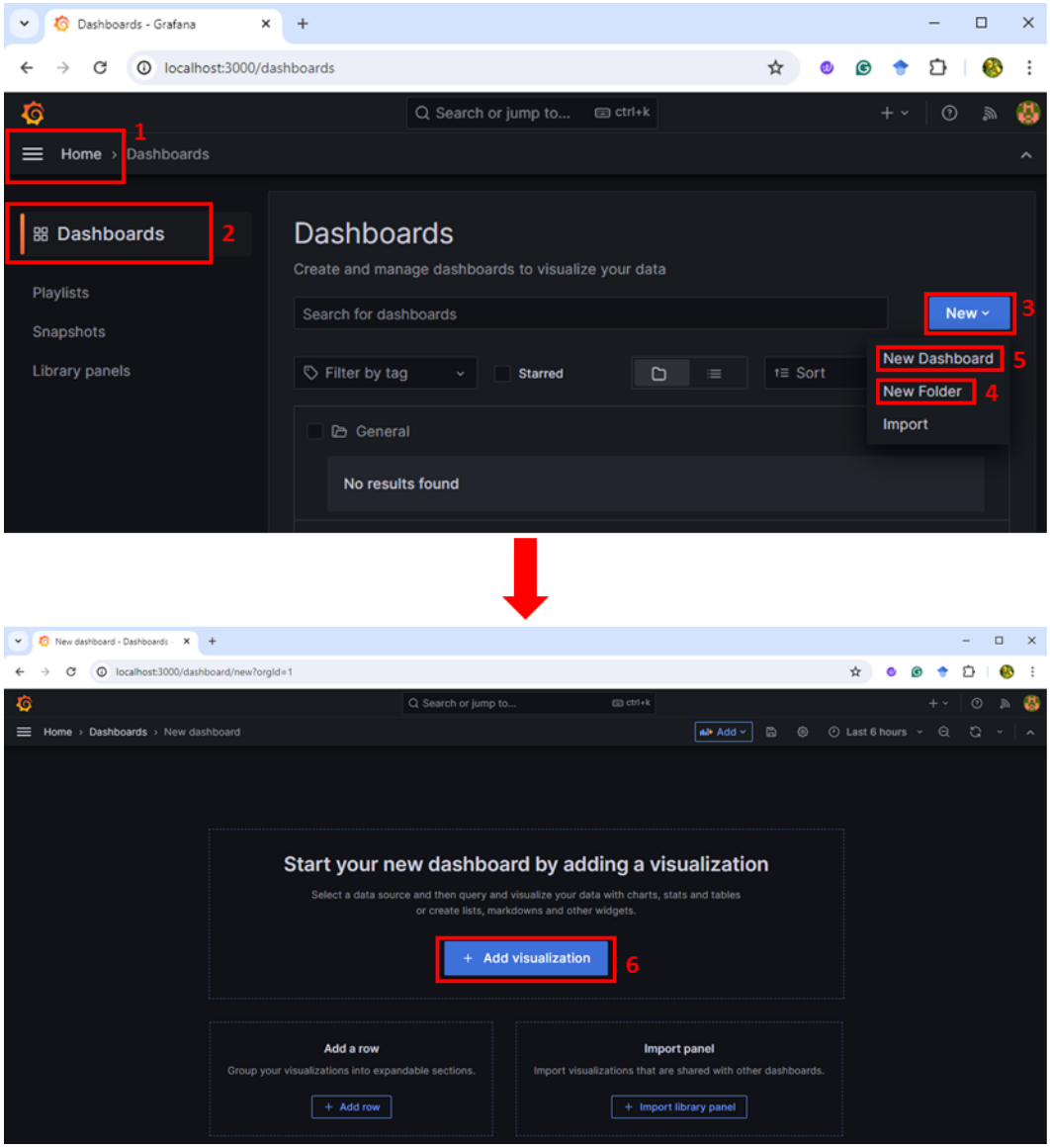



Figure 4-12. Grafana project folder configuration

Once “Add visualization” is selected, the panel editor page appears, as depicted in Figure 4-13. This page allows for flexibility in choosing the preferred visualization tools and type SQL queries to extract relevant information from the “python-sql” database. After configuring the visualization to the requirements, selecting “Apply” and “Save” preserves the work for future reference. Unwanted changes can be disregarded by selecting “Discard”.

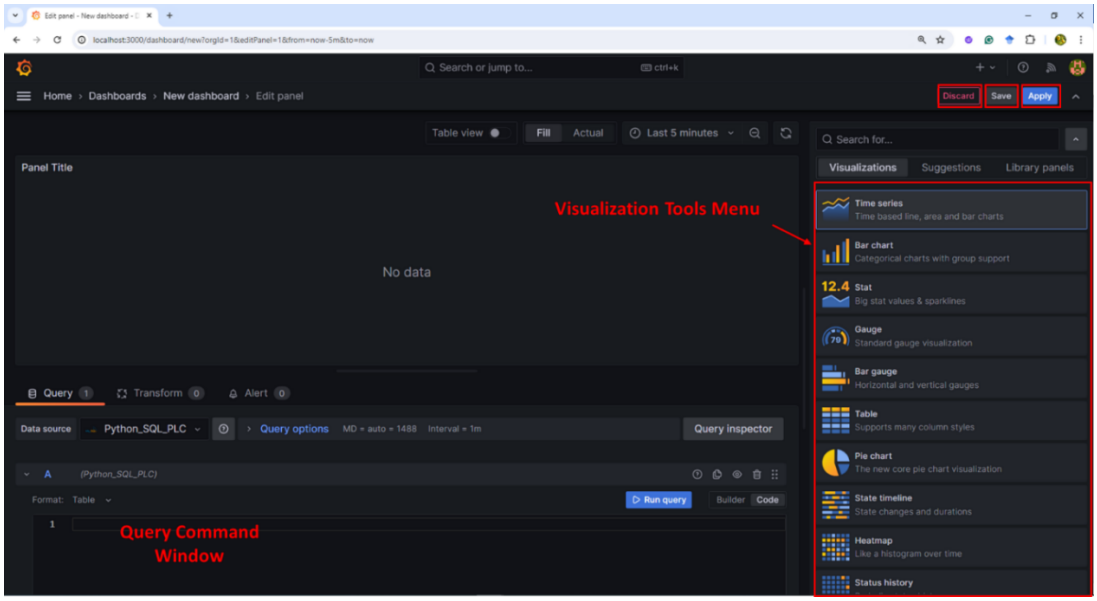



Figure 4-13. Grafana panel editor

To set up the panel displaying the “Cycle Time–T1” duration, which represents the total time needed to produce all outputs, a panel with a “Stat” type was chosen. The query depicted in Figure 4-14 was then entered and run in the query command window of the “Stat” panel. The panel visualized the results of T1 based on a specified timestamp.

```
SELECT
    (SELECT Panel1 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

Figure 4-14. SQL query to visualize T1 in Grafana

Following these steps, the dashboard is built by inputting designated queries into each specific panel. The rest of the queries are listed in the appendix.

Figure 4-15 presents the final Grafana dashboard, which measures the OEE metrics for the LGSFM. The dashboard features gauges for A%, P%, and Q%, as well as the OEE% value.

Alongside the primary OEE metrics, this dashboard has been designed to capture fundamental operational durations of the LGSFM. This includes tracking cycle times, wait times, the quantity of defects, and other data that was described earlier in Figure 4-2.

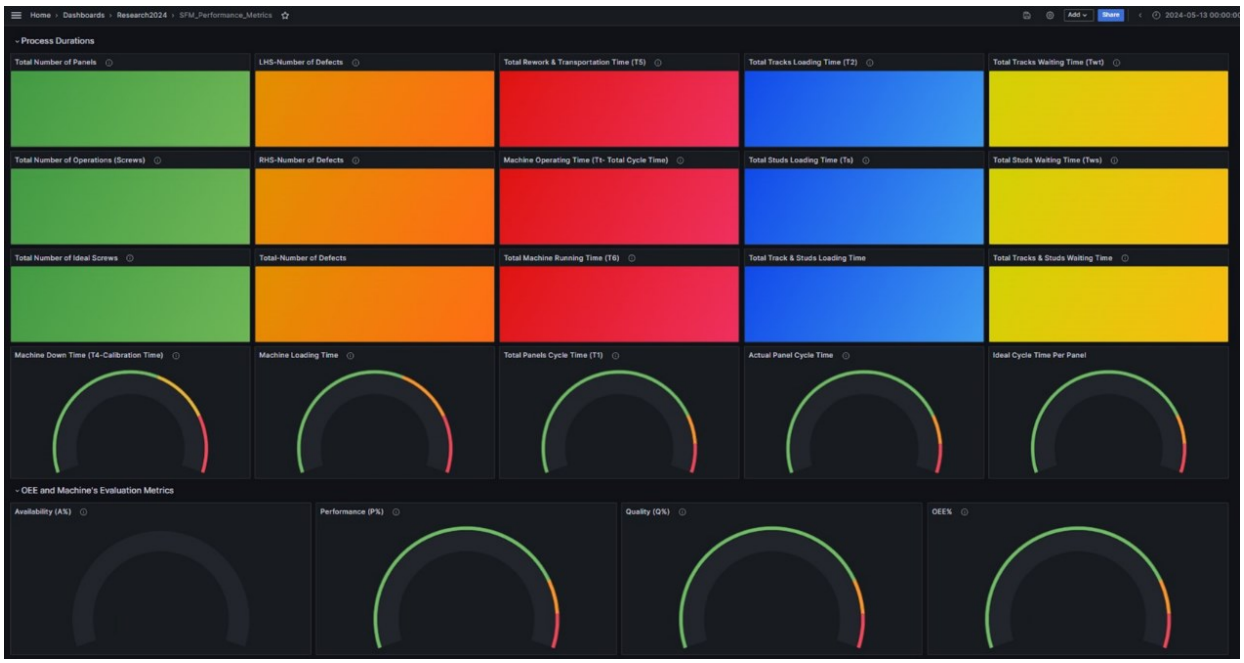



Figure 4-15. Grafana dashboard for OEE% evaluation metrics

This dashboard serves as a crucial tool for real-time monitoring and strategic decision-making in LGSFM and its associated processes. The dashboard uses various, colour-coded visual elements, such as stat panels and gauges, to clearly display these metrics for easy comparison and demonstration.

## 4.3 Metrics analysis and discussion

Chapter 3 discussed an experiment where five panels were constructed. Table 3-3 showed the results of the experiment before and after integrating the CVS with the existing control system of the LGSFM. These results were stored in the "python_sql" database using the "PLC-Python-MySQL-Grafana" framework, as illustrated earlier in Figure 4-3. The results of the experiment from Chapter 3 can also be displayed on the dashboard, visually showcasing the outcomes of these two distinct experiments. Figure 4-16 illustrates the LGSFM's metrics and measurements before the integration of the CVS framework, while Figure 4-17 depicts the metrics after the integration with the CVS framework.


Figure 4-16. Visualization of final results in Grafana without CVS framework integration


Figure 4-17. Visualization of final results in Grafana with CVS framework integration

A comparison between Figure 4-16 and Figure 4-17 shows that the integration of the CVS framework enhanced the machine metrics and process measurements. This improvement is also evident by the complete elimination of defects. To offer a more detailed analysis of these

improvements, the metrics have been categorized into three groups: process duration metrics, process counters, and machine evaluation metrics. These categories provide deeper insights into the impact of the CVS on the LGSFM.

### 4.3.1 Process durations

Process durations are time metrics that measure time periods related to the machine's operations. They are summarized in Table 4-1. This table compares various process durations before and after the integration of CVS, listing the metrics that measure a different aspect of the machine's operational process. These metrics include total machine running time, machine operating time, rework and transfer time, loading time for tracks and studs, waiting times, machine down time for calibration, loading time for the machine, and cycle times for panels manufacturing.

Table 4-1. Comparison of process durations before and after CVS integration

| **Metric** | **Timer(s)** | **Before CVS Integration (minutes)** | **After CVS Integration (minutes)** |
|---|---|---|---|
| Total Machine Running Time | T6 | 5.50 | 6.05 |
| Machine Operating Time | Tt = T1+T5 | 23.70 | 18.70 |
| Total Rework & Transfer Time | T5 | 5.55 | 2.02 |
| Total Tracks Loading Time | T2 | 2.00 | 1.07 |
| Total Studs Loading Time | Ts | 10.70 | 9.57 |
| Total Tracks & Studs Loading Time | T2+Ts | 12.70 | 10.60 |
| Total Tracks Waiting Time | $T_{wt}$ | 0.75 | 0.02 |
| Total Studs Waiting Time | $T_{ws}$ | 8.02 | 6.90 |
| Total Tracks & Studs Waiting Time | $T_{wt} + T_{ws}$ | 8.77 | 6.92 |
| Machine Down Time (Calibration Time) | T4 | 0.56 | 0.56 |
| Machine Loading Time | T1+T4+T5 | 24.20 | 19.30 |
| Total Panels Cycle Time | T1 | 18.10 | 16.70 |
| Actual Panel Cycle Time | $\frac{T1+T5}{\text{No. Panels}}$ | 3.62 | 3.34 |
| Ideal Cycle Time Per Panel | $\frac{T6}{\text{No. Panels}} + \text{No. Studs} \times 10 + 15$ | 2.20 | 2.30 |

The bar chart shown in Figure 4-18 provides a clear visual representation of the changes in process durations listed in Table 4-1. The blue and green bars represent the durations before and after CVS integration respectively, the horizontal axis represents the time in minutes, and each bar's length illustrates the duration for each process labelled with duration in minutes of each process.

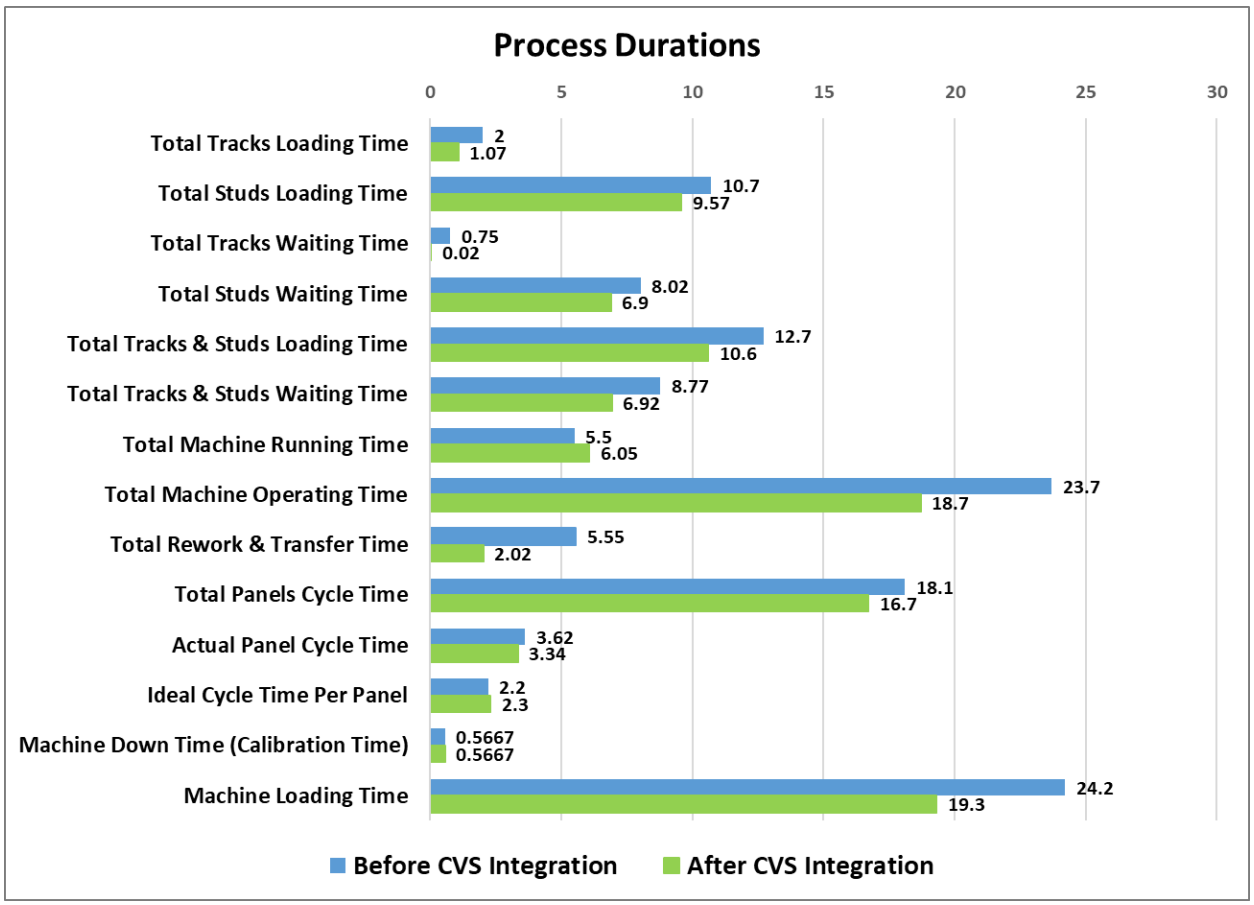



Figure 4-18. Process durations comparison in minutes after and before CVS integration

### 4.3.2 Process counters

The Table 4-2 summarizes metrics such as the total number of manufactured panels, the overall count of fastening operations, and the numbers of defects and ideal screws, showcasing a significant reduction in defects for both the LHS and RHS. Overall, the integration of the CVS result in defects being completely eliminated, decreasing from 19 to 0. Meanwhile, the number of ideal screws increased from 31 to 50, demonstrating the enhanced precision by the CVS integration.

Table 4-2. Comparison of process counters before and after CVS integration

| Metric | Before CVS Integration | After CVS Integration |
|---|---|---|
| Total Number of Panels | 5 | 5 |
| Total Number of Operations (Screws) | 50 | 50 |
| LHS-Number of Defects | 7 | 0 |
| RHS-Number of Defects | 12 | 0 |
| Total Number of Defects | 19 | 0 |
| Total Number of Ideal Screws | 31 | 50 |

Figure 4-19 illustrates Table 4-2 as a bar graph. Process counters from before the CVS integration are in blue, while process counters from after the CVS integration are in green. The graph visually shows the significant decrease in defects due to the integration of the CVS.

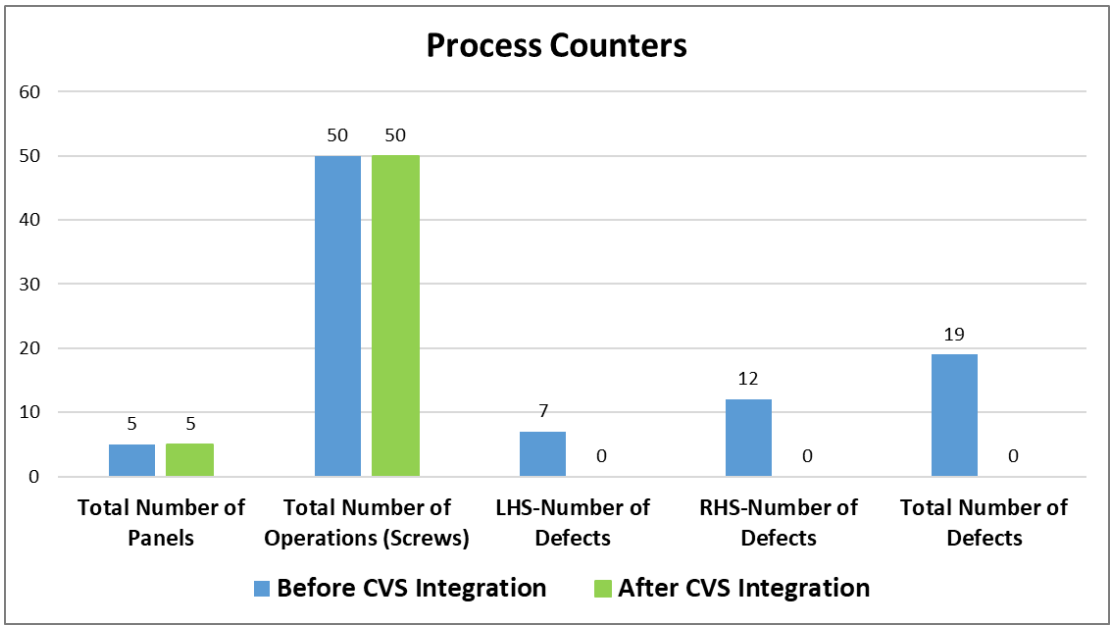



Figure 4-19. Comparison of process counters before and after CVS integration

### 4.3.3 Machine evaluation metrics

The Table 4-3 shows how the integration of the CVS affected the operational performance indicators (A%, P%, Q%, and OEE%). After the CVS integration, A% slightly decreased from 97.7% to 97.1%. On the other hand, the P% significantly improved from 46% to 61.3%. Similarly,

Q% increased from 62% to 100%. And finally, OEE% more than doubled from 27.9% to 59.5%. These changes highlight substantial enhancements on the machine's performance metrics made by the integration of the CVS system with the machine.

Table 4-3. Comparison of evaluation metrics before and after CVS integration

| Metric | Before CVS Integration | After CVS Integration |
|---|---|---|
| A% | 97.7% | 97.1% |
| P% | 46.0% | 61.3% |
| Q% | 62.0% | 100.0% |
| OEE% | 27.9% | 59.5% |

Figure 4-20 illustrates the information in Table 4-3 in a bar graph, visually comparing the operational performance metrics described before and after the integration of the CVS system. Blue bars represent the metric values prior to the CVS integration, while green bars represent the metric values after the CVS integration, providing observations of the resulting enhancements over the LGSFM performance metrics.

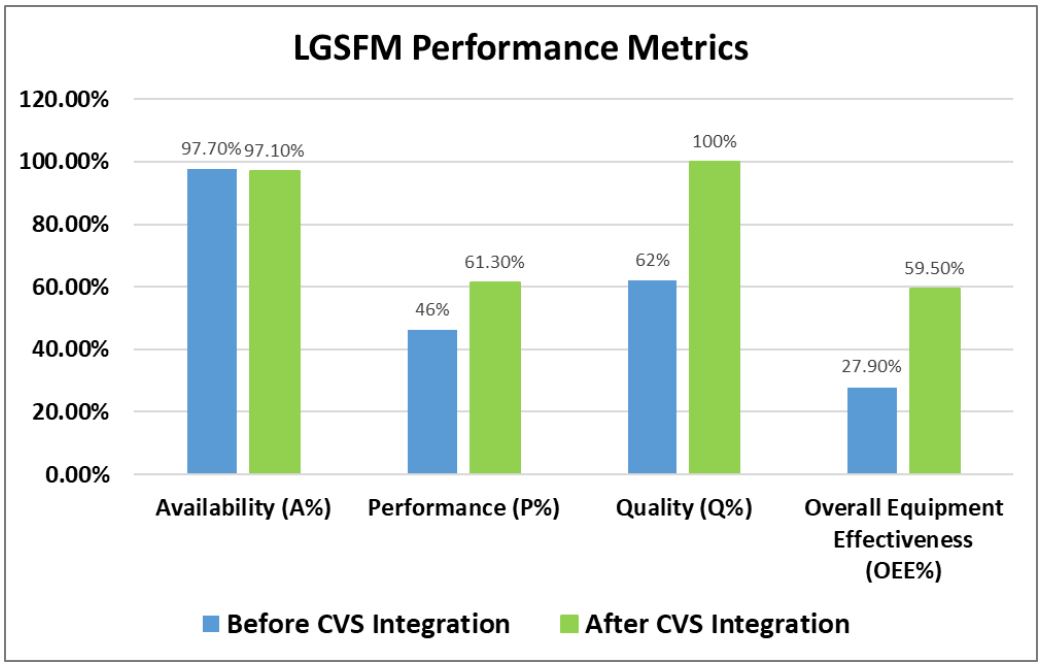



Figure 4-20. Comparison of evaluation metrics before and after CVS integration

### 4.3.4 Results and discussion

The findings in Table 4-1 demonstrate that the CVS framework has significantly optimized the LGS framing process, particularly within the “Rework and Transfer” operational step. The “Rework” portion of this phase, which includes repairing defects before the process can continue, saw a significant time reduction from 5.55 min to 2.02 min, eliminating 3.53 min of repair time. This reduction accounts for nearly two-thirds of the previous duration.

Further insights from Table 4-1 show that the CVS also had an effect on the “Total Panel Cycle Time”, which represents the total time required to produce five panels. It also affected the “Actual Panel Cycle Time” which is the average time needed to construct one panel. Additionally, the “Machine Loading Time” was reduced post CVS integration as well. This represents a substantial improvement in productivity, resulting in faster panel production.

Moreover, Table 4-2 shows the count of defects before and after CVS integration. The CVS’s effectiveness appears through defects elimination during the experiments. Prior to CVS integration, 19 out of 50 fastening operations were defects. Post-CVS integration, no defects were noted, achieving an ideal score of 50 out of 50 perfect fastenings. This flawless execution has increased the quality 62% to 100%, as indicated in Table 4-3, marking a substantial enhancement in panel quality.

Table 4-3 also highlights how integration influenced the machine’s availability and performance metrics. Specifically, while the reduction in T5 slightly modified the availability (decreasing from 97.7% before CVS implementation to 97.1% afterwards), it significantly boosted the performance metric. Performance increased from 46% before CVS integration to 61.3% after integration.

Consequently, and as depicted in TableTable *4-3*, these improvements in P%, and Q% metrics with the slight reduction in A% collectively led to a substantial rise in the OEE%, which soared from 27.9% to 59.5%. This drastic increase highlights the influence of the CVS on the LGSFM’s overall performance.

Additionally, the decreased rework time and the significant reduction in manufacturing defects can be expected to significantly enhance labour productivity and efficiency, leading to higher output rates. This, in turn, will reduce labour costs per unit, thus increasing profit margins.

## 4.4 Sensitivity analysis

As previously discussed, the primary influence of the CVS is observed on the "Rework and Transfer Time", referred to as T5. Table 4-4 categorizes which durations are affected by the CVS and those are not.

Table 4-4. Impact of the CVS over the machine metrics

| Metric | Timer(s) | CVS Direct Impact |
|---|---|---|
| Total Machine Running Time | T6 | No |
| Machine Operating Time | Tt = T1+T5 | Yes |
| Total Rework & Transfer Time | T5 | Yes |
| Total Tracks Loading Time | T2 | No |
| Total Studs Loading Time | Ts | No |
| Total Tracks & Studs Loading Time | T2+Ts | No |
| Total Tracks Waiting Time | $T_{wt}$ | No |
| Total Studs Waiting Time | $T_{ws}$ | No |
| Total Tracks & Studs Waiting Time | $T_{wt} + T_{ws}$ | No |
| Machine Down Time (Calibration Time) | T4 | No |
| Machine Loading Time | T1+T4+T5 | Yes |
| Total Panels Cycle Time | T1 | No |
| Actual Panel Cycle Time | $\frac{T1+T5}{NO.\ Panels}$ | Yes |
| Ideal Cycle Time Per Panel | $\frac{T6}{NO.\ Panels} + NO.\ Studs \times 10+15$ | No |

Additionally, the number of defects is another key metric directly influenced by the CVS. To prove the impact of the CVS, a sensitivity analysis on each A%, P%, and Q% was conducted.

### 4.4.1 A%

Based on Equation 4-14 for A%, a sensitivity analysis was conducted to assess the impact of T5 on A% before and after integrating CVS with the LGSFM. Here are the details of the analysis:

- T5: Considered as the main metric where the direct impact of CVS is primarily observed on T5. Initially, T5 ranged from 5.55 minutes (1.11 minutes per single panel) without CVS to 2.02 minutes (0.4 minutes per single panel) with CVS. For this study, the range was extended to consider "Transfer and Rework time" between 0.4 and 2 minutes for a single panel to cover broader scenarios.

- T4: This duration maintained at a constant value of 0.56 minutes (34 seconds), which represents the calibration time in both scenarios as shown in the experiments.
- T1: The values of this metric influenced by operator and machine performance and unaffected by CVS. In the experiments, T1 ranged from 11.5 minutes (2.3 minutes per panel) to 18.1 minutes (3.62 minutes per panel) without the CVS, and with CVS, it ranged from 11.5 minutes (2.3 minutes per panel) to 16.7 minutes (3.34 minutes per panel). For this study, the analysis was expanded to include T1 values from 2 to 4 minutes per single panel, increasing in 0.5-minute increments, resulting in five distinct curves for the analysis.

This structured approach summarized in Table 4-5 allows us to examine the impact of T5 on A% under varying conditions, providing a comprehensive understanding of the dynamics with and without CVS implementation.

Table 4-5. Study ranges of machine metrics for A% sensitivity analysis

| **Metric** | **Experiment duration range (minutes per panel)** | | **Study duration range (minutes)** |
|---|---|---|---|
| | **Without CVS** | **With CVS** | |
| T1 | 2.30–3.62 | 2.30–3.34 | 2.00 – 4.00 (0.50 increment value) |
| T4 | 0.56 | 0.56 | 0.56 (Constant value) |
| T5 | 1.11 | 0.40 | 0.40 to 2.00 |

Based on the established parameters for T1, T4, and T5, the sensitivity analysis of A% versus T5 for a single panel is depicted in Figure 4-21. This figure shows that A% increases with T5 across all tested T1 values, while T4 remains constant at 0.56 minutes. Notably, the increase in A% is more pronounced at higher T1 values. Essentially, both T5 and T1 positively affect A%. However, this trend contradicts the objectives of this study, which aims to reduce T5 (rework time) through the implementation of CVS.

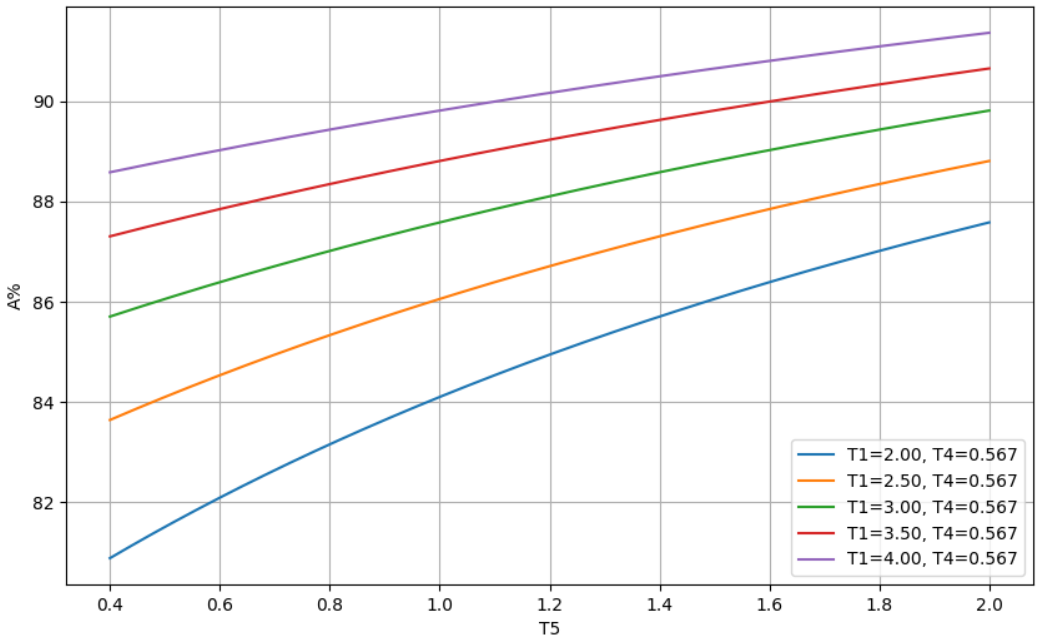



Figure 4-21. Sensitivity analysis of A% with respect to T5 for varying T1 values per panel

Despite this, the overall impact of CVS on A% is minor, where the highest range of A% varies between 81.5% to almost 87.5% when T1 at its best values, and this considered slight impact compared to other metrics such as P% and Q% values, which will be discussed later in the paper.

### 4.4.2 P%

To study the impact of CVS on the P% of the LGSFM, a sensitivity analysis was conducted to assess the impact of T5 on P% of the LGSFM to assemble one single panel. Assumptions for this analysis were based on Equation4-15 and are listed in Table 4-6.

Table 4-6. Study ranges of machine metrics for P% sensitivity analysis

| **Metric** | **Study Range (minutes)** |
|---|---|
| Ideal Cycle Time Per Panel | 2.30 (Constant value) |
| T1 | 2.00 to 4.00 min (0.50 increment value) |
| T5 | 0.40 to 2.00 |
| Actual Output Units | 1 (Constant value) |

Based on the assumptions depicted in Table 4-6, the sensitivity analysis of P% versus T5 for single panel is depicted in Figure 4-22.

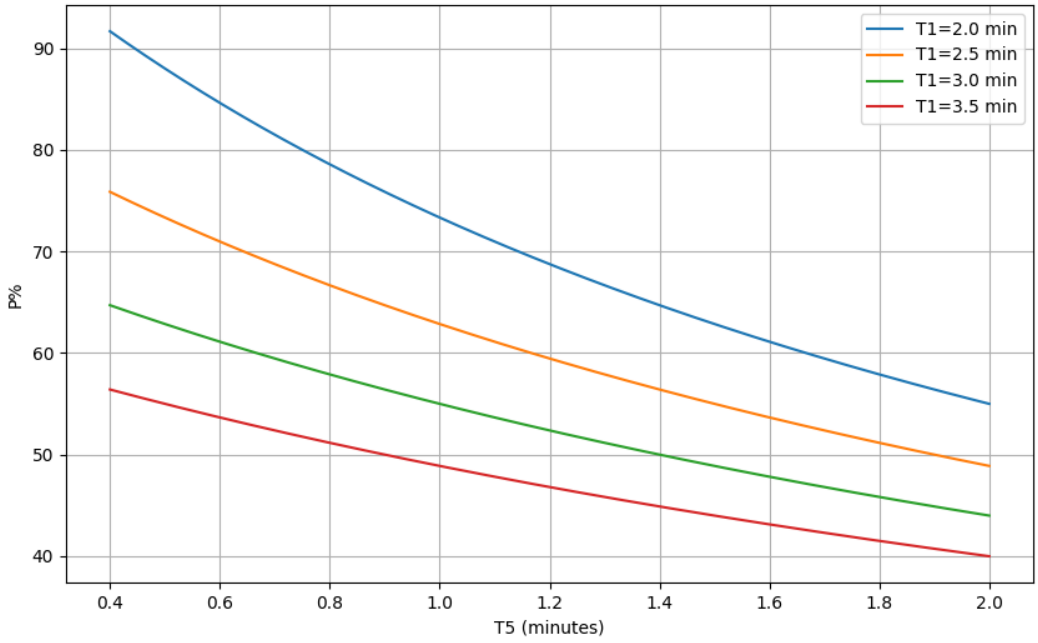



Figure 4-22. Sensitivity analysis of P% with respect to T5 for varying T1 values per panel

Figure 4-22 shows that P% decreases as T5 increases across various T1 values, indicating that reducing T5 enhances the P% metric. This underscores the importance of optimizing T5, particularly when T1 is at its most efficient value.

In summary, focusing on optimizing T5 is key to improving P%. As demonstrated, employing CVS can directly reduce T5, thereby significantly enhancing the P% value and improving the efficiency of the LGSFM.

### 4.4.3 Q%

The impact of CVS on the Q% metric can be assessed through a sensitivity analysis using Equation 4-16, considering that the number of fastening operations is 10 per panel and only the upper side of both the LHS and RHS of the panel is considered. Notably, the relationship between Q% and the number of defects is linear, as illustrated in Figure 4-23.

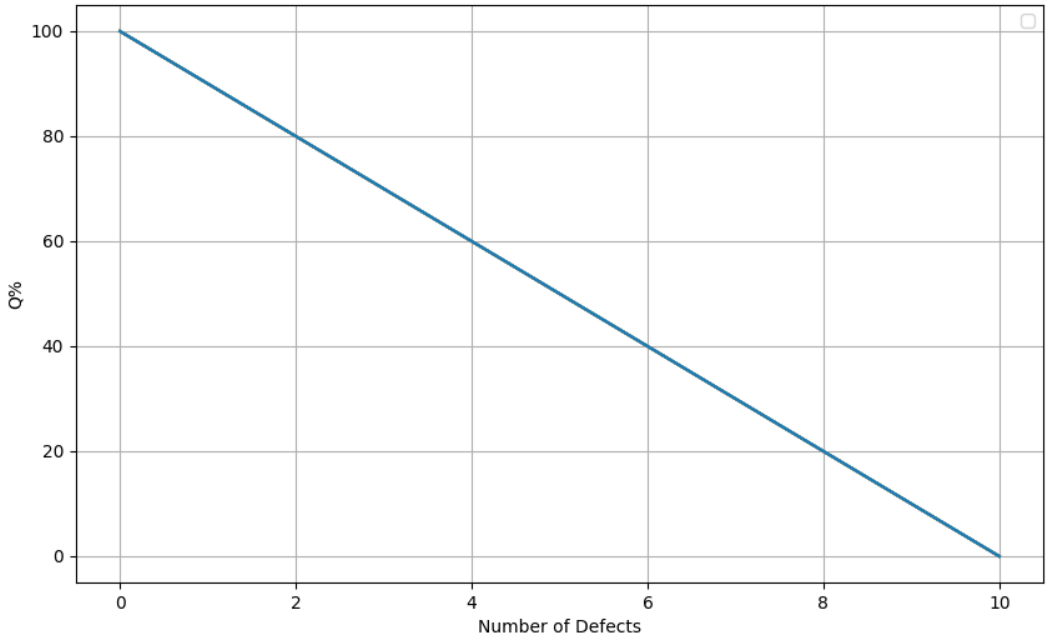



Figure 4-23. Sensitivity analysis of Q% with respect to the number of defects per panel

It is clear form Figure *4-23* that the number of defects significantly impacts the Q% value. The fewer the defects, the higher the Q% value, with an increase of 10% per defect prevented. This represents a significant gain in Q% for each defect prevented during the fastening operation. This underscores the importance of CVS in achieving optimal Q% values, as experimentally proven in Chapter 3, particularly in Table 3-3.

# Chapter 5 . Conclusion

## 5.1 General Conclusion

This research emphasizes the shift towards autonomous manufacturing systems, particularly within the context of smart factories, where real-time monitoring and quality control are essential. The study showcases the benefits of incorporating a computer vision-based deep learning (DL) system into offsite manufacturing of light-gauge steel (LGS) frames. The deployment of the computer vision system (CVS) has significantly enhanced the quality of LGS frames, providing superior improvements to the manufacturing process than traditional human-based methods.

The implementation of CVS notably decreased the occurrence of defect in LGS frame manufacturing, showcasing its ability to conduct precise real-time quality control and thus enhancing the overall accuracy of the manufacturing process. By cutting down on machinery downtime and reducing the need for extensive rework, the CVS increases efficiency of operations, reducing costs and boosting productivity.

Specific achievements from this study include achieving enhanced quality control through the CVS, which proactively detects and corrects defects in screw fastening. Additionally, productivity was increased for both machinery and operators due to reduced interruptions and rework. Furthermore, the CVS enables informed decision-making through detailed data analysis. These advancements collectively highlight the transformative impact of integrating CVS into automated light-gauge steel framing machine (LGSFM), establishing compliance in the panelized construction industry. The study's results clarify the operational benefits of the CVS, emphasizing its role in reducing rework, and machine waiting times. Using CVSs optimizes the manufacturing process, paving the way for future innovations in autonomous manufacturing for panelized construction.

## 5.2 Research Contributions

### 5.2.1 Academic contributions

1) The study showcased the potential of YOLOv8n architecture for real-time object detection in offsite panelized construction. It demonstrated how machine learning (ML) models can be adapted and integrated into specific industrial processes, advancing the application of CVS in LGS frame manufacturing.

2) The utilization of data-driven decision-making (DDDM) through the analysis of generated data using CVS in real time can serve as a reference for future research in optimizing LGS framing processes.

### 5.2.2 Contributions to industry practice

1) The implementation of the CVS notably enhanced quality control by lowering defect rates and increasing the precision of screw fastening in the LGS framing process. This improvement is vital for industrial applications where quality and reliability are essential.
2) This study demonstrated how integrating CVS can increase manufacturing efficiency by streamlining operations and decreasing wait time. This increases overall production efficiency, which is a substantial benefit for the prefabricated construction industry.
3) The use of Python, MySQL, and Grafana for real-time data collection and visualization contributes to industry practice by enabling better monitoring and analytics of machine performance. This enhances operational transparency and efficiency, which are critical in manufacturing environments.
4) The integration of a CVS into the manufacturing process of LGS frames not only enhances production efficiency and product quality, but also significantly affects the workforce by reducing rework and enhancing worker satisfaction.

## 5.3 Limitations and future work

Tackling the current limitations of the introduced CVS is crucial to maximizing its benefits. One important limitation to be addressed is the need to expand the volume and diversity of the dataset used for the models' training. Doing so could significantly refine the system's performance and increase the precision and robustness of the predictive models with greater reliability.

Currently, the models are unable to detect certain defects in screws or defective bits in screwdrivers before fastening, which can lead to significant quality issues and productivity losses in the LGS framing manufacturing process. This highlights the need to train the models to recognize these flaws in the process. Additionally, changes in materials or screwdriver types necessitates further training on a new dataset.

Future work will focus on training the models on a wider dataset to overcome current limitations and integrating the CVS with the machine's maintenance schedule to achieve ZDM. Such an

approach will enhance predictive maintenance and streamline operations, ultimately elevating product quality and operational efficiency.

# References

Abushwereb, M., Liu, H., & Al-Hussein, M. (2019). A knowledge-based approach towards automated manufacturing-centric BIM: Wood frame design and modelling for light-frame buildings. https://doi.org/https://doi.org/10.29173/mocs82

Alsakka, F., Assaf, S., El-Chami, I., & Al-Hussein, M. (2023). Computer vision applications in offsite construction. In Automation in Construction (Vol. 154). Elsevier B.V. https://doi.org/10.1016/j.autcon.2023.104980

Ameli, Z., Nesheli, S. J., & Landis, E. N. (2024). Deep learning-based steel bridge corrosion segmentation and condition rating using Mask RCNN and YOLOv8. Infrastructures, 9(1),3. https://doi.org/10.3390/infrastructures9010003

An, S., Martinez, P., Al-Hussein, M., & Ahmad, R. (2020). Automated verification of 3D manufacturability for steel frame assemblies. Automation in Construction, 118, 103287. https://doi.org/10.1016/j.autcon.2020.103287

Bae, J., & Han, S. (2021). Vision-based inspection approach using a projector-camera system for off-site quality control in modular construction: Experimental investigation on operational conditions. Journal of Computing in Civil Engineering, 35 (5), 04021012. https://doi.org/10.1061/(asce)cp.1943-5487.0000978

Bahar, M. E., Schokry, A. A., & Alhanjouri, M. A. (2023). Real-time predictive maintenance system of industrial equipment without historical failure data. Passer Journal of Basic and Applied Sciences, 6 (Proceedings, 4th International Conference on Recent Innovation in Engineering, Duhok, Iraq, Sep. 13–14, 2023), 266–288. https://doi.org/10.24271/psr.2024.188571

Bateman, B. W. (1997). Light gauge steel verses conventional wood framing in residential construction. Journal of Construction Education 2(2), 99–108

Burstrand, H. (1998). Light-gauge steel framing leads the way to an increased productivity for residential housing. Journal of Construction Steel Research, 46(1–3), pp. 183–186. https://doi.org/10.1016/s0143-974x(98)00141-2

Chabi Adjobo, E., Sanda Mahama, A. T., Gouton, P., & Tossa, J. (2023). Automatic localization of five relevant dermoscopic structures based on YOLOv8 for diagnosis improvement. Journal of Imaging, 9 (7) 148. https://doi.org/10.3390/jimaging9070148

Chikwendu, O. C., Chima, A. S., & Edith, M. C. (2020). The optimization of overall equipment effectiveness factors in a pharmaceutical company. Heliyon, 6 (4), e03796. https://doi.org/10.1016/j.heliyon.2020.e03796

Chung, Y. K., & Kim, K. H. (2006). Image processing based automatic inspection for assembly line of automobiles. Key Engineering Materials, 321–323, 1288–1292. https://doi.org/10.4028/www.scientific.net/kem.321-323.1288

Cinar, G. T., Thompson, J., & Srinivasan, S. (2015). Cost-sensitive optimization of automated inspection. 2015 IEEE International Conference on Big Data Oct 29-Nov 01, 2015, Santa Clara, CA, USA: proceedings.

CVAT. (n.d.). Computer Vision Annotation Tool (CVAT). Retrieved May 3, 2024, from https://www.cvat.ai/

Darwish, M., Mohsen, O., Mohamed, Y., & Al-Hussein, M. (2020). Integrated simulation and lean approach for production line improvement in a prefabricated panelized homebuilding facility. Proceedings, 28th Annual Conference of the International Group for Lean Construction, pp.649-660. https://doi.org/10.24928/2020/0030

De Vincenzo, V., Hichri, I. B., & Plapper, P. (2018). Industry 4.0-Implementation of an automated assembly line in a wooden modular house production plant: The case Leko Labs. Robotix-Academy Conference for Industrial Robotics (RACIR) 2018. https://doi.org/10.5281/zenodo.697610

Dierks, F., & Basler, A. G. (2004). Sensitivity and image quality of digital cameras. Image Quality of Digital Cameras. Retrieved November 24, 2023, from https://api.semanticscholar.org/CorpusID:12088786.

Dos Santos, V. B., Italiano, E. D., Alves, P., Fernandes, J. E., Pimenta, F. A., Pereira, N. B., & Ferreira, N. O. (2023). Online platform for home design and project management in

modular construction. Proceedings, 18th Iberian Conference on Information Systems and Technologies (CISTI). IEEE. https://doi.org/10.23919/cisti58278.2023.10211463

Girshick, R. (2015). Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 1440-1448). https://doi.org/10.1109/iccv.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.18127/j00338486-202109-11

Golatkar, A., Achille, A., & Soatto, S. (2019). Time matters in regularizing deep networks: weight decay and data augmentation affect early learning dynamics, matter little near convergence. https://doi.org/10.5555/3454287.3455245

Google Colab. (n.d). Google Colaboratory. Retrieved March 2, 2024, from. https://colab.research.google.com/

Gunawardena, T., & Mendis, P. (2022). Prefabricated building systems—Design and construction. Encyclopedia, 2(1), 70–95. https://doi.org/10.3390/encyclopedia2010006

Hansen, U. S., Landau, E., Patel, M., & Hayee, B. (2021). Novel artificial intelligence-driven software significantly shortens the time required for annotation in computer vision projects. Endoscopy International Open, 09(04), E621–E626. https://doi.org/10.1055/a-1341-0689

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 2961-2969). https://doi.org/10.1109/ICCV.2017.322

Hütten, N., Alves Gomes, M., Hölken, F., Andricevic, K., Meyes, R., & Meisen, T. (2024). Deep learning for automated visual inspection in manufacturing and maintenance: A survey of open-access papers. In Applied System Innovation (Vol. 7, Issue 1). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/asi7010011

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. Electronic Markets, 31(3), 685–695. https://doi.org/10.1007/s12525-021-00475-2

Jiang, T., Gradus, J. L., & Rosellini, A. J. (2020). Supervised machine learning: A brief primer. Behavior Therapy, 51(5), 675-687. https://doi.org/10.1016/j.beth.2020.05.002

Li, H. X., Yu, H., Gül, M., Al-Hussein, M., & Chmiel, D. (2015). An empirical study on the sustainability of panelized residential building construction in Canada. Proceedings of the International Construction Specialty Conference of the Canadian Society for Civil Engineering (ICSC). https://doi.org/10.14288/1.0371597

Liew, J. Y. R., Chua, Y. S., & Dai, Z. (2019). Steel concrete composite systems for modular construction of high-rise buildings. Structures, 21, 135–149. https://doi.org/10.1016/j.istruc.2019.02.010

Lin, R., Sajeevan Samarasinghe, D. A., & Rotimi, F. E. (2022). Development of a framework for quality assurance of off-site manufactured building components: A case study of the New Zealand housing sector. IOP Conference Series: Earth and Environmental Science, 1101(4), 042006. https://doi.org/10.1088/1755-1315/1101/4/042006

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 2980-2988). https://doi.org/10.1109/iccv.2017.324

Liu, H., Holmwood, B., Sydora, C., Singh, G., & Al-Hussein, M. (2017). Optimizing multi-wall panel configuration for panelized construction using BIM. In Proceedings of the International Structural Engineering and Construction Conference (ISEC), Valencia, Spain, Jul. 24–29, 2017. https://doi.org/10.14455/ISEC.res.2017.15

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. Proceedings, European Conference on Computer, Amsterdam, Netherlands, Oct. 11–14, 2016 (B. Leibe, J. Matas, N. Sebe, & M. Welling, Eds.). Part of the Lecture Notes in Computer Science book series (Vol. 9905). https://doi.org/10.1007/978-3-319-46448-0

Liu, X., Liu, C., Wang, P., Zheng, R., Zhang, L., Lin, L., Chen, Z., & Fu, L. (2023). UFNRec: Utilizing false negative samples for sequential recommendation. In Proceedings of the

SIAM International Conference on Data Mining (SDM) (pp. 46-54). https://doi.org/10.1137/1.9781611977653.ch6

Lu, B., Tan, M. J., & Qian, S. Z. (2016). A Review of 3D Printable Construction Materials and Applications. Proceedings, 2nd International Conference on Progress in Additive Manufacturing, pp. 330–335. https://hdl.handle.net/10356/84559

Luo, B., Kou, Z., Han, C., & Wu, J. (2023). A "Hardware-Friendly" Foreign Object Identification Method for Belt Conveyors Based on Improved YOLOv8. Applied Sciences, 13(20), 11464. https://doi.org/10.3390/app132011464

Lyu, J. J., & Chen, M. N. (2009). Automated visual inspection expert system for multivariate statistical process control chart. Expert Systems with Applications, 36(3 PART 1), 5113–5118. https://doi.org/10.1016/j.eswa.2008.06.047

Ma, J., Ding, Y., Cheng, J. C. P., Tan, Y., Gan, V. J. L., & Zhang, J. (2019). Analyzing the leading causes of traffic fatalities using XGBoost and grid-based analysis: a city management perspective. IEEE Access, 7, 148059–148072. https://doi.org/10.1109/ACCESS.2019.2946401

Malik, N., Ahmad, R., & Al-Hussein, M. (2019). Generation of safe tool-paths for automatic manufacturing of light gauge steel panels in residential construction. Automation in Construction, 98, 46–60. https://doi.org/10.1016/j.autcon.2018.11.023

Martinez, P., Ahmad, R., & Al-Hussein, M. (2019). A vision-based system for pre-inspection of steel frame manufacturing. Automation in Construction, 97, 151–163. https://doi.org/10.1016/j.autcon.2018.10.021

Martinez, P., Al-Hussein, M., & Ahmad, R. (2020). Intelligent vision-based online inspection system of screw-fastening operations in light-gauge steel frame manufacturing. The International Journal of Advanced Manufacturing Technology, 109, 645–657. https://doi.org/10.1007/s00170-020-05695-y

Martinez, P., Al-Hussein, M., & Ahmad, R. (2022). A cyber-physical system approach to zero-defect manufacturing in light-gauge steel frame assemblies. Procedia Computer Science, 200, 924–933. https://doi.org/10.1016/j.procs.2022.01.290

Martins, C., Santos, P., & Simões Da Silva, L. (2013). Lightweight steel framed construction system. Proceedings, Portugal SB13 Conference, Guimarães, Portugal, Oct. 30–Nov. 1, 2013, pp. 395–402. https://www.researchgate.net/publication/312160591

Muchiri, P., & Pintelon, L. (2008). Performance measurement using overall equipment effectiveness (OEE): Literature review and practical application discussion. International Journal of Production Research, 46(13), 3517–3535. https://doi.org/10.1080/00207540601142645

Nasteski, V. (2017). An overview of the supervised machine learning methods. HORIZONS.B, 4, 51–62. https://doi.org/10.20544/horizons.b.04.1.17.p05

Pequeno, J. M., Jorge, P., & Cruz, S. (2009). Timber-glass composite structural panels: Tectonics, sustainability & integrated energetic system solutions. In Proceedings of the 11th International Conference on Architectural and Structural Applications of Glass. Retrieved from http://www.gpd.fi

Rahimi, A., Anvaripour, M., & Hayat, K. (2021). Object detection using deep learning in a manufacturing plant to improve manual inspection. Proceedings, IEEE International Conference on Prognostics and Health Management, Jun. 7–9, 2021. https://doi.org/10.1109/ICPHM51084.2021.9486529

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Reza, A., Amin, J., & Chowdhury, S. R. (2022). A state of art review of light-weight steel structure for residential house construction. Journal of Structural Engineering, its Applications and Analysis. Retrieved from https://www.researchgate.net/publication/362593368

Sanjayan, J. G., & Nematollahi, B. (2019). 3D concrete printing for construction applications. 3D Concrete Printing Technology: Construction and Building Applications (pp. 1–11). https://doi.org/10.1016/B978-0-12-815481-6.00001-4

Sayahi, I., & Ismail, S. (2022). Design and implementation of an embedded vision system for industrial inspection. Proceedings, 9th IEEE International Conference on Sciences of Electronics, Technologies of Information and Telecommunications, May 28–30, 2022, pp. 567–572. https://doi.org/10.1109/SETIT54465.2022.9875471

Schneider Electric. (2019). RAM memory organization. Retrieved March 25, 2024, from https://product-help.schneider-electric.com/Machine%20Expert/V1.1/en/m251prg/m251prg/M2xx_-_Memory_Mapping/M2xx_-_Memory_Mapping-3.htm

Schulenburg, L. (2018). Increased process safety and efficiency through Automated Defect Recognition (ADR) in X-ray inspection. In Proceedings of the 12th European Conference on Non-Destructive Testing (ECNDT 2018), Gothenburg, June 11-15, 2018. Retrieved from https://www.ndt.net/search/docs.php3?id=22662

Sharma, N., Sharma, R., & Jindal, N. (2021). Machine learning and deep learning applications-A vision. Global Transitions Proceedings, 2(1), 24–28. https://doi.org/10.1016/j.gltp.2021.01.004

Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 10781-10790). https://doi.org/10.1109/cvpr42600.2020.01079

Tay, Y. W. D., Panda, B., Paul, S. C., Noor Mohamed, N. A., Tan, M. J., & Leong, K. F. (2017). 3D printing trends in building and construction industry: A review. Virtual and Physical Prototyping, 12(3), 261–276. https://doi.org/10.1080/17452759.2017.1326724

Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of YOLO architectures in computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 1680–1716. https://doi.org/10.3390/make5040083

Thai, H. T., Ngo, T., & Uy, B. (2020). A review on modular construction for high-rise buildings. Structures, 28, 1265–1290. https://doi.org/10.1016/j.istruc.2020.09.070

Unal, A. F., Kaleli, A. Y., Ummak, E., & Albayrak, O. (2021). A comparison of state-of-the-art machine learning algorithms on fault indication and remaining useful life determination by telemetry data. Proceedings, 2021 International Conference on Future Internet of Things and Cloud, Aug. 23–25, 2021, pp. 79–85. https://doi.org/10.1109/FiCloud49777.2021.00019

Wei, Y. (2023). Streamlining Decarbonized Construction through Automated Design and Drafting: Ribbed Precast Concrete Panels. MSc thesis, University of Alberta, Edmonton, AB, Canada. https://doi.org/10.7939/r3-ejj1-f489

Wu, T., & Dong, Y. (2023). YOLO-SE: Improved YOLOv8 for remote sensing object detection and recognition. Applied Sciences, 13(24), 12977. https://doi.org/10.3390/app132412977

Xi, J., Shentu, L., Hu, J., & Li, M. (2017). Automated surface inspection for steel products using computer vision approach. Applied Optics, 56(2), 184. https://doi.org/10.1364/ao.56.000184

Zhang, W., Yang, D., & Wang, H. (2019). Data-driven methods for predictive maintenance of industrial equipment: A survey. IEEE Systems Journal, 13(3), 2213–2227. https://doi.org/10.1109/JSYST.2019.2905565

# Appendix

### Total Number of Panels

```
SELECT
  CurrentTimeStamp,
  (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
   CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) AS ImplementedPanels
FROM
  t1_cycle_time
ORDER BY
  CurrentTimeStamp ASC;
```

### Total Number of Operations (Screws)

```
SELECT
  CurrentTimeStamp,
  ((CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
    CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) * 10) AS ImplementedPanelsMultiplied
FROM
  t1_cycle_time
ORDER BY
  CurrentTimeStamp ASC;
```

### Total Number of Ideal Screws

```
WITH ImplementedPanels AS (
    SELECT
      CurrentTimeStamp,
      ((CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) * 10) AS ImplementedPanelsMultiplied
    FROM
      t1_cycle_time
    ORDER BY
      CurrentTimeStamp ASC
),
TotalDefects AS (
    SELECT
        SUM(
            COALESCE(Panel1, 0) + COALESCE(Panel2, 0) + COALESCE(Panel3, 0) + COALESCE(Panel4, 0) +
COALESCE(Panel5, 0) +
```

```
            COALESCE(Panel6, 0) + COALESCE(Panel7, 0) + COALESCE(Panel8, 0) + COALESCE(Panel9, 0) +
COALESCE(Panel10, 0)
        ) AS total_defects
    FROM (
        SELECT * FROM lhs_defects_qty
        WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
        ORDER BY id DESC
        LIMIT 1
    ) lhs
    UNION ALL
    SELECT
        SUM(
            COALESCE(Panel1, 0) + COALESCE(Panel2, 0) + COALESCE(Panel3, 0) + COALESCE(Panel4, 0) +
COALESCE(Panel5, 0) +
            COALESCE(Panel6, 0) + COALESCE(Panel7, 0) + COALESCE(Panel8, 0) + COALESCE(Panel9, 0) +
COALESCE(Panel10, 0)
        ) AS total_defects
    FROM (
        SELECT * FROM rhs_defects_qty
        WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
        ORDER BY id DESC
        LIMIT 1
    ) rhs
)
SELECT
  ip.CurrentTimeStamp,
  (ip.ImplementedPanelsMultiplied - td.total_defects) AS Result
FROM
  ImplementedPanels ip,
  (SELECT SUM(total_defects) AS total_defects FROM TotalDefects) td;
```

## LHS-Number of Defects

```
SELECT
    (SELECT Panel1 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## RHS-Number of Defects

```
SELECT
    (SELECT Panel1 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
```

```
    (SELECT Panel5 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Total-Number of Defects

```
SELECT
  (
    (SELECT Panel1 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1)
  ) +
  (
    (SELECT Panel1 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1)
  ) AS total_defects;
```

## Total Rework & Transfer Time (T5)

```
SELECT
    (SELECT Panel1 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
```

```
    (SELECT Panel3 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Machine Operating Time (Tt- Total Cycle Time)

```
SELECT
  (
    (SELECT Panel1 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1)
  ) +
  (
    (SELECT Panel1 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1)
  ) AS total_time;
```

## Total Machine Running Time (T6)

```
SELECT
    (SELECT Panel1 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Total Tracks Loading Time (T2)

```
SELECT
    (SELECT Panel1 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Total Studs Loading Time (Ts)

```
SELECT
    (SELECT Panel1 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
```

```
    (SELECT Panel10 FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Total Track & Studs Loading Time

```
WITH Q1 AS (
    SELECT
        (SELECT COALESCE(Panel1, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel2, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel3, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel4, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel5, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel6, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel7, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel8, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel9, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel10, 0) FROM t2_tracks_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values
),
Q2 AS (
    SELECT
        (SELECT COALESCE(Panel1, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel2, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel3, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel4, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel5, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel6, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel7, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel8, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel9, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT COALESCE(Panel10, 0) FROM ts_studs_loading_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values
)
SELECT
    Q1.sum_of_last_values + Q2.sum_of_last_values AS total_sum_of_last_values
FROM
    Q1, Q2;
```

## Total Tracks Waiting Time (Twt)

```
SELECT
    (SELECT Panel1 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
```

```
    (SELECT Panel5 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM twt_tracks_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Total Studs Waiting Time (Tws)

```
SELECT
    (SELECT Panel1 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM tws_studs_waiting_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Machine Down Time (T4-Calibration Time)

```
SELECT
    (SELECT
        (SELECT Panel1 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel2 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel3 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel4 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel5 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel6 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel7 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel8 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel9 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
        (SELECT Panel10 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1)
    ) /
    (SELECT
        (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
         CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END)
    FROM t1_cycle_time ORDER BY CurrentTimeStamp DESC LIMIT 1
    ) AS CalibrationPerPanel;
```

## Machine Loading Time

```
SELECT SUM(sum_of_last_values) AS total_sum
FROM (
    SELECT
        (SELECT
                (SELECT COALESCE(Panel1, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel2, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel3, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel4, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel5, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel6, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel7, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel8, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel9, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel10, 0) FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1)
        ) +
        (SELECT
                (SELECT COALESCE(Panel1, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel2, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel3, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel4, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel5, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel6, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel7, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel8, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel9, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1) +
                (SELECT COALESCE(Panel10, 0) FROM t5_rework_time WHERE CurrentTimeStamp BETWEEN
$__timeFrom() AND $__timeTo() ORDER BY id DESC LIMIT 1)
        ) AS sum_of_last_values
    UNION ALL
    SELECT
        (SELECT
            (SELECT Panel1 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel2 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel3 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel4 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel5 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel6 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel7 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel8 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel9 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1) +
            (SELECT Panel10 FROM t4_calibration_time ORDER BY CurrentTimeStamp DESC LIMIT 1)
        ) /
        (SELECT
            (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
             CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
             CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
             CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
             CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
             CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
             CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
```

```
            CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END)
        FROM t1_cycle_time ORDER BY CurrentTimeStamp DESC LIMIT 1
        )
) AS combined_results;
```

## Total Panels Cycle Time (T1)

```
SELECT
    (SELECT Panel1 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel2 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel3 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel4 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel5 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel6 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel7 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel8 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel9 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) +
    (SELECT Panel10 FROM t1_cycle_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
ORDER BY id DESC LIMIT 1) AS sum_of_last_values;
```

## Actual Panel Cycle Time

```
WITH SumOfLastValues AS (
    SELECT
        CurrentTimeStamp,
        COALESCE(Panel1, 0) + COALESCE(Panel2, 0) + COALESCE(Panel3, 0) +
        COALESCE(Panel4, 0) + COALESCE(Panel5, 0) + COALESCE(Panel6, 0) +
        COALESCE(Panel7, 0) + COALESCE(Panel8, 0) + COALESCE(Panel9, 0) +
        COALESCE(Panel10, 0) AS sum_of_last_values
    FROM t1_cycle_time
    WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
    ORDER BY id DESC
    LIMIT 1
),
ImplementedPanels AS (
    SELECT
      CurrentTimeStamp,
      (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
       CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) AS ImplementedPanels
    FROM
      t1_cycle_time
    ORDER BY
      CurrentTimeStamp ASC
)
SELECT
    ip.CurrentTimeStamp,
    slv.sum_of_last_values / ip.ImplementedPanels AS Result
FROM
    ImplementedPanels ip
```

```
JOIN
    SumOfLastValues slv
ON
    ip.CurrentTimeStamp = slv.CurrentTimeStamp;
```

## Ideal Cycle Time Per Panel

```
WITH Q1 AS (
    SELECT
        (SELECT Panel1 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel2 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel3 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel4 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel5 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel6 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel7 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel8 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel9 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel10 FROM t6_machine_running_time WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) AS sum_of_last_values
),
Q2 AS (
    SELECT
        (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) AS ImplementedPanels
    FROM t1_cycle_time
    WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
    ORDER BY CurrentTimeStamp DESC
    LIMIT 1
)
SELECT
    (Q1.sum_of_last_values / Q2.ImplementedPanels)+15+10*Q2.ImplementedPanels AS result
FROM Q1, Q2;
```

## Availability (A%)

```
SELECT
  ((Q1.sum_of_last_values + Q3.sum_of_last_values) /
  (Q1.sum_of_last_values + Q3.sum_of_last_values + Q2.CalibrationPerPanel))*100 AS Result
FROM
  (SELECT
    Panel1 + Panel2 + Panel3 + Panel4 + Panel5 +
    Panel6 + Panel7 + Panel8 + Panel9 + Panel10 AS sum_of_last_values
   FROM t1_cycle_time
   WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
   ORDER BY id DESC
   LIMIT 1) AS Q1,
  (SELECT
    (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 +
    Panel6 + Panel7 + Panel8 + Panel9 + Panel10) /
    (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
```

```
     CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
     CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) AS CalibrationPerPanel
   FROM t4_calibration_time
   ORDER BY CurrentTimeStamp DESC
   LIMIT 1) AS Q2,
  (SELECT
    Panel1 + Panel2 + Panel3 + Panel4 + Panel5 +
    Panel6 + Panel7 + Panel8 + Panel9 + Panel10 AS sum_of_last_values
   FROM t5_rework_time
   WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
   ORDER BY id DESC
   LIMIT 1) AS Q3;
```

## Performance (P%)

```
SELECT
    (first_query.total_value / second_query.sum_of_QT1_and_QT5)*100 AS ratio
FROM
    (SELECT
        (
            (
                SELECT
                    (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 + Panel6 + Panel7 + Panel8 + Panel9 +
Panel10)
                FROM
                    t6_machine_running_time
                WHERE
                    CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
                ORDER BY
                    id DESC
                LIMIT 1
            ) +
            15 * (
                CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END
            ) +
            10 * (
                CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END
            ) * (
                CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
```

```
                CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END
            )
        ) AS total_value
    FROM
        t1_cycle_time
    WHERE
        CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
    ORDER BY
        CurrentTimeStamp DESC
    LIMIT 1) AS first_query,

    (SELECT
        (
            (SELECT
                (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 + Panel6 + Panel7 + Panel8 + Panel9 +
Panel10)
            FROM
                t1_cycle_time
            WHERE
                CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
            ORDER BY
                id DESC
            LIMIT 1
            ) +
            (SELECT
                (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 + Panel6 + Panel7 + Panel8 + Panel9 +
Panel10)
            FROM
                t5_rework_time
            WHERE
                CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
            ORDER BY
                id DESC
            LIMIT 1
            )
        ) AS sum_of_QT1_and_QT5
    FROM
        t1_cycle_time
    WHERE
        CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
    LIMIT 1) AS second_query
```

## Quality (Q%)

```
SELECT
    (((10 * Q1) - Q2) / (10 * Q1)) * 100 AS defect_percentage
FROM
    (SELECT
        CurrentTimeStamp,
        (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
        CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) AS Q1
    FROM
        t1_cycle_time
    ORDER BY
        CurrentTimeStamp ASC) AS Q1,
    (SELECT
```

```
        (
        (SELECT Panel1 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel2 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel3 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel4 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel5 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel6 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel7 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel8 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel9 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel10 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1)
        ) +
        (
        (SELECT Panel1 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel2 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel3 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel4 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel5 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel6 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel7 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel8 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel9 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
        (SELECT Panel10 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1)
        ) AS Q2) AS Q2;
```

## OEE%

```
SELECT
    (Combined_Result.final_result * QUA_query.defect_percentage)/10000 AS overall_result
FROM
    (SELECT
        (AVI_query.Result * PER_query.ratio) AS final_result
    FROM
        (SELECT
            ((AA1.sum_of_last_values + AA3.sum_of_last_values) /
            (AA1.sum_of_last_values + AA3.sum_of_last_values + AA2.CalibrationPerPanel))*100 AS Result
        FROM
            (SELECT
                Panel1 + Panel2 + Panel3 + Panel4 + Panel5 +
                Panel6 + Panel7 + Panel8 + Panel9 + Panel10 AS sum_of_last_values
            FROM t1_cycle_time
            WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
            ORDER BY id DESC
            LIMIT 1) AS AA1,
            (SELECT
                (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 +
                Panel6 + Panel7 + Panel8 + Panel9 + Panel10) /
                (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
```

```
                CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
                CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) AS CalibrationPerPanel
            FROM t4_calibration_time
            ORDER BY CurrentTimeStamp DESC
            LIMIT 1) AS AA2,
            (SELECT
                Panel1 + Panel2 + Panel3 + Panel4 + Panel5 +
                Panel6 + Panel7 + Panel8 + Panel9 + Panel10 AS sum_of_last_values
            FROM t5_rework_time
            WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
            ORDER BY id DESC
            LIMIT 1) AS AA3
        ) AS AVI_query,

        (SELECT
            (first_query.total_value / second_query.sum_of_QT1_and_QT5)*100 AS ratio
        FROM
            (SELECT
                (
                    (
                        SELECT
                            (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 + Panel6 + Panel7 + Panel8 +
Panel9 + Panel10)
                        FROM
                            t6_machine_running_time
                        WHERE
                            CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
                        ORDER BY
                            id DESC
                        LIMIT 1
                    ) +
                    15 * (
                        CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END
                    ) +
                    10 * (
                        CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END
                    ) * (
                        CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
```

```
                        CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
                        CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END
                    )
                ) AS total_value
            FROM
                t1_cycle_time
            WHERE
                CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
            ORDER BY
                CurrentTimeStamp DESC
            LIMIT 1) AS first_query,

            (SELECT
                (
                    (SELECT
                        (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 + Panel6 + Panel7 + Panel8 + Panel9
+ Panel10)
                    FROM
                        t1_cycle_time
                    WHERE
                        CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
                    ORDER BY
                        id DESC
                    LIMIT 1
                    ) +
                    (SELECT
                        (Panel1 + Panel2 + Panel3 + Panel4 + Panel5 + Panel6 + Panel7 + Panel8 + Panel9
+ Panel10)
                    FROM
                        t5_rework_time
                    WHERE
                        CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
                    ORDER BY
                        id DESC
                        LIMIT 1
                    )
                ) AS sum_of_QT1_and_QT5
            FROM
                t1_cycle_time
            WHERE
                CurrentTimeStamp BETWEEN $__timeFrom() AND $__timeTo()
            LIMIT 1) AS second_query
        ) AS PER_query
    ) AS Combined_Result,

    (SELECT
        (((10 * QQ1) - QQ2) / (10 * QQ1)) * 100 AS defect_percentage
    FROM
        (SELECT
            CurrentTimeStamp,
            (CASE WHEN Panel1 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel2 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel3 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel4 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel5 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel6 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel7 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel8 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel9 > 0 THEN 1 ELSE 0 END +
            CASE WHEN Panel10 > 0 THEN 1 ELSE 0 END) AS QQ1
        FROM
            t1_cycle_time
        ORDER BY
            CurrentTimeStamp ASC) AS QQ1,
        (SELECT
            (
            (SELECT Panel1 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
```

```
            (SELECT Panel2 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel3 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel4 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel5 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel6 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel7 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel8 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel9 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel10 FROM lhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1)
            ) +
            (
            (SELECT Panel1 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel2 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel3 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel4 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel5 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel6 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel7 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel8 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel9 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1) +
            (SELECT Panel10 FROM rhs_defects_qty WHERE CurrentTimeStamp BETWEEN $__timeFrom() AND
$__timeTo() ORDER BY id DESC LIMIT 1)
            ) AS QQ2) AS QQ2
    ) AS QUA_query
```