Technical University of Munich

TUM Department of Civil, Geo and Environmental Engineering

Chair of Computational Modeling and Simulation

Automated Calibration Methods for Digital Production Twins

Master Thesis

for the Master of Science Program Computational Mechanics

Author:	Alperen Kıral
Student ID:	[redacted]
Supervisors:	Prof. Dr.-Ing. André Borrmann
	Dr.-Ing. Alexander Braun
	Dr. Jan Christoph Wehrstedt
Date of Issue:	15. November 2020
Date of Submission:	15. May 2021

Abstract

It has been predicted by Moore’s law that the number of transistors on chips would double every two years which would increase the speed and capabilities of computers. Approximately 55 years later, we have witnessed the factuality of the law and new areas have consequently emerged that make use of advanced processing power of computers which were not available before. Born from Industry 4.0 practices, one of the newly trending fields is called digital twin technology.

Digital twin can simply be defined as a digital living twin of a physical system based on acquired online data. Even though the definitions in the literature are still changing, it is generally agreed that there must be a two-way interaction between digital and physical object, meaning that digital twins need to update itself continuously throughout the physical system’s life cycle. However, this may not be always possible due to a lack of sensors or communication insufficiencies between physical and digital twins. Stemming from that, an operator needs to update digital twin manually for an initial calibration or as regular maintenance that would cost many labor hours and possibly inaccurate outcomes. To overcome this challenge in a generalized way, a calibration software is being developed which integrates field data with a computer vision-aided camera system to automatically calibrate production line based digital twins. Ultimately, many experiments were conducted using a model production line by which the performance of the code was measured. It has been seen that the calibration software updates the digital twins very accurately and efficiently, depending on the configuration of the software. Overall, the automated calibration system performs much better than manual updating and should be preferred in most of the cases.

Zusammenfassung

Nach dem Mooreschen Gesetz wurde vorhergesagt, dass sich die Anzahl der Transistoren auf den Chips alle zwei Jahre verdoppeln würde, was die Geschwindigkeit und die Fähigkeiten von Computern erhöhen würde. Ungefähr 55 Jahre später haben wir erlebt, dass dieses Gesetz tatsächlich zutrifft und neue Bereiche entstanden sind, die die fortschrittliche Rechenleistung von Computern nutzen, die vorher nicht verfügbar waren. Geboren aus den Praktiken der Industrie 4.0, ist einer der neuen Trendbereiche die sogenannte digitale Zwillingstechnologie.

Der digitale Zwilling kann einfach als ein digitaler lebender Zwilling eines physischen Systems definiert werden, der auf erfassten Online-Daten basiert. Auch wenn sich die Definitionen in der Literatur noch ändern, ist man sich allgemein einig, dass es eine Zwei-Wege-Interaktion zwischen dem digitalen und dem realen Objekt geben muss. Das bedeutet, dass sich der digitale Zwilling während des gesamten Lebenszyklus des physischen Systems kontinuierlich aktualisieren muss. Dies ist jedoch nicht immer möglich, da es an Sensoren mangelt oder die Kommunikation zwischen physischen und digitalen Zwillingen unzureichend ist. Daraus ergibt sich, dass ein Bediener den digitalen Zwilling manuell für eine Erstkalibrierung oder als regelmäßige Wartung aktualisieren muss, was viele Arbeitsstunden und möglicherweise ungenaue Ergebnisse kosten würde. Um diese Herausforderung zu überwinden, wird eine Kalibrierungssoftware entwickelt, die Felddaten mit einem Computer-Vision-gestützten Kamerasystem integriert, um digitale Zwillinge in der Produktionslinie automatisch zu kalibrieren. Schließlich wurden viele Experimente an einer Modellproduktionslinie durchgeführt, bei denen die Leistung des Codes gemessen wurde. Es hat sich gezeigt, dass die Kalibrierungssoftware die digitalen Zwillinge sehr genau und effizient aktualisiert, abhängig von der Konfiguration der Software. Insgesamt schneidet das automatische Kalibrierungssystem wesentlich besser ab als die manuelle Aktualisierung und sollte in den meisten Fällen bevorzugt werden.

List of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ASF	Assist System Framework
AW	Automated Warehouse
BGR	Blue-Green-Red Color System
CAD	Computer Aided Design
CEE	Cyclic Event Evaluation
CPS	Cyber-Physical Systems
DOF	Degrees of Freedom
DTA	Digital Twin Aggregate
DTE	Digital Twin Environment
DTI	Digital Twin Instance
DTP	Digital Twin Prototype
FPS	Frames per second
HMI	Human-Machine Interface
HVS	Human Visual System
ILM	Indexed Line with Machining Stations
IoT	Internet of things
KPI	Key Performance Indicator
MPS	Multiprocessing Station

MUs	Mobile Units
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
RFID	Radio-Frequency Identification
RGB	Red-Green-Blue Color System
ROI	Return on Investment
SCM	Supply Chain Management
SLD	Sorting Line with Detection
TLB	Translation Lookaside Buffer
UI	User Interface
URL	Uniform Resource Locator
VGR	Vacuum Gripper Robot

1 Introduction

Nowadays, an increasing trend in manufacturing technologies such as IoT, artificial intelligence, cloud computing led to the concept that is widely pronounced as Industry 4.0. Although every company output different products and values, they all face similar problems related to productivity, growth, and associated processes. To address these problems, it is recommended to couple the physical system with a digital environment by the guidance of the following ideas: Better interconnectivity in production ecosystem and monitoring assets in real-time, smarter systems using AI that can take autonomous decisions, vertical & horizontal integration, flexible automation utilizing past and real-time data (i-SCOOP, 2021). When the machinery and production facilities are shaped around these ideas, the vision of Cyber-Physical Systems (CPS) emerges which is comprised of smarter machinery and facilities capable of sharing information, executing actions, and have control over each other individually (Kagermann, Wahlster and Helbig, 2013). In the last decade, the popularity of CPS has increased drastically which is depicted in Figure 1.1 by showing the number of articles having "Cyber-Physical Systems" in their title. The data is obtained by Google Scholar Advanced Search.

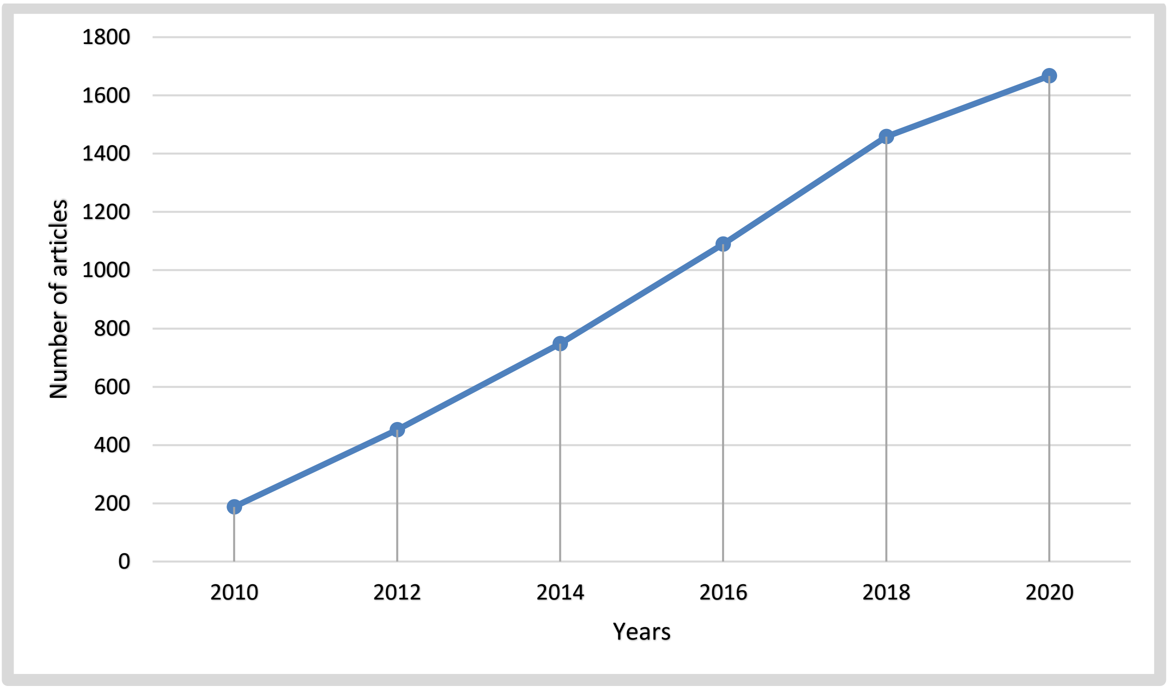


Figure 1.1: Number of articles containing "Cyber-Physical Systems" in the title

For illustrative purposes, an automotive diesel system factory of Bosch in China decided to combine IoT and Big Data by embedding physical sensors to their machinery and collecting information regarding the status of the machines. Following that, the prediction of any equipment failure or detection of a bottleneck during the production can be detected by using advanced data analyzing tools (AMFG / Autonomous Manufacturing, 2019). Such implementation clearly proves the point that a higher level of digitalization often boosts productivity as a more interlinked and holistic approach to create smart factories. All in all, it is safe to say that the demand for improved profitability, maximized OEE, and optimized productivity is the driving factor for most companies to switch to Industry 4.0 practices.

As the capabilities of computers increase exponentially, the ability to process a huge amount of data and storing them afterward are now available for many technological companies. This has opened up new ways that were not available before due to high costs and limited processing power. One of them, which has developed thanks to the advanced technology of Industry 4.0, is digital twin technology. It may help to figure out what is happening on a manufacturing line and even possess capabilities to predict what may happen in the future.

There exist lots of definitions for digital twin but all of them basically walk around the same ideas. A digital twin can be defined as a digitalized portrait of a real-physical system of processes or machinery which ideally operates as an online simulation based on the information & data coming from physical sensors of the corresponding processes. Physical object's properties and logical characteristics are cloned accurately, as Figure 1.2 demonstrates.

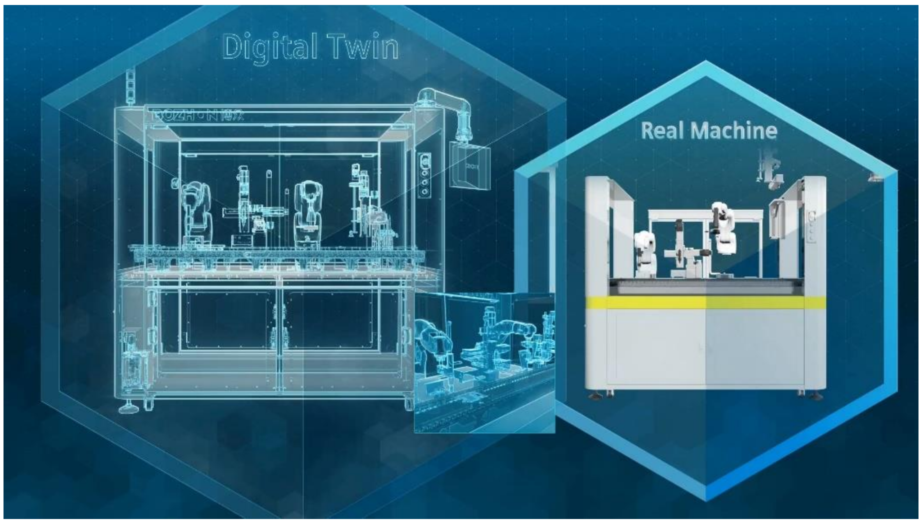


Figure 1.2: A digital twin is the virtual representation of a physical system (Siemens, 2021)

Supplementary to these definitions, Deloitte argues that digital twins provide more value compared to only-CAD or only-IoT systems. CAD has its drawbacks when a complex environment is the setting and IoT lacks the ability to model the interactions between elements (Parrott and Warshaw, 2017).

However, the main question is, how would it actually help organizations to improve their product cycles? The primary advantage of a digital twin is the capability of following the entire lifecycle of a product from the design stage until the disposal. This new business model helps companies detect the root cause of problems & defects sooner, achieve better product quality, and much higher customer satisfaction. Many companies have already switched to digital twins because of these reasons. For instance, a sewer network in Germany, North Rhine-Westphalia makes use of Siemens's control system to perform simulations and make sure that the sewer network functions properly (Breuer, 2020a). It has been also forecasted by Gartner Inc., a research and advisory company, that more than 50% of the world's leading companies will be utilizing digital twins to control and monitor their assets in such a way that recalibration of equipment and optimization of production lines are executed automatically without any human interaction (AMFG / Autonomous Manufacturing, 2019). In addition to that, it has been observed that a digital twin may digitally leave some traces which would then be used to determine any defects that occurred in the production stage. Later, this information can be quite useful to improve the quality in the design stage of a product (Cai *et al.*, 2017). So far, it can be easily seen that making use of a digital twin can be very beneficial for many companies and most of them have already started adopting digital twins in their processes. Nonetheless, the creation of a digital twin may undeniably a challenging task and must be carefully executed.

At the generation stage, it is crucial to take the right step to correctly integrate digital twin in a production process. There has to be an optimal level of detail in a digital twin. If there happens to be too much detail, then the simulation speed can be decreased considerably. Also, complexity and maintenance difficulties may outweigh the benefits that can be gained from it. Elimination of some sections that may not eventually yield significant value can be proposed as a solution for a complicated digital twin. Likewise, in the case of too much simplicity, then a digital twin may not reach the targeted value that it promises. Thereby, it is generally suggested to focus on a specific part of a process that possibly holds a scenario that provides benefits to enterprises. Depending on the success of the applied scenario, extensions of new scenarios can be applied.

The given situation proves that building a digital twin is a step-by-step process that must be carefully executed. Another possibly critical aspect that must be taken care of, while generating a digital twin, is capturing the geometric model. Not always necessary but some applications may require the geometrical information of the related physical components. For example, if it is desired to build a geometric digital twin of an industrial facility, dedicated software and devices are required to model the building's components such as pipes, columns, supports, etc.. (Brilakis *et al.*, 2020). An example case study can be seen in Figure 1.3.

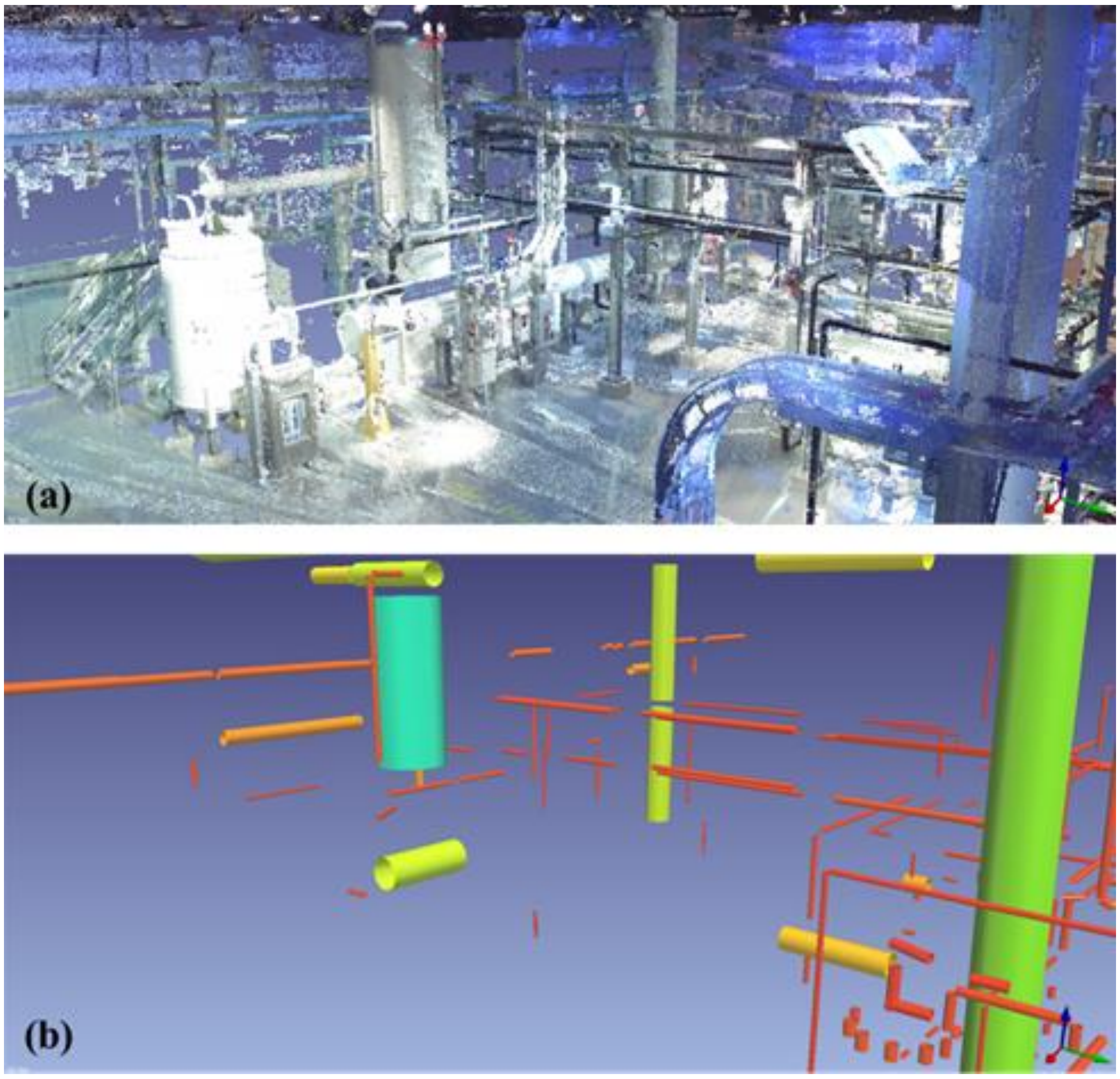


Figure 1.3: (a) Point cloud of an industrial facility, (b) automated cylinder extraction by EdgeWise Plant/MEP (Agapaki, Miatt and Brilakis, 2018)

Furthermore, it is argued that capturing the geometry of a room of a typical industrial facility can cost up to 5,200 hours of labor (Agapaki, Miatt and Brilakis, 2018). These cases prove that building a digital twin from scratch is prone to many difficulties that must be addressed cautiously. Unfortunately, not every difficulty that is associated with digital twins can be solved easily.

One of the hardest challenges of utilizing digital twin is keeping it up to date, either for the initial calibration or as maintenance throughout the lifecycle of the physical twin since an accurate digital twin is required for a correct decision-making. It is being argued that professional digital twin experts should be preferred over regular users for the update of vital sections of twins. Therefore, an easily maintainable digital twin with a pre-defined update frequency can deliver valuable practicality for some sectors. In airplane engine manufacturing, digital models of engines are being updated as frequently as possible since engineers need to know the instant states of engines in real-time (Biba, 2017). Evidently, the same ideas can be applied to any industrial sector that prioritizes safety and efficiency. In some sectors like civil engineering, it is not so crucial to update the models very often since it is not essential and even increases the costs through maintaining digital twin. But if it is the opposite case, then a new method for updating and maintaining a digital twin, which is cost-effective and accurate, would be an appealing option for many organizations.

1.1 Aims and Objectives

As it is shortly described above, one of the primary difficulties of utilizing digital twins is the frequent maintenance and update of the twin. Not all of the industrial sectors suffer from this dilemma, especially the ones where uncomplicated processes are modeled or the worth of frequent updates do not justify the cost of it. On the other hand, most of the high-level industrial facilities or processes consist of intricate & complicated simulation networks which mostly requires deep knowledge of both the implemented framework and the digital twin itself to be updated, not to mention the labor cost, low accuracy, and inclination towards mistakes that are associated with manual maintenance of digital twins. Thus, in this research, an automated solution is being investigated in an effort to replace the labor-intensive approach of updating and maintaining digital twin.

Due to the fact that there are many types of digital twin, it is unfortunately implausible to come up with a generalized solution for any of them. So that, the main focus of the research is given to the manufacturing type of digital twin which simulates an automated production line. The simulation that runs via them can offer ideal operational workflows and gives valuable information about the plant which can be useful for decision-making purposes. Consequently, it can be extremely critical in such a factory to have accurate and updated digital twins.

A closer look at the proposed digital twin would reveal that there are different production paths and stations where the workpieces are processed. In this manufacturing complex, the main optimization variable is the time spent between corresponding sections of production. Depending on the time array of each product, digital twin can propose an optimized workflow or product flow. To acquire a result, the user should normally input the time data manually into the digital twin to get the optimized workflow. Though, as it has been said before, the effort of manually extracting time data is quite a burden and mostly inaccurate.

To obtain an approximate figure on how much time an experienced operator would spend for updating digital twins, a demonstrative experiment was conducted. In the experimental setup, 9 different workpieces that each having 10 timing variables were updated using Tecnomatix Process Simulate. Firstly, a video of the running model plant is taken and then every timing variable was manually extracted using the video. In total, 90 variables are updated, and more than 2 hours are spent just to update a relatively small number of variables compared to an actual manufacturing plant.

Essentially, the following approach is being investigated by this research: A standard camera that is placed on top of the production unit and using computer vision algorithms on C# program, the locations of workpieces are detected. However, it may not be enough to detect only the location of the workpieces. If they look the same but have different production IDs, then it is not possible to differentiate them using computer vision. Since a generalized method is sought, an exchange of field data between the plant and the program needs to be implemented as well in order to fetch product information to the program. Therefore, as a communication protocol, OPC Unified Architecture (OPC UA) is chosen.

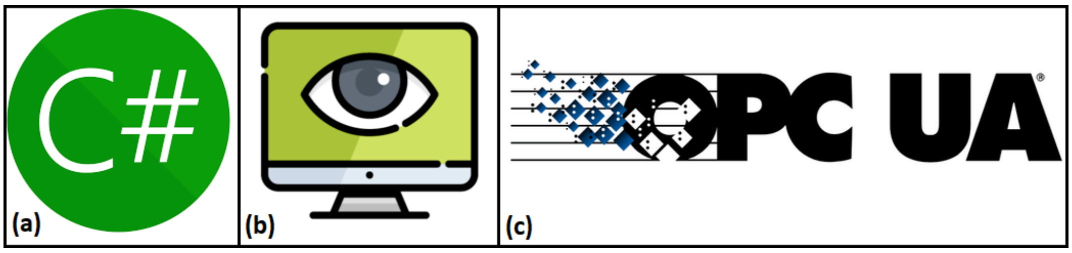


Figure 1.4: Main elements of the software: (a) C#, (b) Computer Vision, (c) OPC UA

2 Theoretical Background

2.1 Digital Twin

The concept of digital twin was first introduced by Michael Grieves in 2002 (Grieves and Vickers, 2017). He proposed the concept as a fundamental building block of Product Lifecycle Management (PLM) in the pursuit of digitalized manufacturing. Following him, many researchers have focused on the subject and contributed with new ideas. That is why there are lots of definitions of digital twin since it is a relatively new concept and scientific contributions are still shaping the boundaries of it. But if one must define it in simplest terms, a digital twin can be defined as a virtual replica of a physical system that simultaneously mimics using real-time data.

Sometimes, it is possible to mix the concept of digital twin with other forms of simulation types. If there is a manual data flow between the physical and digital object, then this is called a digital model. It is a quite common type in general engineering approaches. Assuming that there is an additional automatic data flow from a physical object to the digital object, then it turns into a digital shadow. In order to obtain a digital twin, both directions of the data flow must be automatically controlled (Errandonea, Beltrán and Arrizabalaga, 2020). A summary is depicted in Figure 2.1.

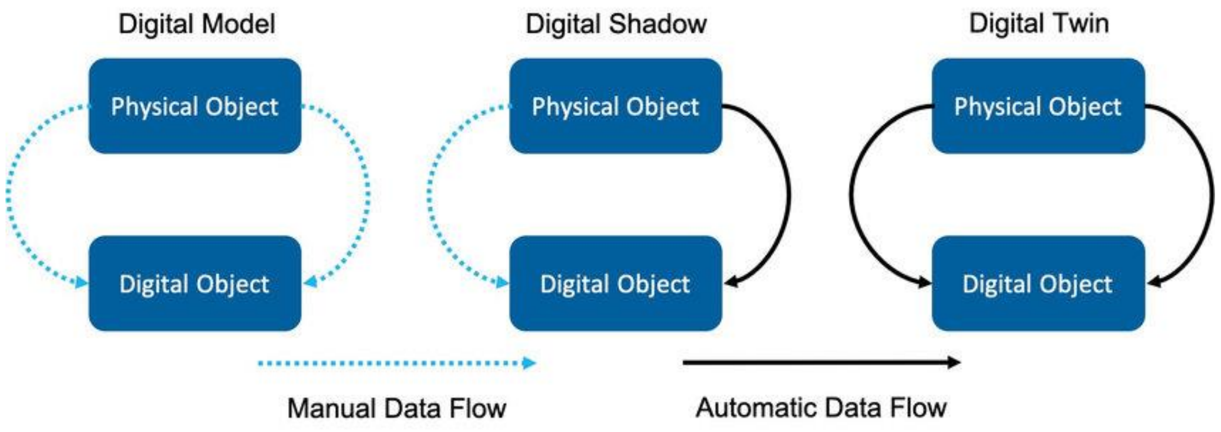


Figure 2.1: Distinctions between a Digital Model, Digital Shadow, and Digital Twin (Fuller *et al.*, 2020)

Owing to the fact that it can be so beneficial, digital twin technology is the main driving power of Industry 4.0 systems. In actuality, it has been forecasted by the industrial experts that digital twins will stay permanently throughout history as it provides much more efficient PLM and increased security to Supply Chain Management (SCM) through faster reaction time to the problems such as bottleneck or varying product demand (Breuer, 2020b). Even when digital twins are combined with other technological advancements such as product systems with modular production units, enhanced flexibility is assured through new opportunities that can be listed as improved decision making via simulations, distributed planning between production units, and execution of production plans and orders automatically (Rosen *et al.*, 2015).

The vision that helped the technology to arise was to simulate a physical system fully with the help of digital tools in an effort to improve the performance of the systems (GE Digital, 2021). Those improvements can be listed as:

- Fast ROI
- Higher Product Quality
- Increased Reliability
- Lower Maintenance Costs
- Better Customer Satisfaction

Some sectors can seriously harness these improvements by the collaboration of IoT, AI, and digital twin technology. Hence most of the researchers are focusing on potential applications of digital twins specifically in the following fields: Manufacturing, Healthcare Industry, and Smart Cities.

Manufacturing currently holds the lead in terms of the number of ongoing research projects in digital twin technology. The key motivation mostly originates from the historical need of saving money and time during production. There are many ways to implement a digital twin in a manufacturing scenario when coupled with other technologies such as IoT, AI. For example, before production, digital twins can be used to guide throughout the design process. Later, at the production stage, it is possible to use them to optimize production line performance or to inspect the machinery to avoid a potential breakdown. Following the production, depending on the quality of the product, coupling with AI algorithms can predict and make suggestions for further improvements. A real-life usage of a digital twin in the automotive industry can be briefly explained as: Engineers are using digital twins in combination with the company's analytical tools to observe how the drivers operate the cars. Then the system can suggest new features that can decrease the rate of accidents (W. Cearley *et al.*, 2017).

Following manufacturing, the healthcare industry also makes use of digital twins recently with the advancements in IoT technologies (Joordens and Jamshidi, 2018). The motivation of using them in this sector is to increase patient care. For example, it has been argued that personalized medicines based on the results coming from digital twins can save lives. An extremely detailed model of an individual patient combined with huge amount of similar patient data constructed inside a digital twin can guide doctors to choose the best medicines for a specific patient (Björnsson, Borrebaeck and Elander, 2020).

One of the best ways to build a smart city is to use IoT devices with digital twins. A virtual city can be built inside a digital twin as a testbed and based on the planned modifications, the possible effects of them can be observed which clearly accelerates the design process. For instance, Siemens developed a digital twin for its new headquarters of Building Technologies. This has allowed them to increase the energy efficiency of the building by running varying simulations and adjusting their design accordingly before the start of construction as it can be seen in Figure 2.2 (Siemens RT, 2021). With such implementation, it is also possible to examine the building's components through their operational lifetime.

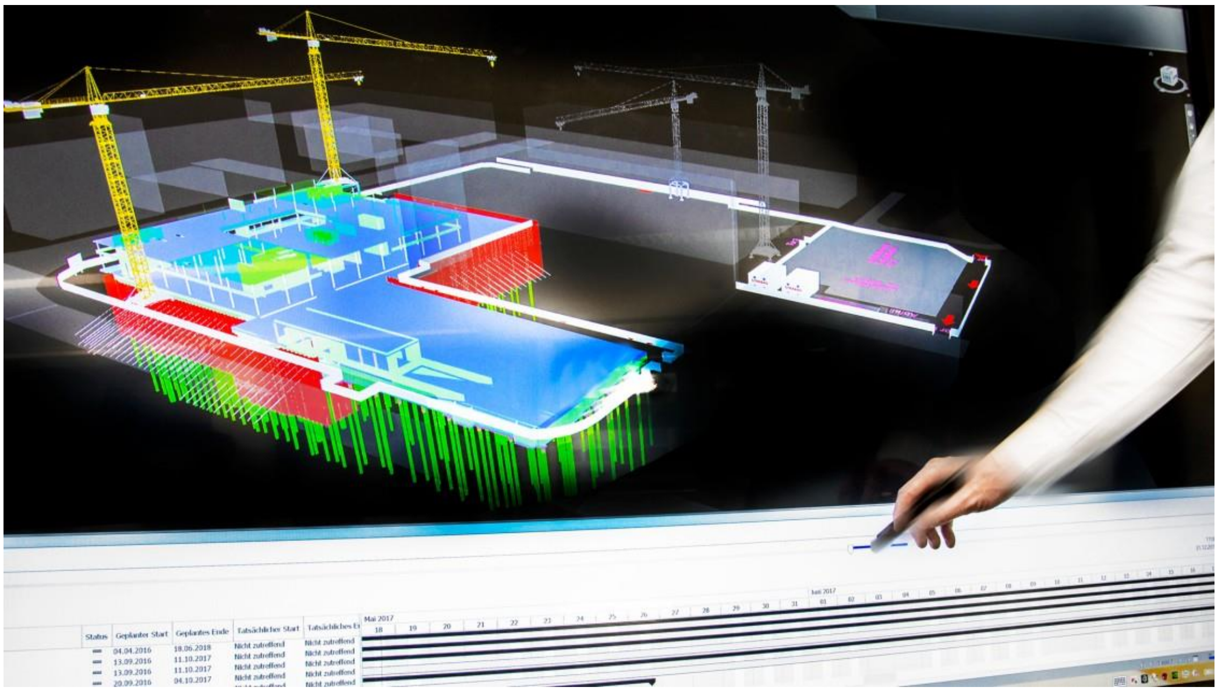


Figure 2.2: The Digital Twin of Siemens Building Technologies' new headquarters (Siemens RT, 2021)

As the new and innovative ways to solve problems with digital twins present themselves, fresh terms are being used to describe the procedures. For example, with the intention of having a simpler and modular structure, it was decided to separate different types of digital twin into three:

- Digital Twin Prototype (DTP)
- Digital Twin Instance (DTI)
- Digital Twin Aggregate (DTA)
- Digital Twin Environment (DTE)

Different digital twin types are aimed to be used in distinctive scenarios. The purpose of DTP is to design and analyze a physical product before it is manufactured. After the production, DTI can be used to track down every instance of the product during the life cycle. Following that, DTA can be employed which is a computing construct aggregated from the data of DTIs. Finally, as the name suggests, DTE is an application space designed for diagnostics, predictions of the future, and or learning from the life cycle of a product (Grieves and Vickers, 2017). The given division allows higher modularity, and it is often the case that companies choose a specific digital twin type that they need.

However, there exist many challenges associated with digital twins which are currently being addressed by many researchers. The first issue is with the cost. If a digital twin is coupled with AI and Big Data to perform neural network workflow, then depending on the size of the network, the cost of buying an adequate graphics card with a workstation can be quite demanding.

Security of digital twins and corresponding data are also trending topics in the scientific platforms that must be carefully approached. The data that digital twins hold, and control is generally very critical for many companies. To ensure the safety of the data, it is suggested to have a complex and multi-layered security approach to safely operate assets.

Another challenge related to the generation of digital twins can arise specifically if a Geometric Digital Twin is needed. It has been discussed that many of the facilities or buildings do not have pre-existing digital CAD models when they were in the construction phase. This obstacle is currently being passed by some dedicated software that makes use of capturing technologies such as photogrammetry or laser scanning for geometrical data, whereas radar or ultrasound can be applicable for capturing environmental data (Brilakis *et al.*, 2020). A sample 3D laser scanning application of LandTech Consultants is given in Figure 2.3.

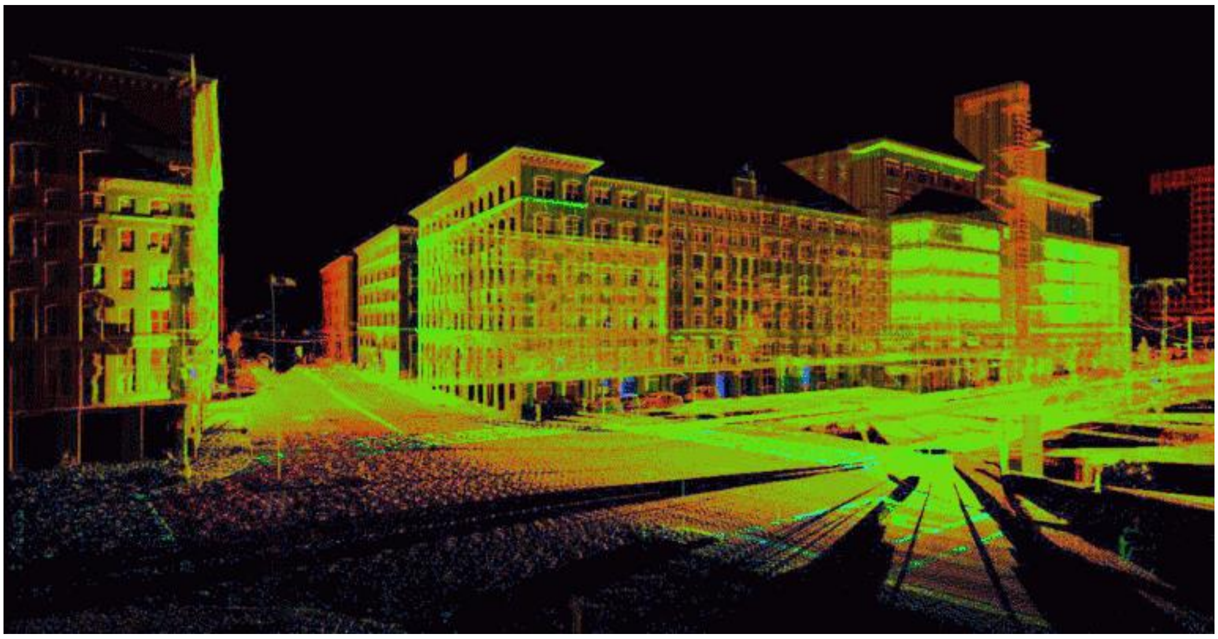


Figure 2.3: 3D laser scanning of sample infrastructure & buildings (LandTech, 2021)

2.2 Computer Vision

In the scientific area, there exist mainly two definitions of computer vision that are cut from the same cloth. The first definition describes that computer vision is aimed to mimic a human visual system (HVS) which is mostly used by biological point of view. As humans can extract visual information by image and video processing, researchers in the biological fields were inspired to do the same by using HVS models. It was assumed that HVS has predominantly the following characteristics: sensitivity to motion, 3D depth vision, recognition of features.

As opposed to the first definition, the second description is adopted by engineers who seek to automate the visual tasks done by computers and likewise biological experts, engineers also intend to accomplish what the human eye is capable of. Overall, it can be briefly said that computer vision is designed to construct autonomous systems that are as capable as the human eye and perform extraction of 3D and pull information from time-varying 2D data (Huang, 1996).

It was around the 1960s when the first illustrations of computer vision were attempted by Larry Roberts at MIT, where he studied the possible methods to extract 3D information by looking from a 2D perspective angle and he is therefore called “The Father of Computer Vision” (Huang, 1996). Followed by his efforts, there was a break of roughly 20 years until new studies have emerged in which the low-rank computer vision tasks such as detection of edges and image segmentation were studied.

The tasks that are being utilized by computer vision algorithms are quite varying and some of them are picked to be summarized here. Most of the tasks focus on recognition using a taken frame. For example, object segmentation can be used to partition a digital image into multiple segments to analyze the image easier. Object detection can be used to identify and locate the desired object in an image or video. If it is also desired to describe the object, then object recognition is capable of providing both the location and the type of the object. Furthermore, object classification would be useful in cases where the broad category of the object needs to be acquired. Moreover, there are some specific tasks that focus on analyzing the frame. Scene reconstruction is included in this category in which images can be converted into a 3D model (Mihajlovic, 2019).

As of now, computer vision is a large field that combines many sets of disciplines to be able to solve varying problems. Its capabilities are remarkable, thereupon many applications are developed understandably. For instance, the integration of deep learning and computer vision results in promising benefits. A representative case of face detection can be applied by making use of neural networks, a huge amount of training data consisting of the sample faces, and tuning parameters. Similar algorithms are currently being used by modern smartphones as well. Besides this, computer vision is commonly used in Healthcare Industry for the registering of pre-operative and intra-operative imaginaries (Szeliski, 2010). In Civil Engineering, it may be useful for the topographical modeling or construction of 3D models using lasers or aerial photographs. Moreover, computer vision has shown great advantages in the military. In the past, the target of the missiles was selected before the launch of it and the coordinates of the selected target would be the objective of the missiles. Currently, with the increased intelligence of computer vision algorithms, the missile guidance system escorts the missiles into a close region, and then the targets are acquired by using onboard object detection systems (Zarchan, 2012). Nevertheless, the most famous applications of computer vision are arguably in Automotive Industry. As autonomous vehicles are becoming more popular, the investment in computer vision technologies increases simultaneously. One of the pioneers in this discipline, Tesla, has been working on autonomous tasks such as lane changing, obstacle avoidance, automatic parking, automatic steering, and controlling of driving inputs. Those tasks depend on a good deal of sensors and 8 built-in cameras that constantly work and collect valuable information from the environment (Cohen, 2020). It is obvious that none of those tasks could be handled if the cameras were not continuously detecting objects such as road signs, lane lines, etc. More details are given in the figure below.

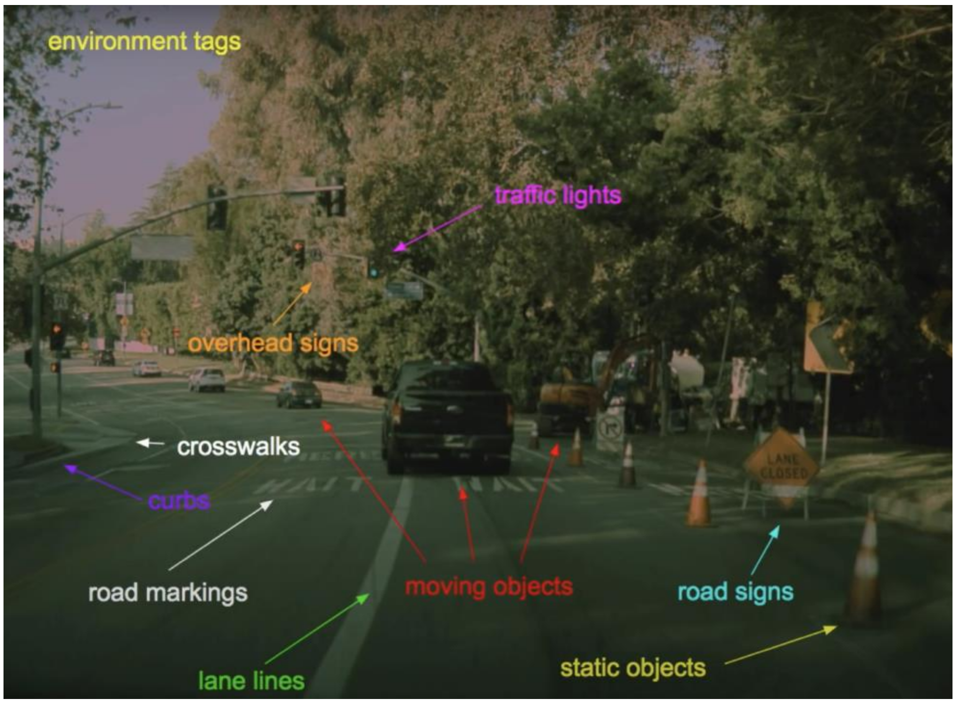


Figure 2.4: Object detection goals of an autonomous vehicle by computer vision algorithms (Cohen, 2020)

However, this is agreed by everyone that even though it has been 60 years since its first development, Computer Vision is still quite a challenging problem to work with and not perfectly accomplished yet. The human brain has been evolving for a long time and to survive, the human visual system should have performed rapidly and perfectly. For example, it is an extremely easy task for humans to detect faces and extract emotions expressed by facial figures. We are born with these skills. But it may be quite a demanding task for computers depending on the lighting conditions, viewing angle, and distance. This can be generalized under the topic of Object Detection in which computers still cannot process as accurately as humans. Furthermore, a vast amount of data coming from pixels also creates some challenges in this field since what computers get is only some integer numbers and forming meaningful information using pixels is very hard. As a comparison, Dr. Roger Clark states that the resolution of the human eye is around 576 megapixels (Shykora, 2020). Assuming that the corresponding color format of computer is RGB, meaning that each pixel has 3 integer value, there is going to be in a total of 1,728*10^9 values to be processed in each frame if computers are subjected to human standards. It is a too much workload for an average computer. But despite that, the tasks of computer vision that based on convolutional neural networks is believed to be the best possible algorithms and in fact, the performance of the convolutional neural networks was turned out to be very close to human’s capabilities (Russakovsky *et al.*, 2015).

3 Methodology

3.1 Overview

Firstly, it should be noted that the key focus of this research is on the calibration of the manufacturing type of digital twins that simulates a set of sequential operations to manufacture a product. In such a facility, raw materials can be processed in varying operation stations. To establish a secure material handling method between the stations and warehouse, autonomous systems such as conveyors, bucket elevators, automated guided vehicles, or vacuum grippers are commonly preferred in the industry (Thomas Publishing Company, 2021). If a proposed manufacturing plant holds the given characteristics, only then the approach suggested by this paper may be useful.

The main approach of the research is based on detecting the workpieces via a camera setup which is operated by the developed calibration software. However, the camera should be placed in a specific location where there would not be so many visual interferences that block the view of the camera. Otherwise, the discontinuities in detection can cause inaccurate results. To validate this theory, a FischerTechnik model plant coupled with Siemens's automation hardware and software has been used. The camera setup is then positioned in an appropriate location of the model plant where the calibration process is desired.

To test the capabilities of the approach, two digital twins of the model plant have been built. Those are implemented in the software packages, namely Tecnomatix Plant Simulation and Tecnomatix Process Simulate which have been designed by Siemens Digital Industries. Plant Simulation focuses on material flow simulation, whereas Process Simulate is specialized on 3D behavior simulation. These digital twins can successfully replicate the motions and material flows in the model plant. However, they had to be designed in a way such that they cannot continuously update themselves if the related variables in the physical plant change. Meaning that they give incorrect results with respect to what is actually happening on the model plant. To put it concisely, this unwanted situation creates the need for calibration.

On the software side, it is decided to use C# as the main framework to program the calibration software. The choice has been taking since Process Simulate and Plant Simulation have C# COM Interfaces that make it easier to communicate with the digital models. Even though it is much simpler to implement computer vision algorithms in Python, because of COM Interfaces, C# is chosen to preserve the software's integrity.

As a communication protocol between the model plant and calibration software, OPC UA is planned to be implemented since transferring runtime data between the server and the client is very fast and straightforward. There are many prepared C# codes for running OPC Client on a machine, which is preferred by this research. On the other hand, OPC Server is created using free and commercial software, specifically Softing dataFEED OPC Suite.

Combining all of these different segments of techniques result in a data flow described in the Figure 3.1. Basically, the calibration software is located in the middle as a bridge between the digital twins and the OPC server. To accomplish the goal of this research, there is a two-way data flow between digital twins and one-way data flow from OPC Server to calibration software. The data flow mostly results from running computer vision algorithms, OPC Client, user interfaces, and some other methods such as statistics, decision-making, and so on.

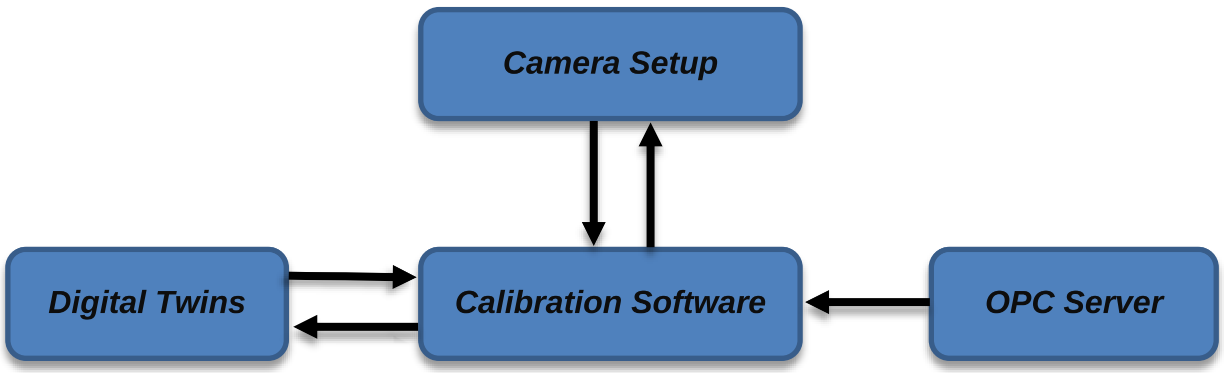


Figure 3.1: Data flow chart of the Calibration Software

3.2 Mini Manufacturing Plant: FischerTechnik Factory Model

As the cyber-physical systems emerge with the developments in Industry 4.0, the need for increased flexibility followed by modular production systems is surfacing more than ever. The main purpose is to achieve an automated system that can not only be efficient in terms of time, quality and money but should also react to changing conditions like variation of orders at a short notice. It is being argued that the given picture creates pressure on operators of the manufacturing industry who require additional support to handle the extra workload coming from flexibility (Zhou *et al.*, 2019).

To address the given challenge, Siemens initiated a project to solve the flexibility-related challenges in a plant through an Assist System Framework (ASF). It is designed to accomplish tasks such as monitoring the plant, deviation detection, bottleneck analysis, and corresponding optimization (Wehrstedt *et al.*, 2020). ASF has three sublayers which are UI Layer, Service Layer, and Data Layer. Project-specific UI has been implemented to guide the operator and it is being fed by the Service Layer which deals with the execution of domain and project-specific services. The required data for the Service Layer comes from Data Layer which incorporates Domain meta-model and Plant meta-model (Wehrstedt *et al.*, 2020).

Following the implementation, it has been decided by the researchers in Siemens to test the framework in a sample use case, which is bottleneck identification. In production terminology, a bottleneck can be defined as obstacles that reduce the ability of a system. These can be either process or equipment-related constraints or even something far more unexpected like management, policy, and so on (Timilsina, 2012). The particular use case that is pictured in Siemens focuses on a process-related constraint. The proposed bottleneck happens on a manufacturing line which slows down the material flow of production and causing longer operation time.

To simulate the given problem and assess ASF, a model manufacturing system has been built by using the actual automation software and hardware, which can be seen in Figure 3.2 (Wehrstedt *et al.*, 2020). The components of the model plant are being supplied by FischerTechnik and there exist many light sensors, actuators, and motors. All of the components are constructed in such a way that the workflow resemblance to a real plant is very accurate and the model will be summarized by categorizing the whole installment into four subsections: hardware, physical plant, logic behavior, and software.

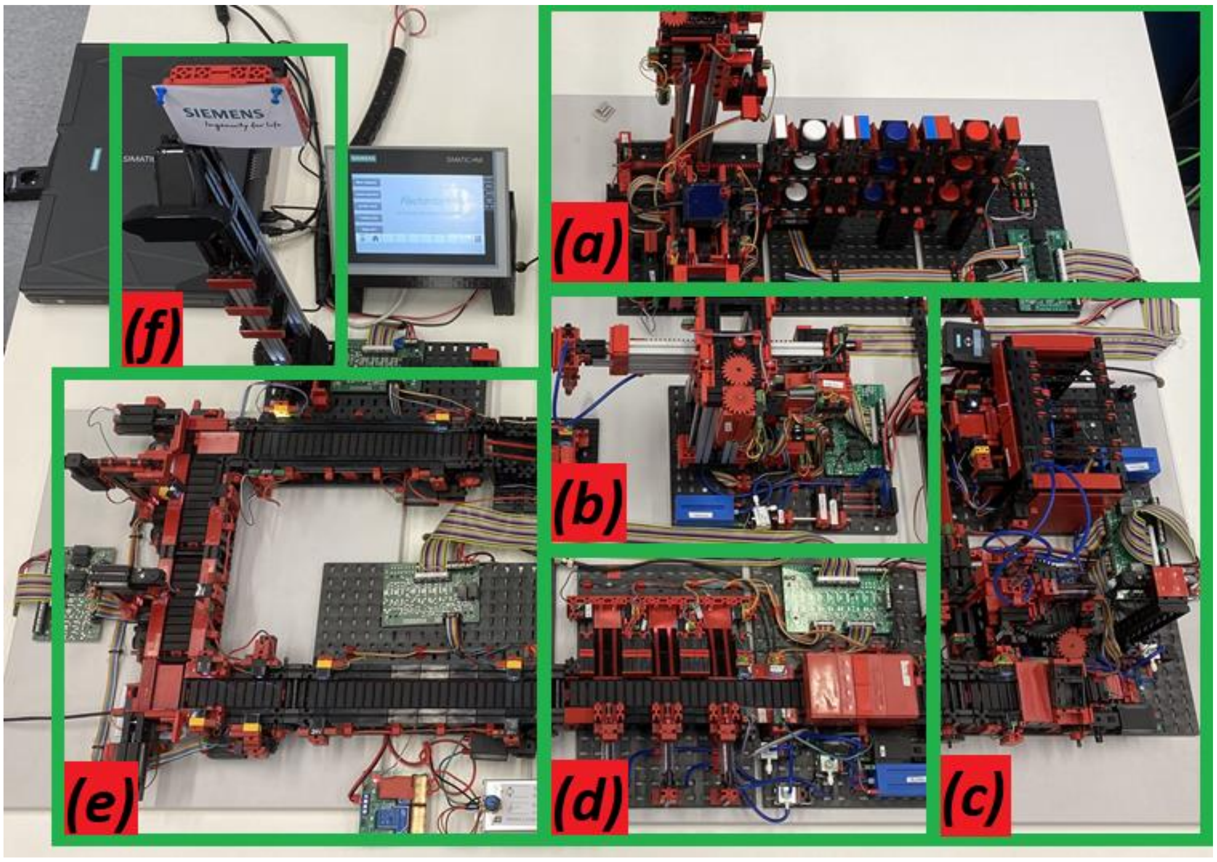


Figure 3.2: Bird’s-eye view of FischerTechnik Model Plant Assembly, (a) Automated Warehouse, (b) Vacuum Gripper Robot, (c) Multiprocessing Stations, (d) Sorting Line with Detection, (e) Indexed Line with Machining Stations, (f) Camera Setup

The core hardware of the model plant is made of 3 industrial PLCs that each control some different sections of the production sites. One of them is S7-1200, whereas the other two are the S7-1500 series. They communicate with each other using an industry-standard data communication over ethernet, namely Profinet. In addition to them, TP 700 – HMI is also installed in the plant. The purpose of the HMI is to have more effective control over the plant, increase the easiness of operations, and debugging any process-related problems. It has a 7-inch TFT touchscreen display for convenience and is connected to other PLCs via Profinet connection as well. Finally, an RFID system is adopted in the plant which is intended to create a flexible processing plant. By reading the information in RFID tags, which are located on the workpieces, the automation software decides on the amount of processing time in each specific processing station for the workpieces. The given RFID system consists of one RF680R reader, four RF615A antennas that are spread around the plant, one SITOP PSE 200U to monitor for overload and short-circuit conditions. It operates in the band of Ultra High Frequency (UHF) which ranges from 300 MHz to 1 GHz. Compared to other common frequencies, UHF is capable of reading from larger distances. Even though UHF does not work between objects containing water, the model plant is completely water-free (RF Editorial Team, 2018). Regarding the tags, standard UHF-compatible RFID tags are being utilized which have 96 bits EPC Memory to store all of the information such as workpiece number, process requirements, and quality flag. Hence, the values in the memory are changed using a UHF Reader. The hardware of the RFID system is shown in Figure 3.3. Similar to the previous devices, the RFID system is also connected to PLCs by Profinet.

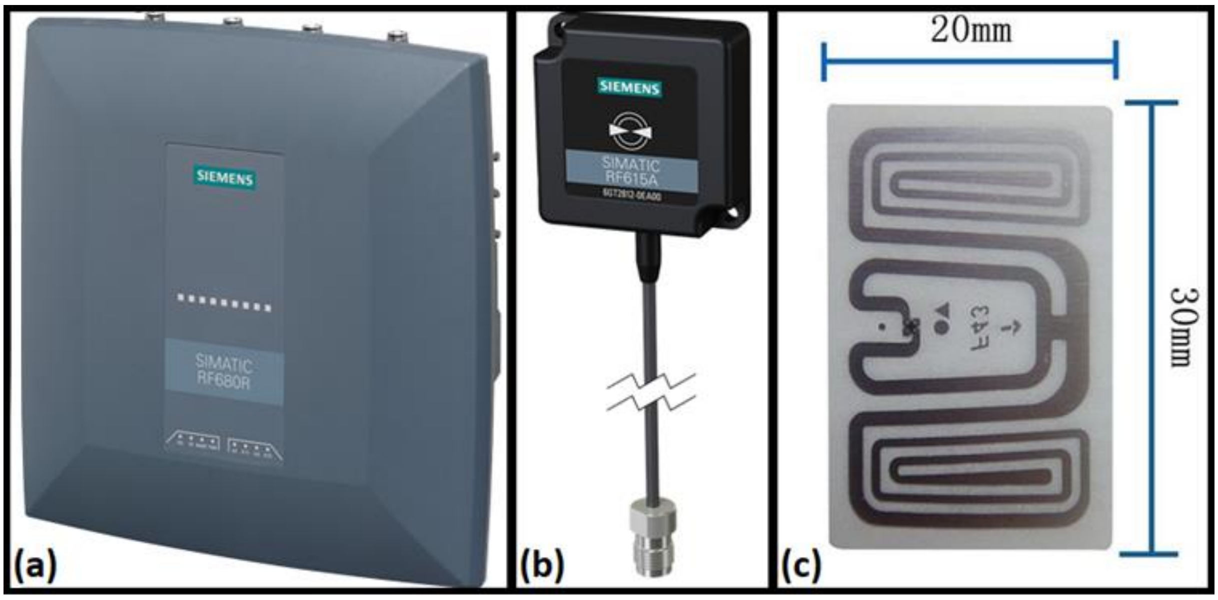


Figure 3.3: (a) RFID Reader, (b) RFID Antenna, (c) UHF RFID Tag

The construction of the physical plant is made of varying operational units that each serve a different purpose. The initial start point of the whole processing phase is the Automated Warehouse (AW). This is the primary structure for storing the workpieces. It has a similar shape with a 3 by 3 pile where the workpieces are stashed and thereby reaches a total capacity of 9 workpieces, as it can be seen in Figure 3.4. There is also an integrated 3-DOF picker robot that has prismatic joints to pick and place the workpieces in AW.

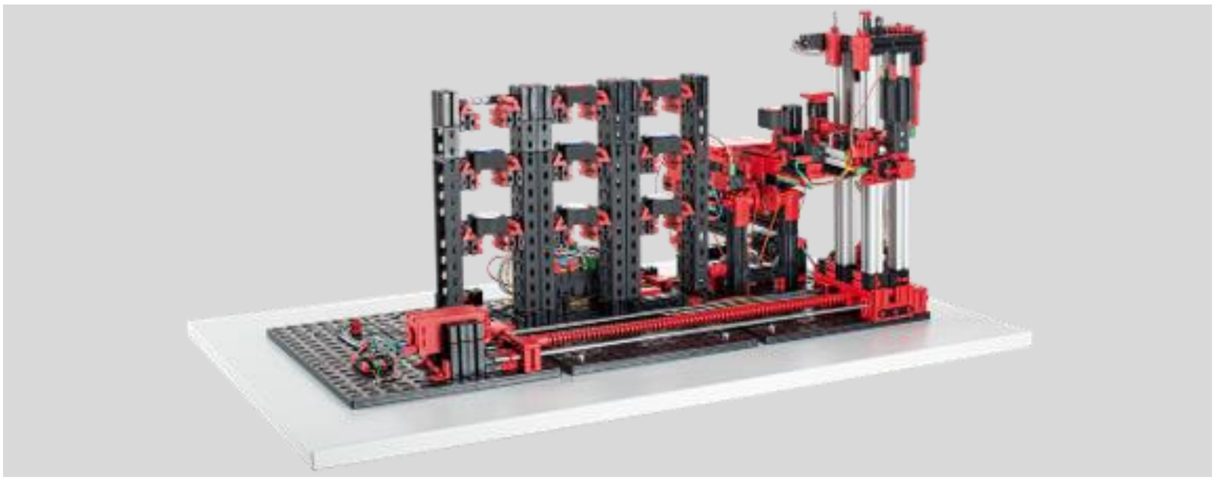


Figure 3.4: Automated Warehouse (AW) / Model Plant (fischertechnik GmbH ©, 2021a)

In the heart of the model plant, Vacuum Gripper Robot (VGR) is positioned which is designed to take the workpieces from AW and deliver them to the corresponding processing station. Similar to the robot in AW, VGR has 2 prismatic joints that enable 2-DOF motion, as shown in the figure below. In addition to them, one more rotational joint exists in the base which provides 3-DOF motion. In contrast to the AW robot which can only carry the workpiece vertically, VGR has a vacuum gripper as a tooltip that provides the ability to hold, lift and move the workpieces by generating a negative pressure with respect to atmospheric pressure.

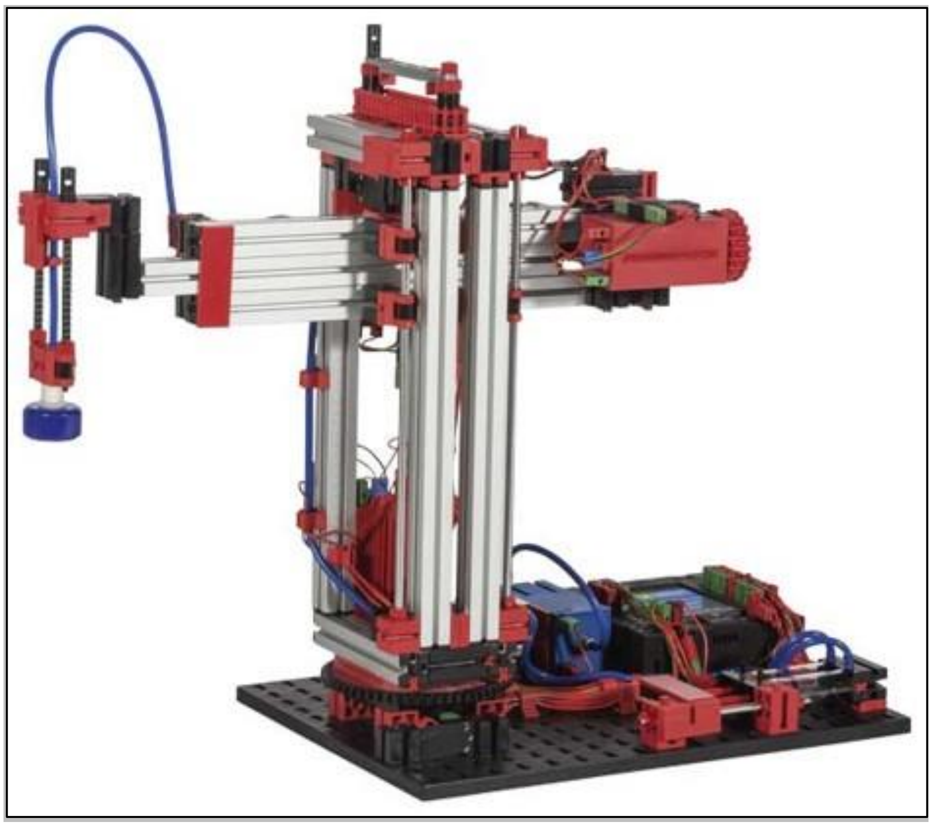


Figure 3.5: Vacuum Gripper Robot (VGR) / Model Plant (Kristie, 2017)

Following that, the workpieces arrive at the next locations where they are going to be processed. This can be either Indexed Line with Machining Stations (ILM) or Multiprocessing Stations (MPS). In MPS, there are two separate processing units that simulate an oven and sawing machine, as displayed in the figure below. In the beginning, a workpiece is carried to the oven unit by VGR. It is being operated there for some predefined amount of time which is obtained by reading the taped RFID tag below the workpiece. Shortly afterward, once the operation is ended, a 2-DOF gripper robot inside MPS lifts the workpiece and transfers it to the next operation unit, which is the sawing machine. When the processing in the sawing machine is finished as well, the workpiece is being pushed on top of a conveyor belt system that leads to a section called Sorting Line with Detection (SLD).

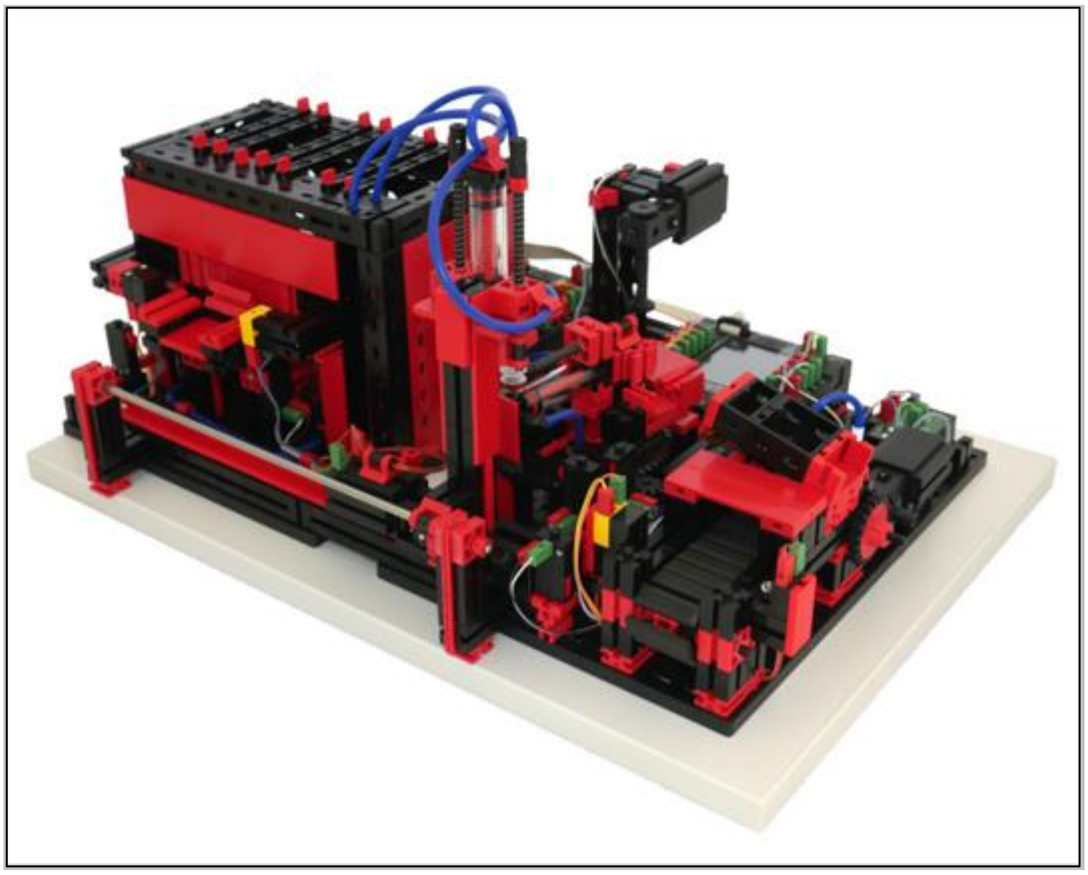


Figure 3.6: Multiprocessing Stations (MPS) / Model Plant (fischertechnik GmbH ©, 2021c)

The main objective of SLD is to sort and store the workpieces depending on their types in the buffers. However, if the workpiece also needs to be processed by ILM, then two sequential conveyor belt systems next to SLD would carry the workpiece to ILM. In there, two processing units can be found, namely milling and drilling machines. All the material flow in ILM is realized by 5 consecutive conveyor belt systems that are connected to each other. Eventually, the workpiece is being stored in the buffer of ILM.

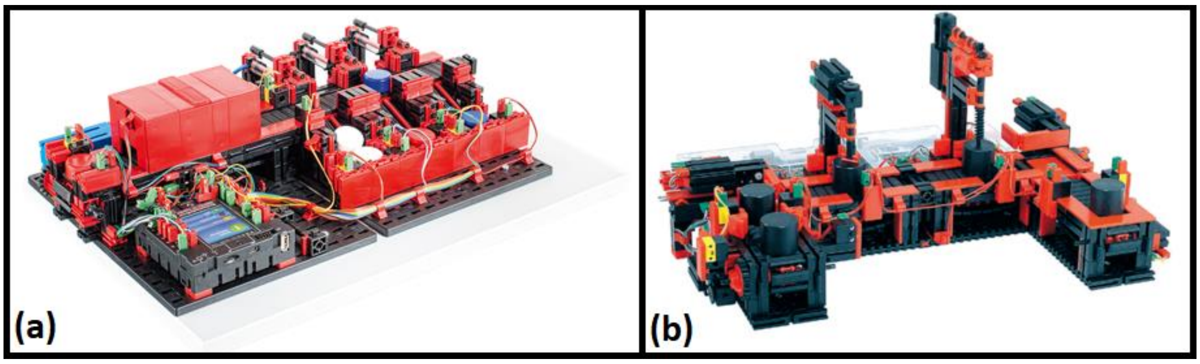


Figure 3.7: (a) Sorting Line with Detection (SLD) (fischertechnik GmbH ©, 2021d), (b) Indexed Line with Machining Stations (ILM) (fischertechnik GmbH ©, 2021b)

On the other hand, it can be argued that the most critical aspect of the model plant is the logical characteristics of it. Firstly, there exist three different kinds of workpieces and depending on the information written in their RFID tags, they may be processed in different operation units for different amounts of time. Thereby, there are three possible paths of production in total. A workpiece can be processed in only one of the stations or both stations. However, due to the structure of the model plant, a bottleneck can occur. This stems from the fact that the final location of a workpiece that is processed by MPS may intersect with another workpiece that is going to be processed by ILM, as it is described in Figure 3.8. Consequently, a flexible plant such as the FischerTechnik model requires a digital twin to smooth out potential bottlenecks via simulations and optimization.

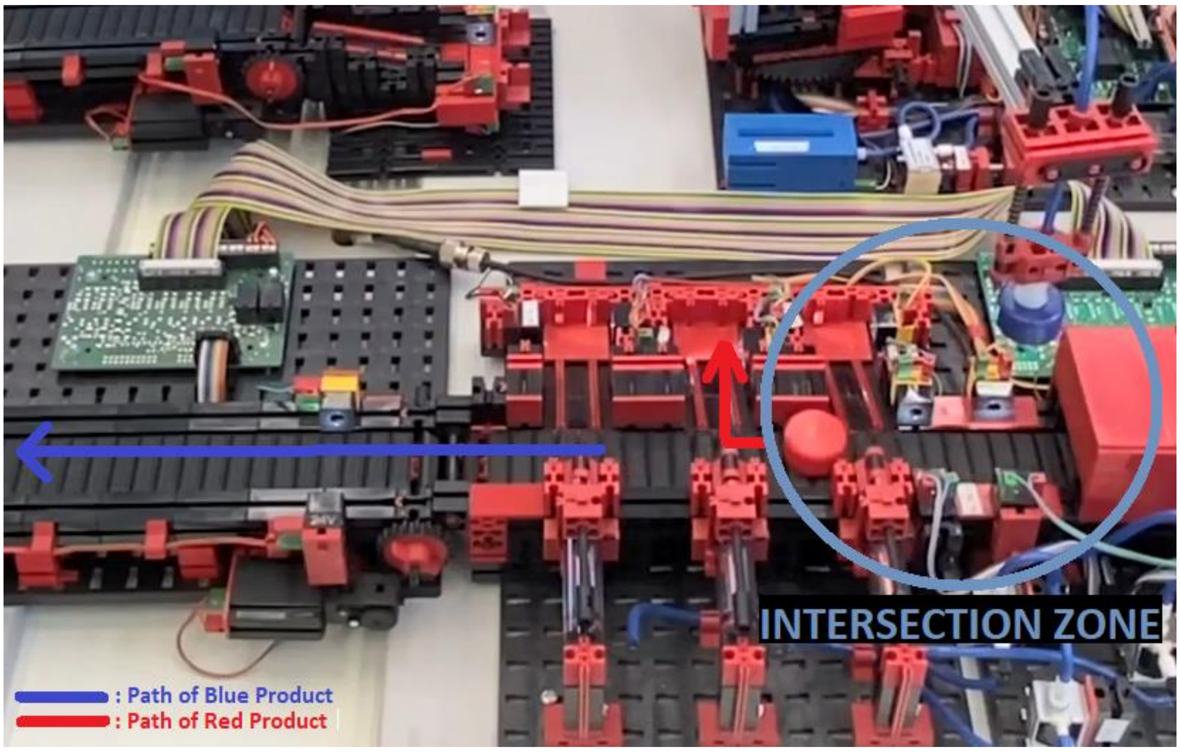


Figure 3.8: Bottleneck description of the model plant

3.3 Digital Twins of the Model System

As it has been pointed out previously, flexible manufacturing plants sometimes face bottlenecks depending on the physical setup and order of the workpieces during production. A similar problem was experienced in the FischerTechnik model and it was decided to make use of digital twins to monitor and eliminate the problem as much as possible.

A general approach to solving the problem via digital twins can be established with two consecutive procedures which are analysis and optimization. Considering that all of the variables related to workpieces should be well-known before the production begins, simulation inside a digital twin can analyze the material flow in the plant and output key performance indicators (KPIs) which then can be used to analyze and optimize the production process. In our given case, total manufacturing time is the most relevant KPI to be analyzed. Consequently, if the model plant experiences a bottleneck, simulations would reveal a higher KPI value than expected, meaning that total production time is higher due to a bottleneck.

It is stated that there exist many different strategies to eliminate bottlenecks for a more optimized plant. If applicable, one of the easiest methods is to change the order of particular workpieces which are somehow causing a bottleneck. Other solutions include more costly techniques such as altering the physical plant configuration or changing production steps of particular products (Wehrstedt *et al.*, 2020). Therefore, numerous simulations with different order configurations can be experimented with by digital twins and the configuration that has better KPI values can be eventually selected. To verify the given theory, it was decided to implement two digital twins of the model plant by using two different frameworks.

The first framework is called Tecnomatix Plant Simulation and it was developed by Siemens PLM Software. It is based on material flow analysis and the software aims to increase the manufacturing system performance via several methods such as modeling, simulation, and optimization even though it is primarily used as a tool for design and searching of new possibilities in the industry. There are some sample cases in which Plant Simulation is very capable. For example, Plant Simulation can make use of discrete event simulation (DES) and statistical analysis tools to optimize logistics, utilization of machines, workshop requirements (Siemens Digital Industries, 2021a).

What is required from Plant Simulation in this research is to perform material flow simulations of the FischerTechnik model and determine whether there is a bottleneck in the plant or not. However, before all else, the plant needs to be designed and modeled in Plant Simulation.

The study is conducted in the model version of 15.1 and the license type is selected as “Highest Available” to be able to access all of the tools and libraries in Plant Simulation. The next step is the creation of Mobile Units (MUs). Plant Simulation provides MUs to imitate material flow through the simulation model. In the model plant, there are three different kinds of workpieces and it is necessary to create three types of MUs in the Plant Simulation model. Each of these workpieces has RFID tags attached in their bottom sides and process-related values are saved in those. To model such information in Plant Simulation, specific MUs should be selected and then the user-defined tab needs to be opened. From there, the user can add any type of attribute that is desired. In our case, five attributes are integrated which can be listed as: Oven Time, Saw Time, Drill Time, Mill Time, and Product ID.

Once the configuration of MUs is done, the next step is to add a Source object from which the workpieces can be created. Following that, a table that contains the type and number of workpieces needs to be specified. The final step of workpiece creation is to add a control method by which specific attributes can be assigned to each workpiece.

The rest of the design step only consists of building up the physical plant in Plant Simulation. There are a couple of tools to assist the user with that. Firstly, physical motions of workpieces are simulated in Plant Simulation via Converter, Turntable, and PickAndPlace. Converter mimics the linear motion of a conveyor and takes length and speed as inputs to process incoming MUs. On the other hand, Turntable and PickAndPlace imitate a circular motion and both of them need to be assigned with two tables that contain information about angles and corresponding time between those angles.

Secondly, another very practical tool is called SingleProc. It is a single processing station that can handle one MU at a time and has three critical attributes which are called Processing Time, Set-up Time, and Recovery Time. For Processing Time, the corresponding operation time of the workpiece in a specific station is given as an input that was previously assigned by Source.

Thirdly and lastly, Store is a very commonly used element to stock the MUs. It has three attributes which are X-dimension, Y-dimension, and Z-dimension to be filled in and thereby simulating a real warehouse with any dimensions.

All of these described methods are combined in a simulation to model the Fischer-Technik plant in Plant Simulation, which is presented in Figure 3.9.

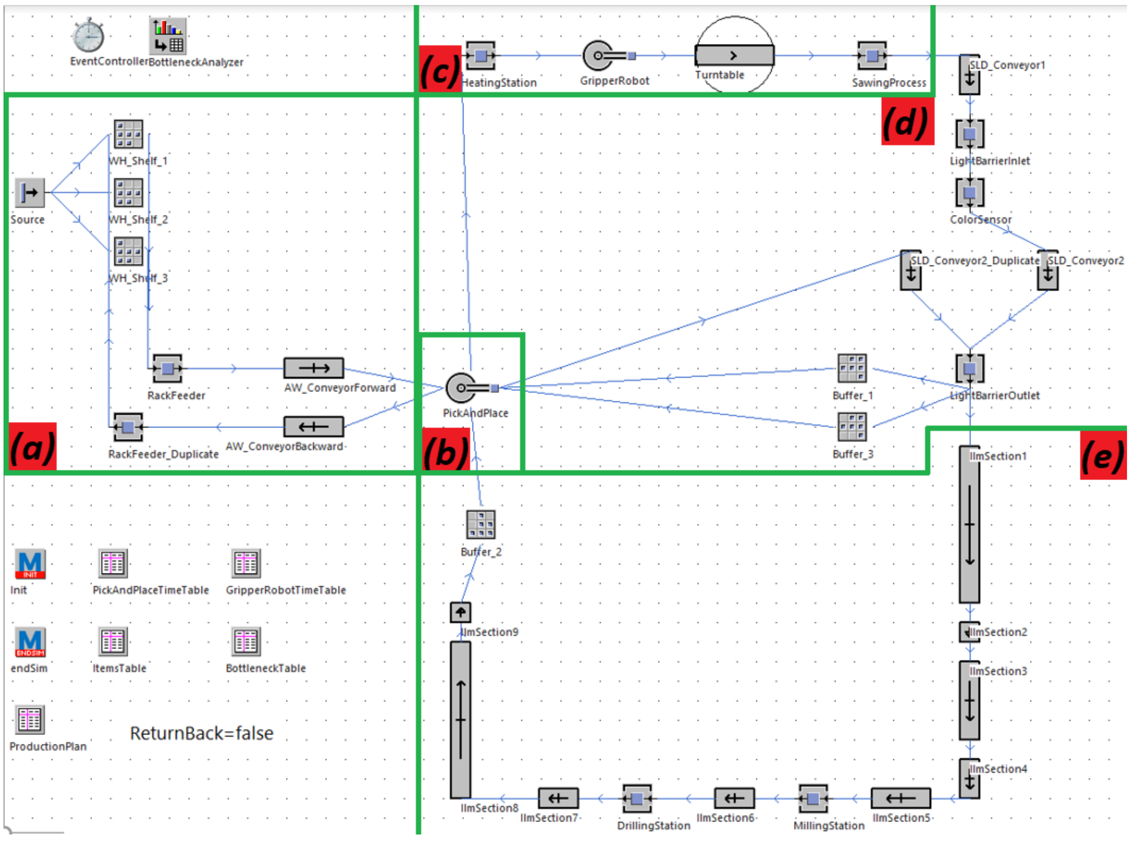


Figure 3.9: Plant Simulation model of the FischerTechnik plant (a) Automated Warehouse, (b) Vacuum Gripper Robot, (c) Multiprocessing Stations, (d) Sorting Line with Detection, (e) Indexed Line with Machining Stations

Besides Plant Simulation, there is a second framework on which another digital twin of the model plant is built upon. It is called Tecnomatix Process Simulate and developed by Siemens PLM Software as well, likewise Plant Simulation. The case study in this research has been implemented in version 15.0.2 of Process Simulate. The software is a result of necessity coming from process design and validation in a 3D environment. It may be beneficial for companies to virtually validate, optimize and commission complex manufacturing environments before actual physical construction which results in a faster launch and higher production quality (Siemens Industry Software, 2011). However, it should be noted that 3D data of the products and organizational knowledge must be present beforehand since resource modeling of 3D models and kinematics of them are essential for Process Simulate.

Many simulation tools are implemented in Process Simulate that allow the user to virtually realizes every aspect of a manufacturing complex. Accordingly, the software is very advantageous in many ways. It is capable of significantly reducing the risk of actual implementation by virtual simulations, decreasing the amount of time and labor cost by validation tools, ensuring compliance with the ergonomic safety rules, and improved product quality by testing different manufacturing methods (Siemens Digital Industries, 2021b). On the grounds of them, it has been decided to use Process Simulate for the implementation of the second digital twin of the model plant.

The initial step of constructing a digital twin in Process Simulate is the design procedure. It is always a good idea to have 3D models of each component in the plant before going further from this step.

Any desired structure is built and linked with each other and then exported as a CAD file so that it can be manipulated by Process Simulate. When all of the components are successfully added under Resources, Placement Manipulator can be used to set up the model same way as the physical plant. It allows the user to manipulate the position and orientation of any component by rotation and translation in any direction.

Following the design phase, information about the moving parts needs to be fed to the simulation model. This step is called the Kinematics phase in which the motion of components is introduced. Unfortunately, there are many moving parts in FischerTechnik model and each of them needs to be manually explained to the simulation model. At first, to perform the modeling of kinematics, all related components have to be selected, and then Set Modeling Scope needs to be applied which can be found under the Modeling tab. After that, Kinematics Editor should be opened and links must be created, which are collections of components that move relative to each other. In the same window, it is possible to connect two links with a joint. The type of joint can be either prismatic or revolute. And finally, from the Joint Jog window, Upper Limit and Lower Limit value of a joint can be specified.

There exist two main types of simulation modes in Process Simulate. These are called Cyclic Event Evaluation (CEE) and PLC. In CEE mode, the simulations are carried out offline, meaning that no signals are coming from an outside source. Hence, operations start by triggering operation start conditions or transitions. In contrary to CEE mode, the operations in PLC mode can only start if signals generated from Operation Start Signals are connected to a Logic Resource and actuated from there by triggered PLC signals. Also, there are many options for external connection types in PLC Mode. Users can make selection between OPC DA, OPC UA, Simulation Unit, PLCSIM Advanced, WinMOD. All of these different connection types offer data to Process Simulate, either by virtual behavioral simulations or from an actual automation. Under the given facts, it was decided to use PLC as the simulation engine and OPC UA as an external connection.

To set a successful connection between PLCs and Process Simulate, an OPC UA server needs to be running which converts the hardware communication protocol of PLC into OPC protocol. In our case study, the OPC UA server is configured in such a way that every 100 milliseconds, PLC variables in the server are updated.

From here on, the most important and complex step comes which is Signal Importing. To reach the signals in an OPC UA server correctly, every signal that is required to run the simulation needs to be imported one by one. For example, in the case of the model plant, motor encoder values and Boolean actuator signals are imported to Process Simulate to actuate their counterparts in the simulation as well.

In the model plant, there are many stations in which workpieces are processed in diverse ways. Consequently, there needs to be a dedicated operation for each station in the simulation. In total, 2 different kinds of operations have been used in this study, namely Non-Sim and Object Flow. With Object Flow operation, an object can be transported between predefined points in the simulation space. Additionally, Non-Sim operations are used to add a new event such as Attach Event or Detach Event. The specified events are methods to lock or unlock two separate objects in space.

Process Simulate offers substantially two kinds of Logic Resources. The first version belongs to kinematic devices in the simulation which are modeled in the Kinematics phase and it is a bit more extended version of the second type of Logic Resource because of added kinematics tools. Inside the Logic Resource window, 5 tabs can be found that are named Entries, Exits, Parameters, Constants, Actions. In the Entries tab, all of the signals which are planned to be used for the specific purpose of Logic Resource need to be given. In the Exits tab, operation start signals can be connected with any other signals so that they can be turned on and off. In Parameters and Constants tabs, any desired variable can be created or calculated. In the Actions tab, the joint movements of a kinematic device can be defined. In this part, two methods are commonly used, namely Jump Joint and Move Joint to Value. The jump Joint method immediately sets the joint´s value to a predefined value under Joint Value Expression. This method is more ideal for motors with encoders since the position of a joint is known exactly. The second method, Move Joint To Value method helps the user to set a target for the joint’s value with a given velocity and acceleration.

The last step of designing a digital twin in Process Simulate lies in Material Flow Viewer. It is required to visualize the material itself and its movements in simulation mode. In the window of Material Flow Viewer, successive operations of individual workpieces need to be connected so that the effect of those operations can be visualized. Ultimately, a successful Process Simulate model of the FischerTechnik plant can be achieved with these methods, that can be seen in the figure below.

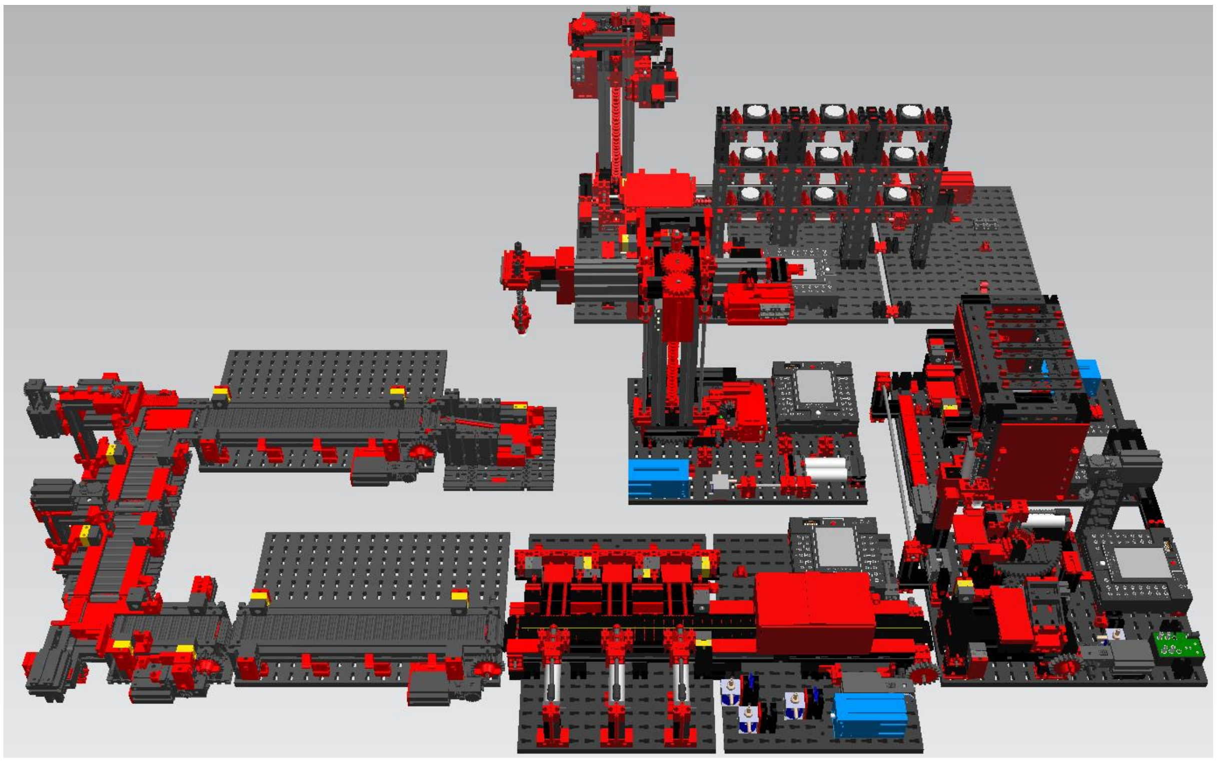


Figure 3.10: Process Simulate model of the FischerTechnik plant

Experiments after this step revealed that by using real-time data coming from OPC UA and Process Simulate, it is possible to create a digital twin that represents the model plant exactly. The motion of motor actuated components is represented precisely and the motion of the workpieces on the conveyor was estimated by the total time spent on the specific sections of conveyors.

3.4 The Calibration Software

3.4.1 Hardware Configuration and Requirements

Camera Setup

One of the most innovative approaches in this research is the automatic calibration method which is a product of collaboration between field data and computer vision-aided camera system. Thus, it should be ensured that both of them operate as reliable as possible. Field data is being collected using OPC UA and it is already accepted as a secure and trustworthy mechanism for transferring information between Client and Server. On the other hand, the situation is not as straightforward in the case of a camera system.

Many factors are involved when it comes to choosing an appropriate smart camera system for the machine vision and imaging needs. These factors can be listed as: camera type, resolution requirements, imaging rate, interconnection standards (Photonics Media, 2021). There are mainly two different kinds of camera types, namely area scan, and line scan. Producing 2D images with vertical and horizontal components can be achieved by an area scan camera, whereas a line scan camera grabs a slice and continually build an image by stacking the slices. In terms of resolution requirements, selection can be done based on a field of view and the minimal feature that needs to be captured. Similarly, the choice of imaging rate depends on how often an image needs to be captured so that critical details would not be missed. Lastly, an interconnection standard of a camera is also very important and can significantly affect the machine vision performance since it alters the data transfer rate in each cycle. If high speeds are required for the application, then USB3 Vision or Camera Link HS supported camera modules can be chosen. All in all, considering the given specifications and characteristics of the model plant, it was decided to employ Logitech C920 camera which has 15 megapixels & 30 FPS setup that features autofocus and automatic low-light correction. Even though the camera is equipped with USB 2.0 connectivity which is much slower than the latest solutions, it still serves the purpose adequately.

Another critical issue is the placement of the camera setup on the model plant. To demonstrate the research's goal, it was proposed to use one section of the plant, ILM, which mostly consists of conveyors and two processing stations. Furthermore, the section offers great visibility for the camera system to track the workpieces if the setup is placed high enough. Under the given circumstances, a tower-shaped structure has been built by using FischerTechnik components just near ILM to place and fix the camera system in an elevated location, described in the figure below. One of the downsides of such a structure is the instability stemming from the vibrations and motions of mobile parts. This could have induced negative effects on the reliability of the results coming from the computer vision algorithm. Therefore, to reinforce the construction against bending and oscillations, two aluminum rods have been integrated which later proved to be an effective solution.

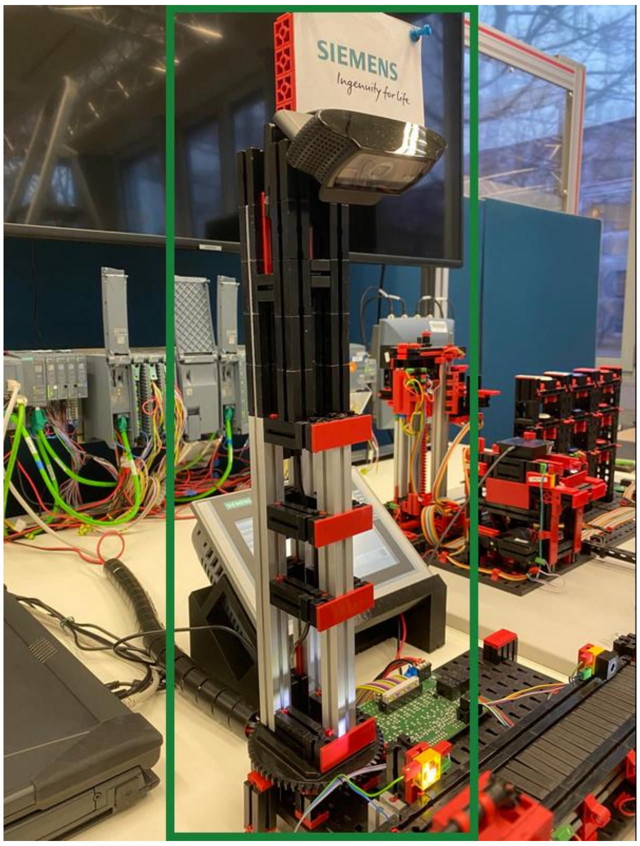


Figure 3.11: Camera setup

Workstation Configuration

The calibration software is divided into mainly four independent components that mostly run simultaneously throughout the program's execution with the help of thread-level parallelism. But if it were the opposite, which is the case of sequential flow, there would be a massive performance reduction on the computer vision algorithms and object tracking may fail. However, making use of multithreading in an application can be quite costly since all of the threads share the same resources such as CPU caches, translation lookaside buffer (TLB), and computing units. In our case, a workstation should be capable of running Process Simulate, Plant Simulation, computer vision algorithms, and rest of the software at the same time. The most demanding of them is Process Simulate and the recommended system requirements have been described as at least a quad-core processor and 4 GB of RAM for a sufficient user experience. It is followed by Plant Simulation 2D which can sufficiently operate with a dual-core processor and 2 GB of RAM. The least demanding segment of the software is the computer vision algorithms which require only a 700 MHz single-core processor and 256 MB of Ram. Therefore, the workstation should have better specifications than the given requirements so that the calibration software can run smoothly. In our research, a workstation with given specifications has been utilized:

Table 3-1: Specifications of the workstation

Component	***Specifications***
Processor	*Intel i7-6820EQ 2.80 GHz*
RAM	*16 GB 2133 MHz*
Graphics Card	*Intel HD Graphics 530*
Operating System	*Windows 10 64-bit*

3.4.2 General Software Configuration

The calibration software is entirely based on C# which is a programming language developed by Microsoft that operates on the .NET Framework. As a result, during the development of the software, Microsoft Visual Studio 2015 has been utilized as an integrated development environment (IDE). Concerning .NET, the latest version of the framework has been used which is currently .NET Framework 4.8. Besides these, the solution configuration has been selected as Debug for allowing easy debugging, and the experiments are carried out in this configuration.

Considering that the main purpose of the software is only to automatically calibrating the digital twins, there is fundamentally no need for any visual feedback back to the operator related to background processes. Be that as it may, to set up the software for the initial usage and monitor it for debugging purposes, it has been decided to implement the software with a user interface. In this context, Windows Forms has been chosen over WPF since it is less time consuming and the research is not an end-user project (Pedamkar, 2020).

There exist in total five different source codes in the project. Four of them are form files in which the necessary methods and functions of the windows are written. The organizational workflow of the form files can be seen in Figure 3.12. The details of individual form file will be explained in the next section.

Furthermore, there is also the main file of the application that contains several microservices. It is being executed firstly when the software runs. Only then the rest of the form files can be called from the main of the program. But, as it has been pointed out before, multithreading is required to avoid any delay or performance declination on the computer vision algorithms. Thereupon all of the form files have their separate threads to run the codes.

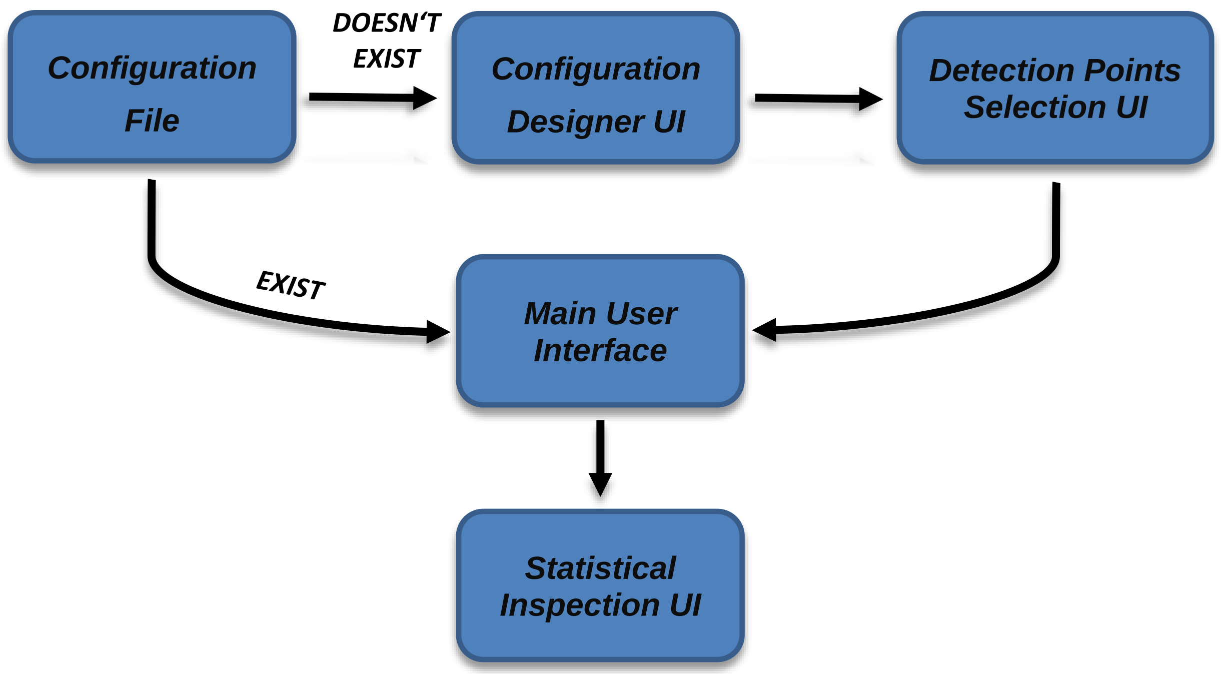

Configuration File
DOESN'T EXIST
Configuration Designer UI
Detection Points Selection UI
EXIST
Main User Interface
Statistical Inspection UI

Figure 3.12: Organizational chart of the User Interfaces

3.4.3 UI Frameworks

Overview

In the figure below, UML diagram of the user interfaces are given. As it can be seen, the main program mostly controls the core of the software. Two of the user interfaces, namely *Configuration Designer* and *Detection Points Selection,* sets the required parameters in the beginning. Other interfaces, namely *Statistical Inspection* and *Main UI*, are responsible for the interaction with the user and they behave like derived classes in terms of inheritance.

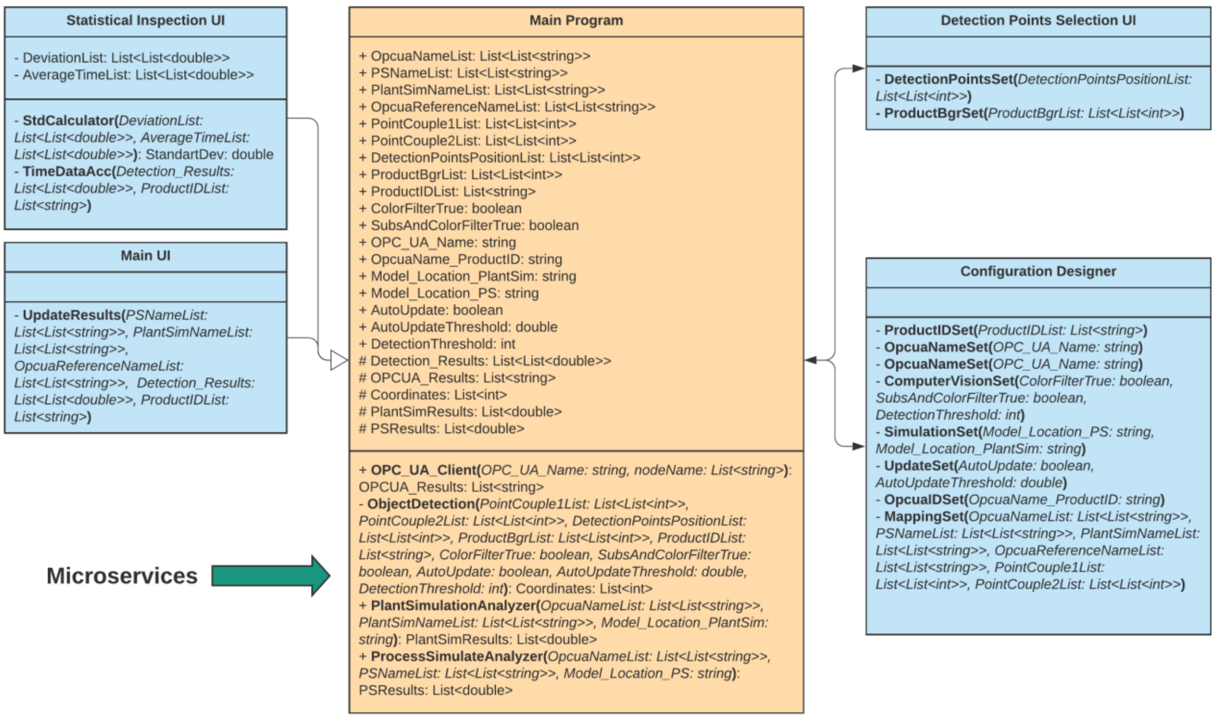

Statistical Inspection UI
- DeviationList: List<List<double>>
- AverageTimeList: List<List<double>>
- StdCalculator(DeviationList: List<List<double>>, AverageTimeList: List<List<double>>): StandartDev: double
- TimeDataAcc(Detection_Results: List<List<double>>, ProductIDList: List<string>)
Main UI
- UpdateResults(PSNameList: List<List<string>>, PlantSimNameList: List<List<string>>, OpcuaReferenceNameList: List<List<string>>, Detection_Results: List<List<double>>, ProductIDList: List<string>)
Microservices
Main Program
+ OpcuaNameList: List<List<string>>
+ PSNameList: List<List<string>>
+ PlantSimNameList: List<List<string>>
+ OpcuaReferenceNameList: List<List<string>>
+ PointCouple1List: List<List<int>>
+ PointCouple2List: List<List<int>>
+ DetectionPointsPositionList: List<List<int>>
+ ProductBgrList: List<List<int>>
+ ProductIDList: List<string>
+ ColorFilterTrue: boolean
+ SubsAndColorFilterTrue: boolean
+ OPC_UA_Name: string
+ OpcuaName_ProductID: string
+ Model_Location_PlantSim: string
+ Model_Location_PS: string
+ AutoUpdate: boolean
+ AutoUpdateThreshold: double
+ DetectionThreshold: int
# Detection_Results: List<List<double>>
# OPCUA_Results: List<string>
# Coordinates: List<int>
# PlantSimResults: List<double>
# PSResults: List<double>
+ OPC_UA_Client(OPC_UA_Name: string, nodeName: List<string>): OPCUA_Results: List<string>
- ObjectDetection(PointCouple1List: List<List<int>>, PointCouple2List: List<List<int>>, DetectionPointsPositionList: List<List<int>>, ProductBgrList: List<List<int>>, ProductIDList: List<string>, ColorFilterTrue: boolean, SubsAndColorFilterTrue: boolean, AutoUpdate: boolean, AutoUpdateThreshold: double, DetectionThreshold: int): Coordinates: List<int>
+ PlantSimulationAnalyzer(OpcuaNameList: List<List<string>>, PlantSimNameList: List<List<string>>, Model_Location_PlantSim: string): PlantSimResults: List<double>
+ ProcessSimulateAnalyzer(OpcuaNameList: List<List<string>>, PSNameList: List<List<string>>, Model_Location_PS: string): PSResults: List<double>
Detection Points Selection UI
- DetectionPointsSet(DetectionPointsPositionList: List<List<int>>)
- ProductBgrSet(ProductBgrList: List<List<int>>)
Configuration Designer
- ProductIDSet(ProductIDList: List<string>)
- OpcuaNameSet(OPC_UA_Name: string)
- OpcuaNameSet(OPC_UA_Name: string)
- ComputerVisionSet(ColorFilterTrue: boolean, SubsAndColorFilterTrue: boolean, DetectionThreshold: int)
- SimulationSet(Model_Location_PS: string, Model_Location_PlantSim: string)
- UpdateSet(AutoUpdate: boolean, AutoUpdateThreshold: double)
- OpcualDSet(OpcuaName_ProductID: string)
- MappingSet(OpcuaNameList: List<List<string>>, PSNameList: List<List<string>>, PlantSimNameList: List<List<string>>, OpcuaReferenceNameList: List<List<string>>, PointCouple1List: List<List<int>>, PointCouple2List: List<List<int>>)

Figure 3.13: UML Diagram of the User Interfaces

Configuration Designer

From the beginning of the research, a generalized approach for calibrating digital twins of production lines in a manufacturing context has been sought. However, this imposes some restrictions on the way that the software is implemented. To be able to adapt the piece of code to any manufacturing system, a systematic approach needs to be established in which the plant-specific variables are flexible. So that, a different production plant with different variables can use the same software for their applications.

The proposed solution that is given by this research is making use of a .xml type configuration file that contains the characteristics of the target plant and other simulation settings such as filter type of computer vision, workpiece detection threshold, calibration settings, and so on. The complete list can be seen from the Table 3-2.

Table 3-2: Content of the Configuration File

Settings Name	**Description**
OpcuaNameList	2D string-type list containing OPC UA destinations for data retrieval
PSNameList	2D string-type list containing Process Simulate operation names
PlantSimNameList	2D string-type list containing Plant Simulation operation names
OpcuaReferenceNameList	2D string-type list containing OPC UA reference destinations for data retrieval
PointCouple1List	2D integer-type list containing order of first point couples
PointCouple2List	2D integer-type list containing order of second point couples
DetectionPointsPositionList	2D integer-type list containing coordinates of detection points
ProductBgrList	2D integer-type list containing color codes of products
ProductIDList	1D string-type list containing IDs of products
ColorFilterTrue	Boolean-type variable setting Color Filter mode
SubsAndColorFilterTrue	Boolean-type variable setting Subtraction&Color Filter mode
OPC_UA_Name	String-type variable containing OPC UA address
OpcuaName_ProductID	String-type variable containing OPC UA destination of current product
Model_Location_PlantSim	String-type variable defining location of Plant Simulation model
Model_Location_PS	String-type variable defining location of Process Simulate model
AutoUpdate	Boolean-type variable setting Auto Update mode
AutoUpdateThreshold	Double-type variable setting threshold limit for Auto Update mode
DetectionThreshold	Integer-type variable setting detection threshold for object detection

In practice, there are two different approaches to construct the configuration file. Either the user manually fills up the .xml file with a given template or employ the *Configuration Designer* which can automatically create and save the configuration file depending on the preferences of the user. After compiling and running the software, it is first asked the user whether a new configuration file needs to be built or not. If a configuration file has been constructed before, the *File Dialog* process runs. It displays a dialog box from which the user can browse and select the location of a configuration file. Then a new function called *ConfigurationFileReader* initializes with the location of the configuration file and reads it using *System.Xml*. On the other hand, if a new configuration file is preferred, only then starts the *Configuration Designer*.

To begin with, the *Configuration Designer* asks the user regarding the product information in the plant such as the number of different types of products and their IDs defined in the automation. The reason why product IDs are vital is that two distinctive products may have the same exterior appearance and computer vision algorithms would fail to detect the distinction between them. Following that, OPC UA information is being demanded from the user which will later be used to obtain varying field data. Next comes the additional settings, in which the user can choose the desired filter type of computer vision, update, and digital twin settings. There are two update modes, namely *Auto-Update* and *Debug*. For development and diagnostics purposes, *Debug* mode has been implemented which disables calibration of digital twins in case it is selected. Contrastingly, *Auto-Update* mode enables automatic calibration of digital twins in case *Update Threshold(sec)* value is crossed which can be specified as the time duration difference of the same material flow operation occurring in the real plant and the digital twin. This value is also requested from the user if *Auto-Update* mode is selected. Regarding the settings of digital twins, the software allows users to choose a calibration framework between Plant Simulation and Process Simulate, or alternatively both. Afterward, the simulation files of the selected frameworks are acquired by a file dialog window as well. The user interface of this stage is shown in the figure below.

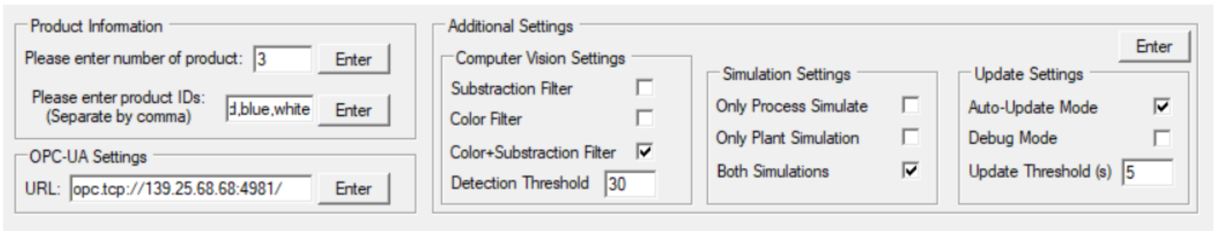
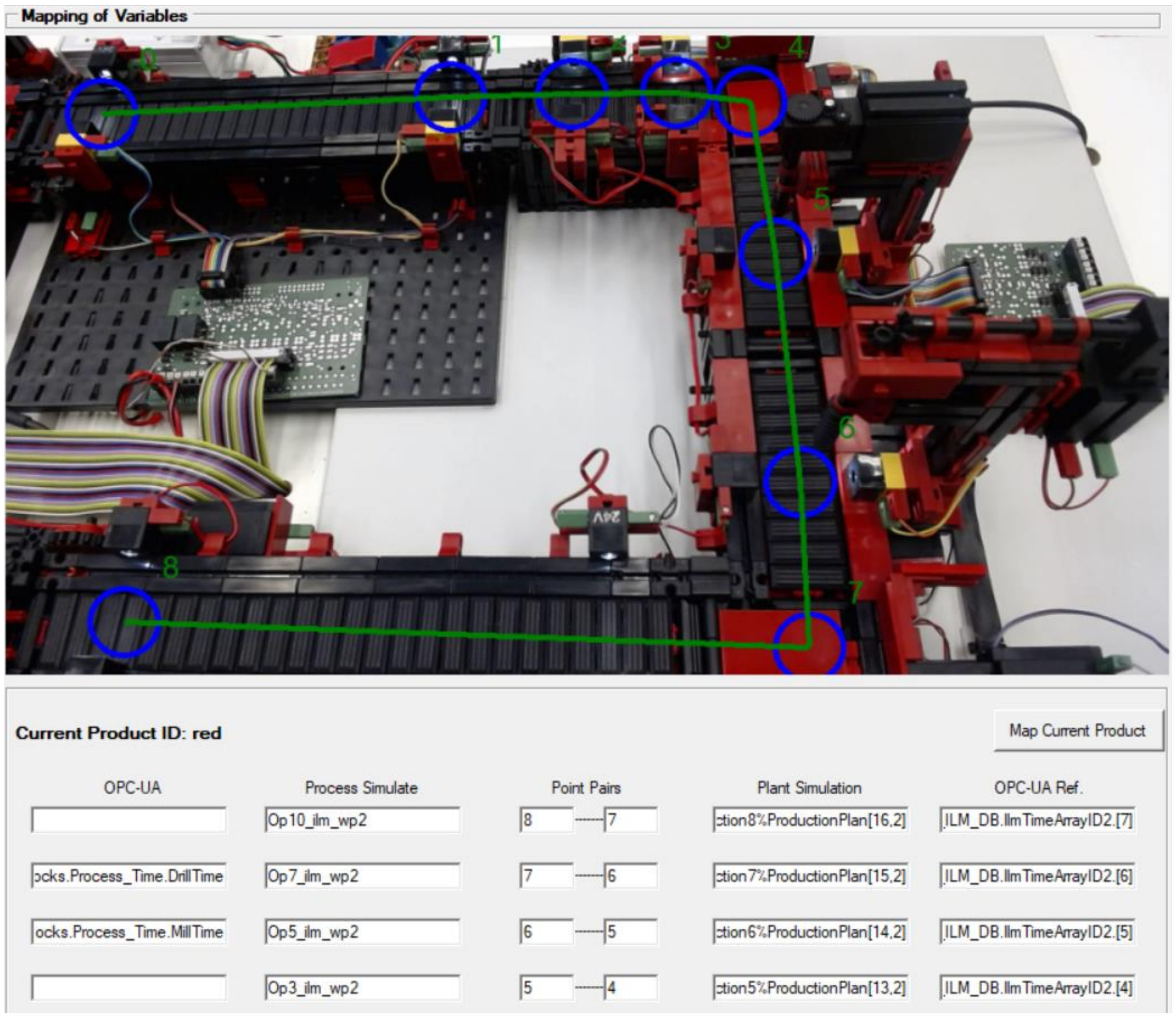

Product Information
Please enter number of product: 3
Please enter product IDs: (Separate by comma) d,blue,white
OPC-UA Settings
URL: opc.tcp://139.25.68.68:4981/
Additional Settings
Computer Vision Settings
Substraction Filter ☐
Color Filter ☐
Color+Substraction Filter ✓
Detection Threshold 30
Simulation Settings
Only Process Simulate ☐
Only Plant Simulation ☐
Both Simulations ✓
Update Settings
Auto-Update Mode ✓
Debug Mode ☐
Update Threshold (s) 5

Figure 3.14: First stage of the Configuration Designer

From here on, the second stage of *Configuration Designer* becomes available which is solely based on mapping of the variables and generation of the configuration file. The general outlook of the second stage is given in Figure 3.15. Before this stage, the *Detection Points Selection* UI is activated in which detection locations of workpieces have been registered as points. Workpieces move from one point to another in a real plant and this is denoted as *Point Pairs* in *Configuration Designer*. Each of these pairs corresponds to an operation in the digital twins and they must be mapped manually by the user. For Process Simulate, operation names should be entered and for Plant Simulation, either a table or component name should be provided. To ease up this process, selected detection locations are projected onto the image of the plant and as the user adds point pairs, a line is drawn between the selected points that is achieved by *System.Drawing* class. In addition to them, the user has the option to add a variable from the OPC UA server if any variable is causing an excessive calculated time due to a stationary operation. Lastly, there is an additional OPC UA implementation in which the user can enter a variable from OPC UA as a reference result to evaluate the performance of the calibration software. Unfortunately, all of these steps must be repeated for each different type of product which is defined at the beginning of the *Configuration Designer*. When everything is completed, it is asked to the user whether the current configuration wants to be saved or not. If it is permitted by the user, an XML document is going to be created using *System.XML* which is going to be saved on a desired location in the computer.



Current Product ID: red

OPC-UA	Process Simulate	Point Pairs		Plant Simulation	OPC-UA Ref.
	Op10_ilm_wp2	8	7	ction8%ProductionPlan[16,2]	ILM_DB.IlmTimeArrayID2.[7]
ocks.Process_Time.DrillTime	Op7_ilm_wp2	7	6	ction7%ProductionPlan[15,2]	ILM_DB.IlmTimeArrayID2.[6]
ocks.Process_Time.MillTime	Op5_ilm_wp2	6	5	ction6%ProductionPlan[14,2]	ILM_DB.IlmTimeArrayID2.[5]
	Op3_ilm_wp2	5	4	ction5%ProductionPlan[13,2]	ILM_DB.IlmTimeArrayID2.[4]

Figure 3.15: Second stage of the Configuration Designer

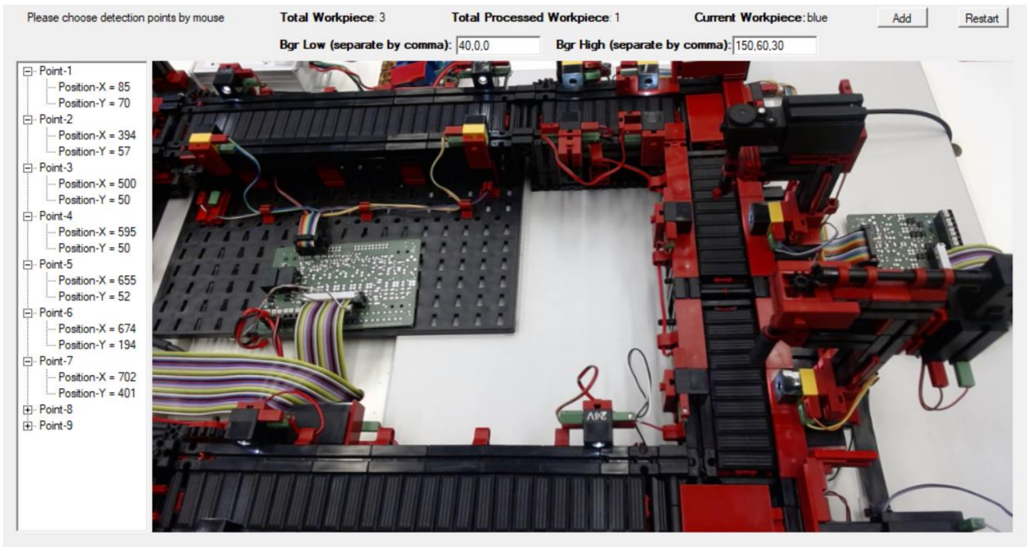
Detection Points Selection

This is also a very useful user interface to assist the user in generating the configuration file. The design intention is to configure the detection points which will be then compared with the results coming from the computer vision codes. If an algorithm detects that the workpiece is very close to a detection point, then a timer starts until the workpiece reaches to another detection point in the list and consequently total time spent between two point pairs can be calculated.

The layout of *Detection Points Selection* mainly features two sub-controls which are a *TreeView* and a *PictureBox*. The image obtained by the camera setup is presented in the *PictureBox*. Along with it, the mechanism of adding detection points utilizes *Click-Event* of the *PictureBox*. Once it is disposed of, the event creates a function that has *MouseEventArgs* as an input. From there on, whenever the user clicks somewhere on the *PictureBox*, the function that contains the click location of the mouse triggers.

Afterward, by inspecting the image, the user would decide on the detection points of the corresponding workpiece. Each mouse click on the *PictureBox* produces a new detection point that can be found under the *TreeView*. X-coordinate and Y-coordinate of every selected point are presented there as well for debugging purposes. If all of the detection points are selected successfully, the user can press *Add* to save the configuration of the selected workpiece and continue with the other ones. In case that all of the workpieces have the same detection points, then pressing *Set Same Points To Rest* will assign the same points for the rest. There is also an option called *Restart* that would reset the current configuration and start over. A sample image of this interface is presented in Figure 3.16.

From this window, it is also possible to assign the colors of the workpieces for the computer vision algorithms. There are two *TextBoxes* dedicated to the task which are called *Bgr Low* and *Bgr High*. Details of them will be explained in Computer Vision Methods.


Please choose detection points by mouse
Total Workpiece: 3
Total Processed Workpiece: 1
Current Workpiece: blue
Bgr Low (separate by comma): 40,0,0
Bgr High (separate by comma): 150,60,30

```
⊟ Point-1
    Position-X = 85
    Position-Y = 70
⊟ Point-2
    Position-X = 394
    Position-Y = 57
⊟ Point-3
    Position-X = 500
    Position-Y = 50
⊟ Point-4
    Position-X = 595
    Position-Y = 50
⊟ Point-5
    Position-X = 655
    Position-Y = 52
⊟ Point-6
    Position-X = 674
    Position-Y = 194
⊟ Point-7
    Position-X = 702
    Position-Y = 401
⊞ Point-8
⊞ Point-9
```

Figure 3.16: Sample run of the Detection Points Selection UI

Main User Interface

If the configuration file is successfully imported and the connection to the OPC UA server is validated by the main program, a signal has been set to initialize the components of the *Main User Interface*.

This is the main window that the user mostly interacts with. It has basically three objectives which are displaying results, monitoring critical variables via a *Watch Table*, and controlling some additional program parameters.

These additional program parameters include the commands of pausing, continuing, and exiting. Just in case that the user wants to pause the automatic calibration functions and object detection algorithms, it is possible to take advantage of the *Pause* button. This is a much faster solution than closing the program and reconfiguring every variable from the beginning since there is also an integrated button for continuing the automatic calibration process. At the last, a button for exiting from the calibration software is provided too which does not only close the program but also clear some unwanted leftovers of Plant Simulation which cannot be released from the memory using the garbage collector. More details are explained in Plant Simulation Analyzer.

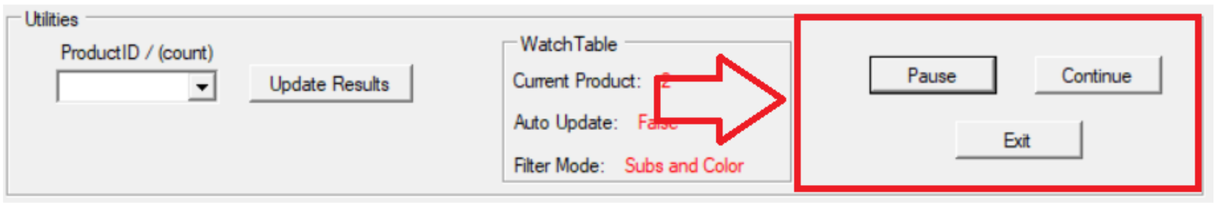
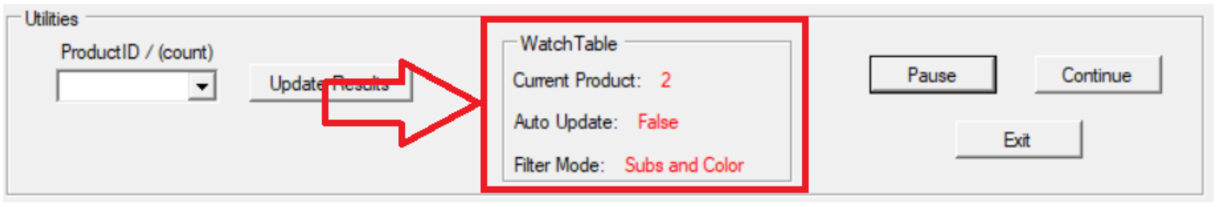


Figure 3.17: First objective of the Main UI / Controlling program parameters

The second objective of the form is to inform the user about some of the critical parameters in the software, namely *Current Product*, *Auto Update* and *Filter Mode*. They are defined as *Label* type objects in the user interface and changing the text of them is the only method to update the observable value. *Filter Mode* describes the type of the selected filter mode used in the computer vision algorithms. *Auto Update* shows what kind of update method is chosen for the application. It becomes either *True* or *False* to denote the mode. Perhaps the most essential of all is *Current Product* that notifies the user about the latest product which is presently processing in the plant for debugging purposes. A timer has been implemented which connects to OPC UA server to extract the information each second so that the user can effectively identify the current product in the plant.

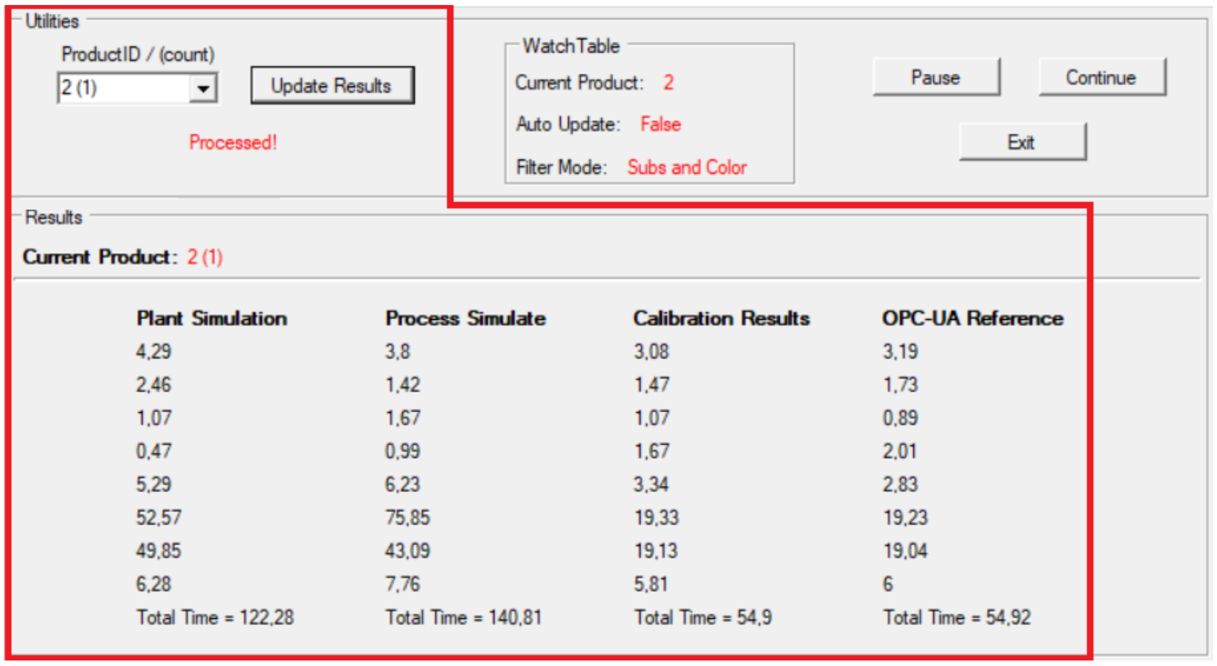


Current Product: 2
Auto Update: False
Filter Mode: Subs and Color

Figure 3.18: Second objective of the Main UI / Monitoring critical variables via Watch Table

The third objective of the user interface is to display the results acquired from the detection algorithms. Firstly, a *comboBox* is constructed which is initialized as empty.

Once the object detection algorithms process a workpiece and gather the time results of it, the product ID and count of the workpiece are added to the *comboBox* to avoid a selection of unprocessed products. Furthermore, when the user selects a product from the *comboBox* and presses the *Update Results* button, a new thread is invoked to process all of the time-related data of the chosen product. Otherwise, the *main UI* would be frozen during that time. Besides that, the results coming from the thread are displayed in four columns as labels which specifically belong to Plant Simulation, Process Simulate, Calibration Results, and OPC UA Reference. First of all, the product information is sent to Plant Simulation Analyzer and Process Simulate Analyzer functions in read-only mode. Time values are extracted from these digital twins and sent back to the thread to display the results. After that, calibration and reference results are also collected from the main program and OPC UA server respectively. By summing the results in each column, the total time estimation in these frameworks can be visualized. In the circumstances that the user wants to view another product's result, the current labels become hidden and new labels are constructed on top of the previous ones. This method is very beneficial since it is enough to make the labels visible again if the user wants to view the results of the previously selected product again. As a final remark, the columns of results are allocated in a *flowLayoutPanel* and the size of the panel is determined dynamically to be able to fit any number of variables inside the window.



2 (1)

Processed!

Current Product: 2

Auto Update: False

Filter Mode: Subs and Color

Current Product: 2 (1)

Plant Simulation	Process Simulate	Calibration Results	OPC-UA Reference
4,29	3,8	3,08	3,19
2,46	1,42	1,47	1,73
1,07	1,67	1,07	0,89
0,47	0,99	1,67	2,01
5,29	6,23	3,34	2,83
52,57	75,85	19,33	19,23
49,85	43,09	19,13	19,04
6,28	7,76	5,81	6
Total Time = 122,28	Total Time = 140,81	Total Time = 54,9	Total Time = 54,92

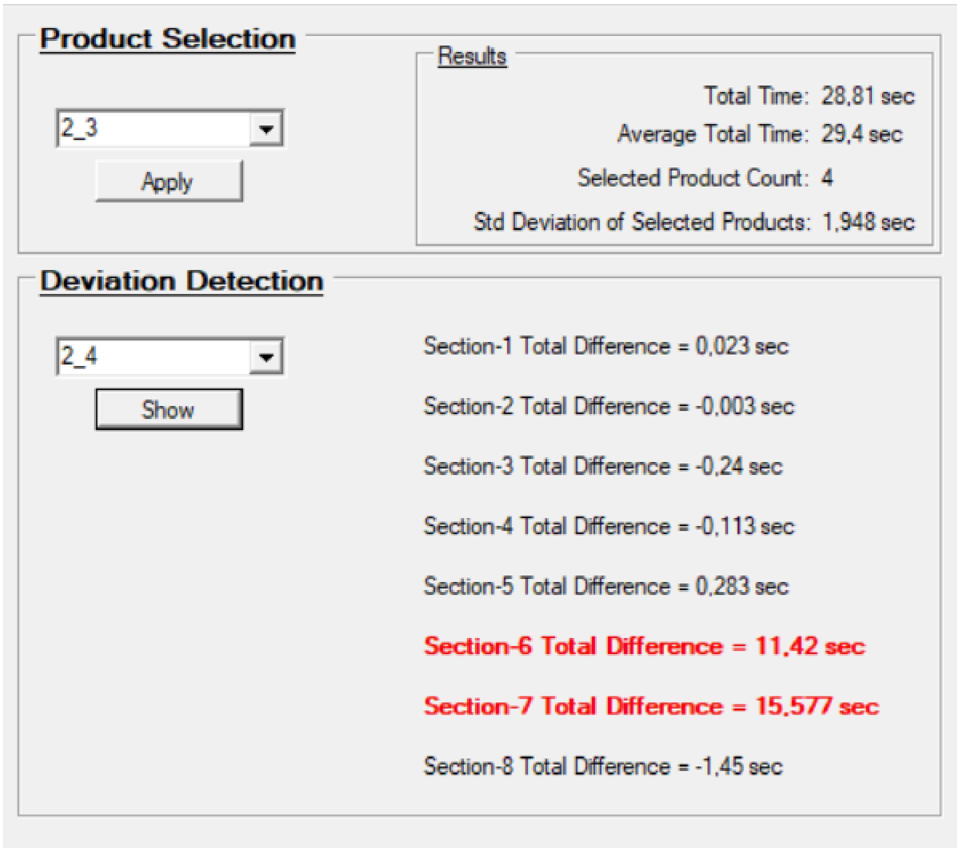
Figure 3.19: Third objective of the Main UI / Displaying results

Statistical Inspection

Statistical Inspection is the last user interface in the calibration software. It is designed to make basic statistical deductions and analyses based on the previous results of the products. Like the configuration file, previous results are also stored in a .xml type file named FormerResults.xml and similar with the other user interfaces, this also has its own thread for the execution.

At the beginning of the initialization, a function checks whether there is an already existing previous results file or not. Depending on that, the creation or reading of the existing file takes place.

As the main program and computer vision algorithms process the data, new results are gathering in the lists. To check whether there is a new result or not, a timer has been implemented in the Statistical Inspection which operates every 10 seconds to collect the latest data from the main program. Following that, when a new result is detected, the deviation of each timing element compared to the average is calculated. If the deviation does not surpass the limit, then the timing data of the current product is saved in FormerResults.xml and statistical analyses such as total processing time, average time, standard deviation can be carried out in the Product Selection section by entering the product ID. On the other hand, if the deviation exceeds the tolerance level, then an alarm is triggered which may indicate an unexpected situation that the user should be aware of. In this case, the results are not saved in FormerResults.xml to preserve the reliable data. Also, the user has the option to view the unreliable data from the Deviation Detection section for a review. Figure 3.20 shows a sample run of this user interface.



Results

Total Time: 28,81 sec
Average Total Time: 29,4 sec
Selected Product Count: 4
Std Deviation of Selected Products: 1,948 sec

Section-1 Total Difference = 0,023 sec
Section-2 Total Difference = -0,003 sec
Section-3 Total Difference = -0,24 sec
Section-4 Total Difference = -0,113 sec
Section-5 Total Difference = 0,283 sec
Section-6 Total Difference = 11,42 sec
Section-7 Total Difference = 15,577 sec
Section-8 Total Difference = -1,45 sec

Figure 3.20: Sample run of the Statistical Inspection UI

3.4.4 Microservices

OPC UA Client

The calibration software requires field data to perform properly without any doubt. This can be provided via an OPC UA interface that sort of connects PLCs with the calibration software.

To be able to build such an interaction in C# environment, .NET-based OPC UA Client/Server SDK has been adapted. The given package consists of three libraries, namely C# Base, Server, and Client Library. In this research, the OPC UA server is supported by commercial software, meaning that only UnifiedAutomation.UaBase and UnifiedAutomation.UaClient are added to the references.

Many of the OPC UA related operations come to ease with this library. First of all, only two variables are required to make a successful connection with the server. These can be specified as a session variable and a string type address name. An integrated *Connect* function, that takes these variables as inputs, can be called to create a secure channel to the OPC UA address and activate a session in the server application. Although it is not compulsory, there is also an additional *Disconnect* function that closes the session and the secure channel connection. When closing the software, this function is called to increase the reliability of potential connections in the future.

The handiest functions in this library are *Read* and *Write* functions. Since it is not the aim of this research to change the variables in a plant, *Write* function is absent in the software though *Read* function is commonly used throughout the procedures. The *Read* service is used to read one or more attribute values synchronously from the OPC server. From the design, it is optimized for bulk read operations instead of reading a single value. As a result, *Read* function takes a list of *node IDs* to read which should be parsed using an integrated function, and returns a list of type *DataValue* which should be then converted to a common data type before the usage.

Computer Vision Methods & Object Detection

The computer vision algorithms are the backbone of the calibration software and make up the majority of the codes that are implemented in the main program. They are going to be described briefly in this chapter.

The choice of the computer vision algorithm was in the favor of Emgu CV which is a cross-platform .NET wrapper that contains the image processing libraries of OpenCV. It can be downloaded and quickly installed with the help of NuGet, an open-source package manager designed to share code in .NET framework. In this research, version 3.2.0.2721 of Emgu CV has been employed. In addition to that, two references are added, namely Emgu.CV.UI and Emgu.CV.World, for a basic-level computer vision and graphical interaction.

Initially, a new dynamically allocated variable is created using *Emgu.CV.VideoCapture* command. It is a function to create a capture variable using a specific camera linked to the computer. If the camera is opened successfully, then the camera capture settings are modified in such a way that the workpiece detection is more robust and reliable. First of all, the resolution is set to 480p for a smoother image capturing along with saturation, brightness, and contrast values which are also adjusted to minimize reflections and increase the vividness of the colors. The changes can be reviewed from Figure 3.21. Following that, an image frame with the edited attributes is captured. As a color format, Emgu CV utilizes a 24-bit representation called BGR. Individual color codes can take values between 0 and 255, in a similar fashion to the RGB color scheme which is the most used type.

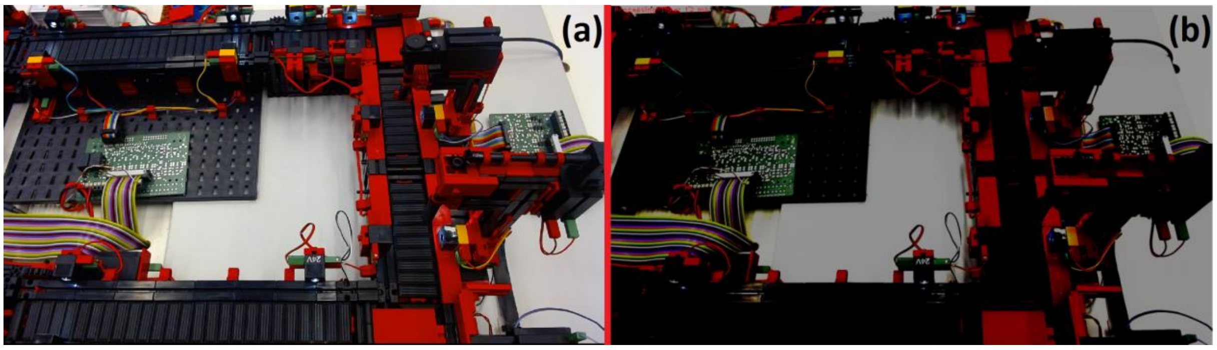


Figure 3.21: (a) Unprocessed Captured Frame, (b) Initial Adjustments on the Captured Frame

Subsequent to the frame capturing, image processing begins with a *StopWatch* which is going to be used later to measure the object detection's performance. In total, there are three different methods to process the frame, namely *Subtraction*, *Color*, *Subtraction&Color Filter*.

In the *Subtraction* mode, *AbsDiff* function is used which takes two successive frames as inputs and calculates the absolute difference between them. In this way, it is possible to eliminate non-moving components in a plant and focus on a workpiece.

In the *Color Filter* mode, before anything else, the frame is converted to a BGR type image, and *Gaussian Smoothing* is performed on it. Then a *mask* is constructed which takes *low* and *high BGR* values to define a range. Finally, the *SetValue* function sets every pixel of the image to a specific color, using the *mask*. The pixels, that are not in range of the *mask*, consequently blackened. An example is demonstrated in the Figure 3.22 below.

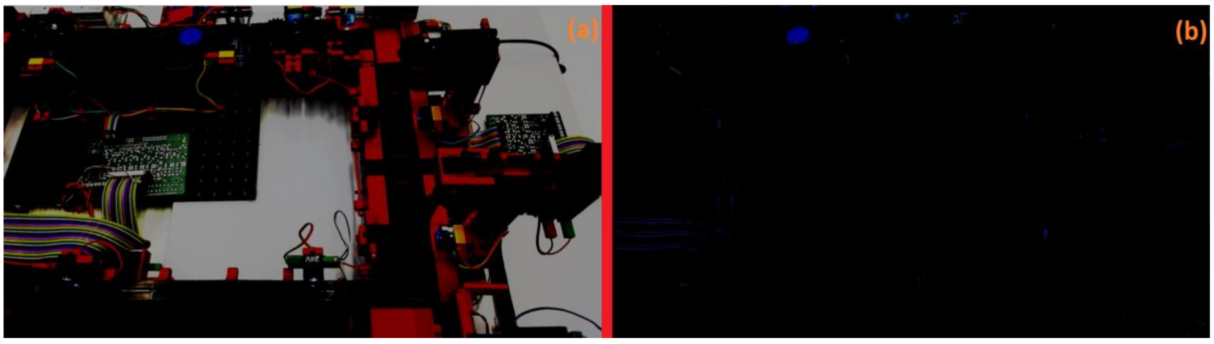


Figure 3.22: (a) Adjusted Captured Frame, (b) Blue-type Mask Implementation

In the *Subtraction&Color Filter* mode, two successive frames are both subjected to the color filter process before being subtracted from each other.

When the given methods are processed, *CvtColor* function is applied to create a gray-scale image which is then treated with the *Threshold* function to apply a binary threshold, as it can be seen in the figure below.

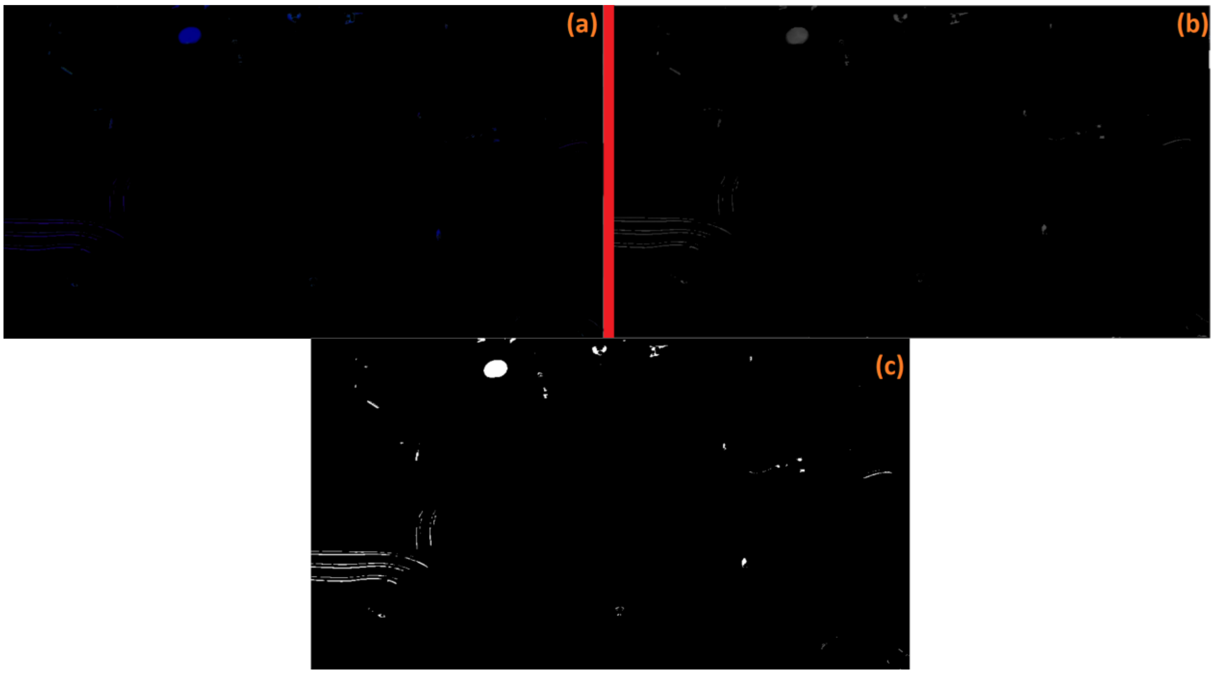


Figure 3.23: (a) Color Filtered Frame, (b) Grayscaled Frame, (c) Binary Frame

Through the end, *Erode* and *Dilate* functions are used one after another to remove the noise and enrich what is left. Figure 3.24 shows an example the processes of removing noise and dilation of the survivors.

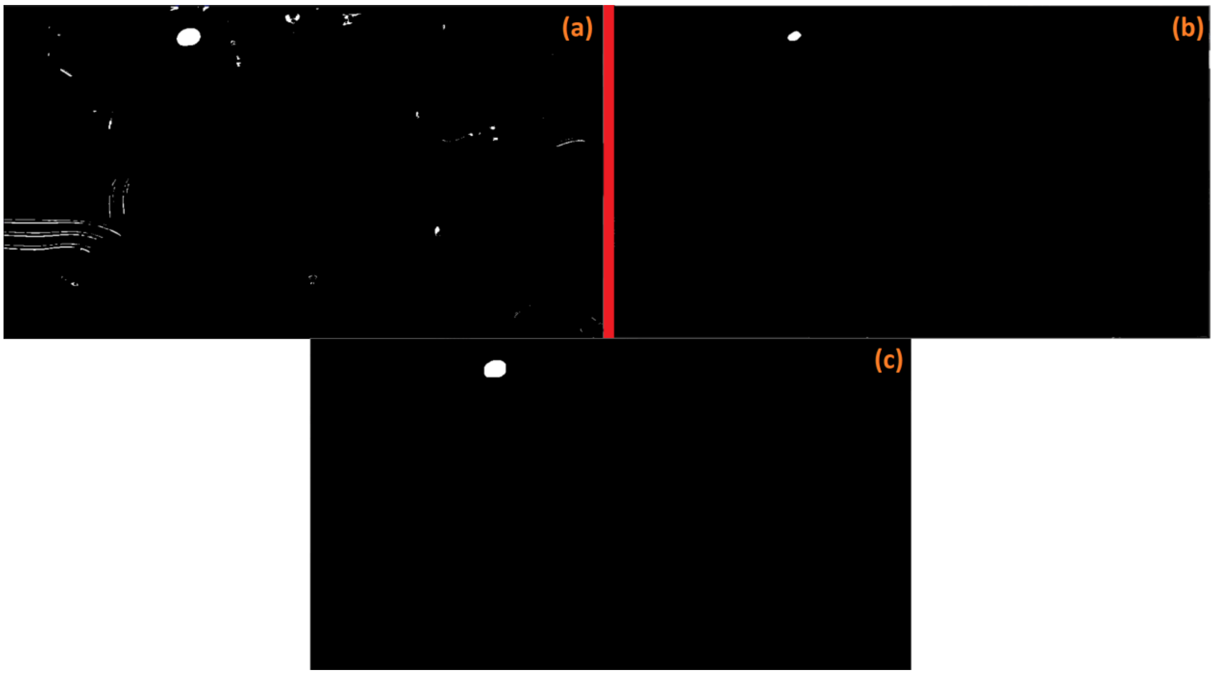


Figure 3.24: (a) Binary Frame, (b) Denoised Frame, (c) Dilated Frame

General frame modifications in the *Subtraction&Color Filter* mode are summarized in the Figure 3.25. Also, all of the different stages of image processing can be shown in windows by using *CvInvoke.Imshow* function.

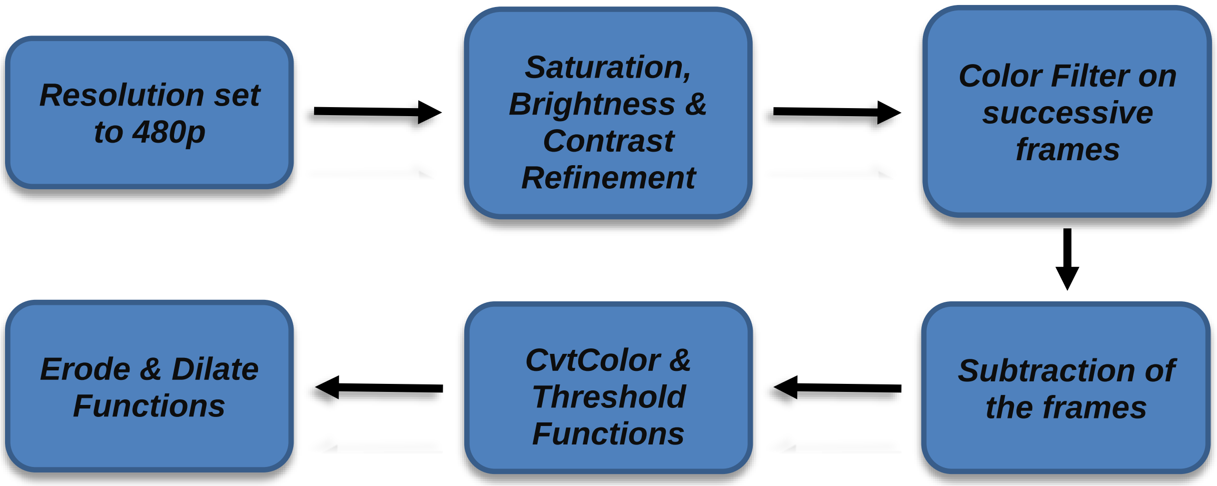

Resolution set to 480p
Saturation, Brightness & Contrast Refinement
Color Filter on successive frames
Subtraction of the frames
CvtColor & Threshold Functions
Erode & Dilate Functions

Figure 3.25: Computer vision algorithms in Subtraction&Color Filter

The final image consists of white pixels that denote the location of the object. However, previous filters are sometimes not enough to completely remove the background. Therefore, *CvInvoke.FindContours* is initially used to retrieve the contours of the remaining pixels. Afterward, the contour that has the biggest area is found which indicates the position of the object. The figure below summarizes the main steps of obtaining the location of the products, which are adjusting the initial frame, filtering, and calculating the biggest contour.

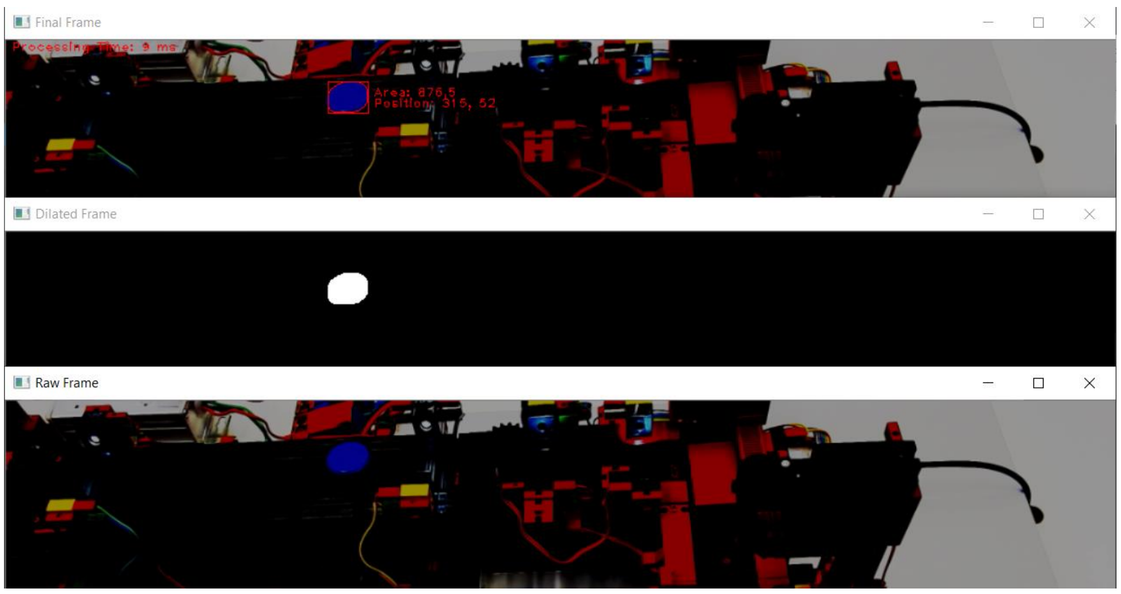


Figure 3.26: Summary of the steps that lead to the position of an object

The rest of the algorithm is straightforward. Once an object is detected, X and Y coordinates of the object center are extracted and compared with each point in the *Detection Points* list. In case the distance between them is smaller than a detection threshold, a timer begins until the workpiece is close to another point. When all the points in the *Detection Points* list are used, timing results and workpiece specifications are sent to *Plant Simulation Analyzer* and *Process Simulate Analyzer*. This is how the time results of workpiece motion are acquired with the help of computer vision algorithms.

Plant Simulation Analyzer

Tecnomatix Plant Simulation offers a dll-type file for C# development. It can be found under the installation folder, namely PlantSimCore.dll. When this file is added to Visual Studio project as a reference, a library that is called eMPlantLib becomes available to get COM reference working. The library provides *RemoteControl* method which can control most aspects related to a Plant Simulation model. The most used ones in this research are *GetValue* and *SetValue*. As the names suggest, they are used to get and set any variable in the model.

The analyzer function firstly loads a model described by its location. Following that, a mode of the analyzer is activated. In the first mode, based on the information in the configuration file, time values of the workpiece are extracted from the model. For this purpose, *GetValue* function is used. This is the case when the user runs the calibration software in *Debug* mode and wants to assess the performance of object detection. Otherwise, in the second mode, the results are directly assigned to the corresponding variables in Plant Simulation model. At the end of each update step, the model is saved to prevent any data loss.

Process Simulate Analyzer

The analyzer methodology of a Process Simulate model is completely different than what is implemented for Plant Simulation. Since opening and loading of a Process Simulate model takes some time, the analyzer works offline and directly alters the simulation file.

To start with, the simulation file is a zip-type document that contains many files inside. However, specific interest is in the file called StandaloneStudy_PsState.xml. Most of the simulation-based parameters are stored in that file, such as operation times, 3D data, IDs, and coordinates of the components. What we are interested in is the operation times for every workpiece. To that end, the *System.IO.Compression* library has been used to extract the target file and delete it afterward.

Like the Plant Simulation Analyzer, two modes are defined in the analyzer which are *Reading* and *Setting*. Once StandaloneStudy_PsState.xml file is acquired, the *nodes* of the file are read one by one to find the operation name specified in the configuration file. Then, depending on the mode, *InnerXml* of the *node* is either read or changed.

In the end, the .xml file is saved and added back to the zip file. Here, one must be careful and should not forget to *Dispose* the zip variable. Otherwise, the file is not updated in Windows Explorer.

4 Experiments and Results

4.1 Performance of the Object Detection Methods

The performance of varying object detection methods, which have been utilized in the calibration software, is presented in this section. In total, three methods have been evaluated. These are called Subtraction, Color Filter, and Subtraction&Color Filter. Speed, accuracy, and flexibility are the main assessment parameters in the experiments. Also, to avoid any error related to environmental conditions, several experiments were conducted consecutively.

4.1.1 Frame Processing Rate

One of the most important decision criteria to select an object detection method is the amount of time that is needed to process a frame. If an application consists of fast-moving objects which have to be captured relatively quickly, then the object detection algorithms need to process frames rapidly. However, many factors affect the frame processing rate, and they have to be addressed individually. These factors include algorithm selection, multithreading, environmental conditions, computer specifications, and camera type. For example, more advanced computer vision algorithms such as Faster R-CNN or YOLO have proved to be quicker and more efficient detection methods. Nonetheless, relatively simple requirements of the model plant have led us to Subtraction and Color Filter Methods.

Subtraction Method – Frame Rate

Subtraction is the simplest method implemented in this research. It finds the difference between two consecutive frames by subtracting each other and the algorithm ends. Therefore, it is expected that the given method consumes the least resource in terms of processing power.

An experiment is conducted to approximate the speed of the Subtraction Method. During a sample object detection analysis, 30 instances, that are separated by 0.5 seconds, were selected and the amount of time that took to process each frame was recorded. Figure 4.1 shows the results. Also, it should be noted that the given numbers only represent the frame processing section of the calibration software.

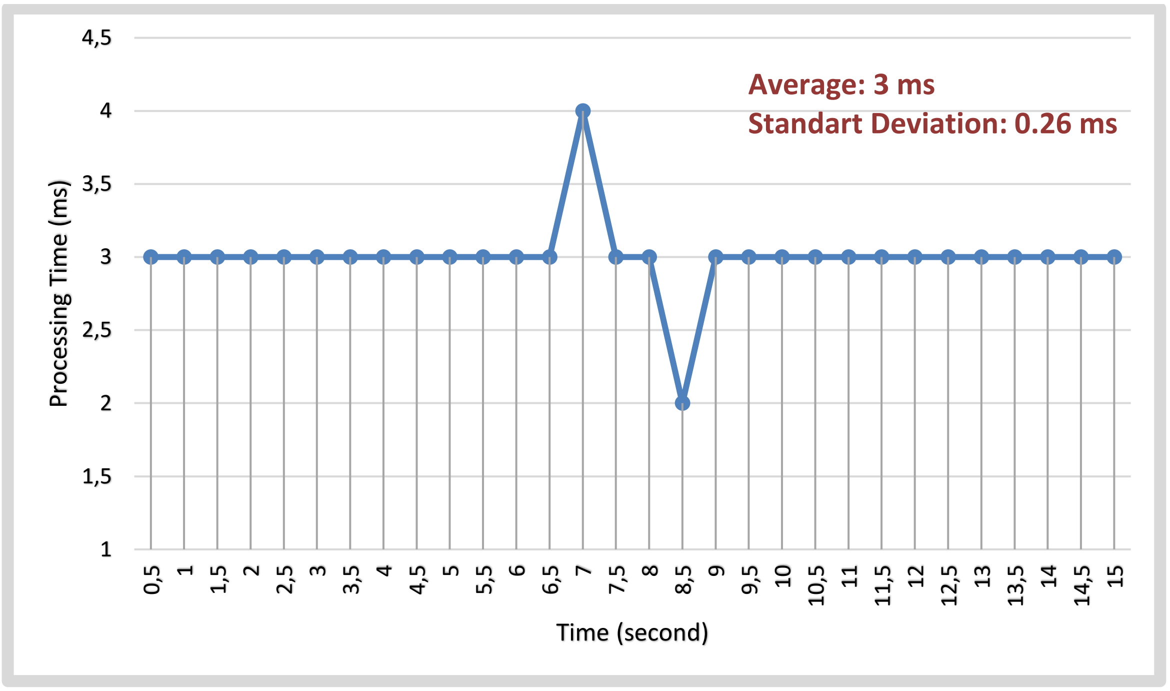


Figure 4.1: Frame Processing Rate – Subtraction Method

In this specific case, the average processing time is turned out to be around 3 ms. It means that, theoretically, 333 frames can be processed each second if there were not any limitations on the side of the camera system.

Color Filter Method – Frame Rate

The second method, that has been frequently used in this research, is called Color Filter Method which firstly captures a frame. Then a mask is applied to eliminate unwanted colors from the frame. This is a very efficient method if an object has a distinctive color compared to the background.

The same experiment was conducted for the Color Filter Method as well to obtain some figures for the frame processing rate. The figure below explains the frame processing results of the Color Filter Method.

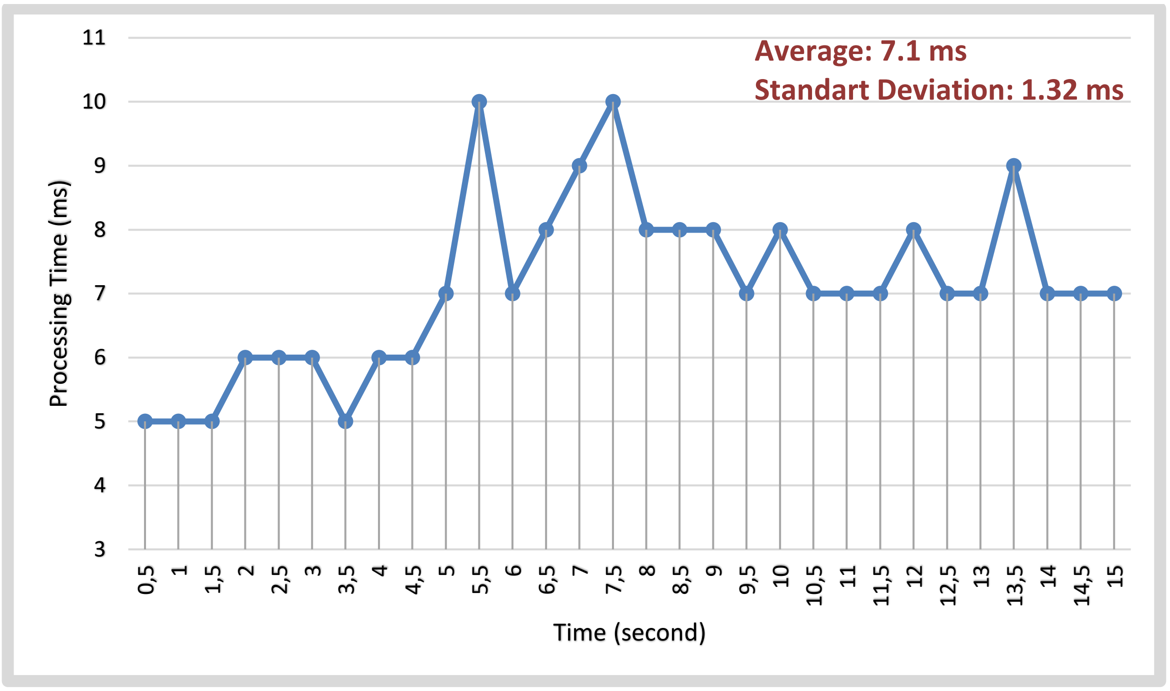


Figure 4.2: Frame Processing Rate – Color Filter Method

There are two things to mention about the results. Firstly, at the beginning of the experiment, there is not any object to detect. As the object enters the vision of the camera around "*t = 5 seconds* ", we can see that the processing times increase drastically and unpredictably. This proves the unstable nature of the mask algorithm which mainly depends on the size of the pixels that are in the range of the mask.

Secondly, the results suggest that Color Filter Method is more than two times slower than the Subtraction Method if we compare their average values. However, it is still quite acceptable to process around 140 frames per second for most computer vision applications.

Subtraction&Color Filter Method – Frame Rate

The third object detection method is called Subtraction&Color Filter Method and as the name suggests, it is a combination of Subtraction and Color Filter methods. The design aim was to benefit from the advantageous aspects of both methods. A mask is first being applied to the consecutive frames and then they are subtracted from each other. The test of the Subtraction&Color Filter can be seen in the figure below.

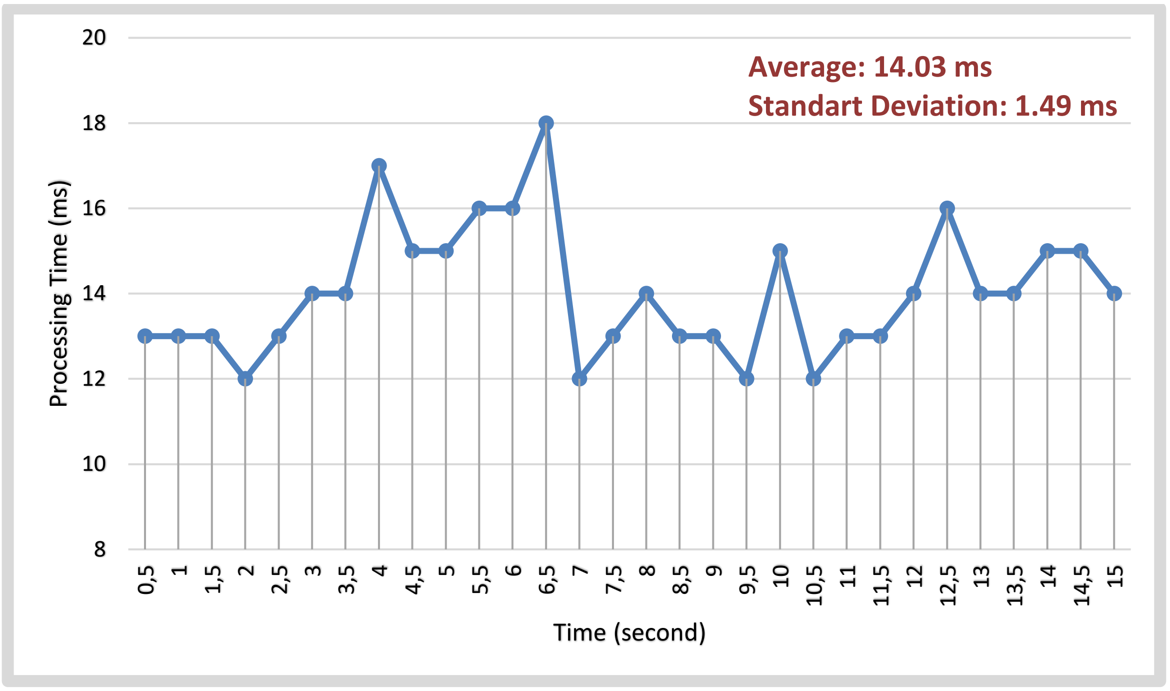


Figure 4.3: Frame Processing Rate – Subtraction&Color Filter Method

The average time required for the Subtraction&Color Filter to process a frame is much higher than the sum of Subtraction and Color Filter. This is due to an additional second frame that needs to be masked. Besides that, the characteristics of the Subtraction&Color Filter are very similar to Color Filter. However, processed frames per second dropped to around 71 frames and with some additional coding, the number is going to drop even more. Thus, the method should be preferred when there is an absolute need for its capabilities.

4.1.2 Accuracy of Object Detection Methods

Accuracy is also a very important subject in the context of object detection. It is always preferred to have a very accurate detection if the processing cost can be compensated. In this section, experiments are carried out to determine the accuracy level of the object detection methods in this research.

In total, three different object positions are selected, and the locations of the object center are acquired. Then the object detection methods are executed to obtain the results. Comparing the actual positions of the center and the results coming from the algorithms is proposed to give us some ideas about the accuracy levels.

Subtraction Method – Accuracy

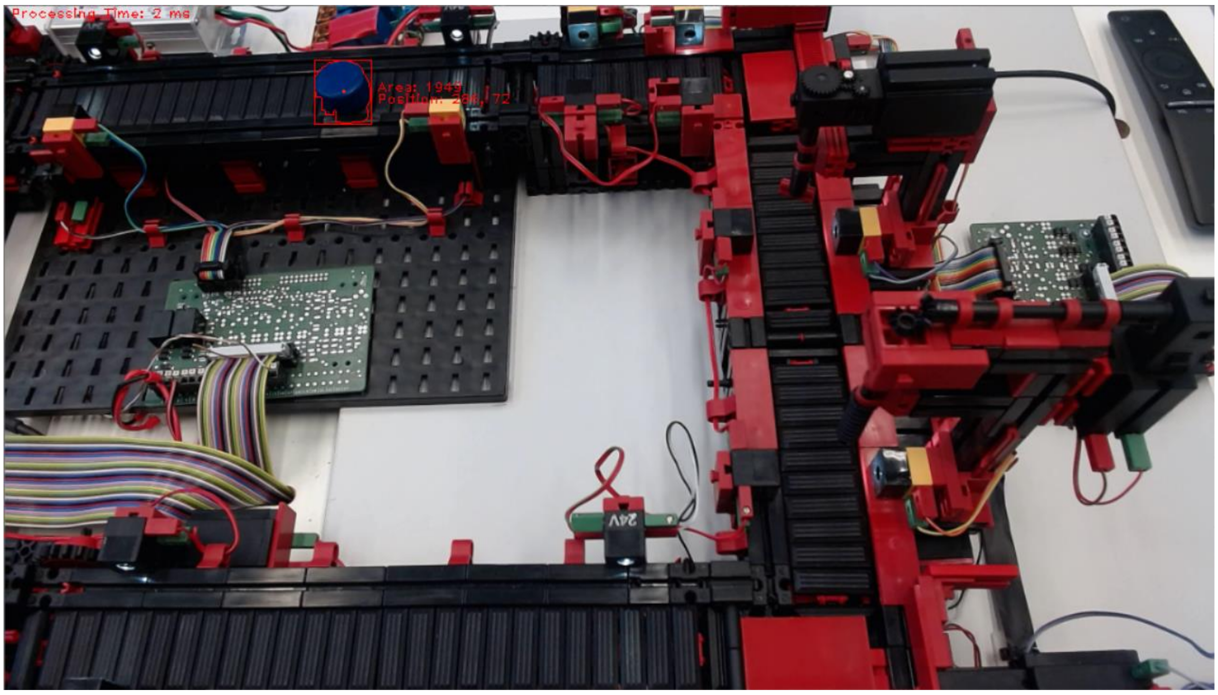


Figure 4.4: Subtraction Method – Accuracy / Position-1

The figure above shows the object detection performance of the Subtraction Method at the first position. It can be seen that the algorithm picks up some residual areas that do not belong to the object. The center of the object is detected at the pixel-coordinates of (286,72) and the total error is estimated to be around 9.21 pixels.

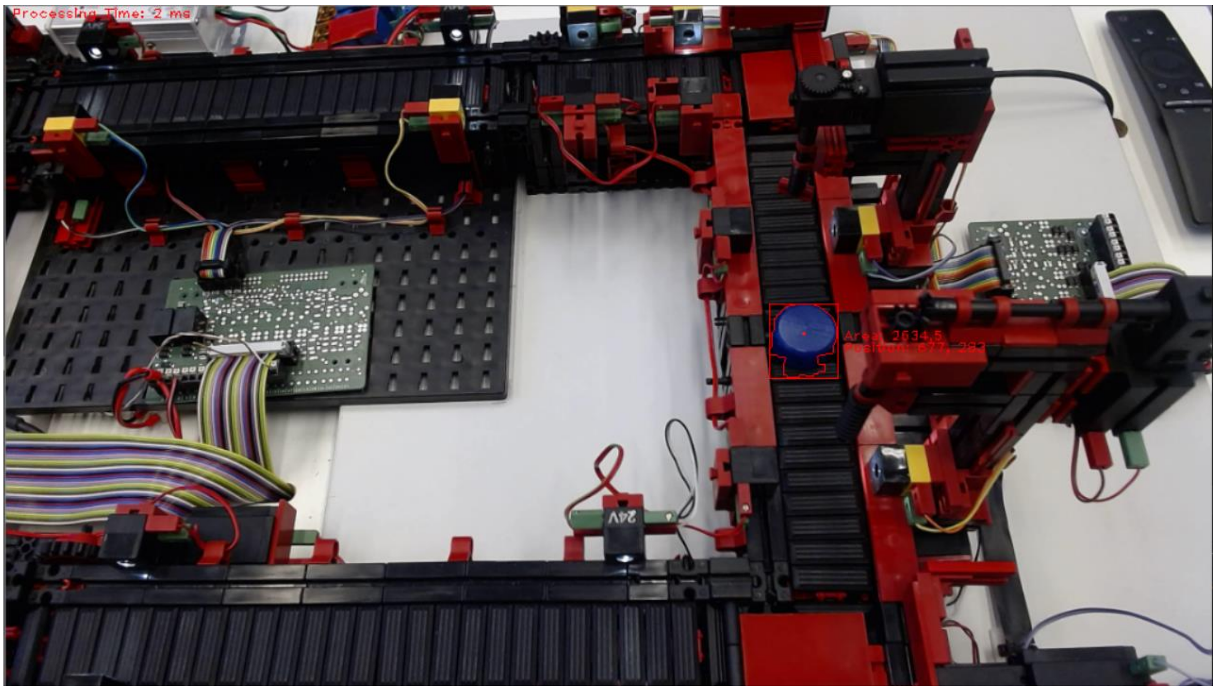


Figure 4.5: Subtraction Method – Accuracy / Position-2

This time, the object detection performance of the Subtraction Method is evaluated at the second position which can be seen from Figure 4.5. Similarly, the algorithm detects a wider area than reality. The center of the object is detected at the pixel-coordinates of (677,283) and the total error is estimated to be around 9.05 pixels.

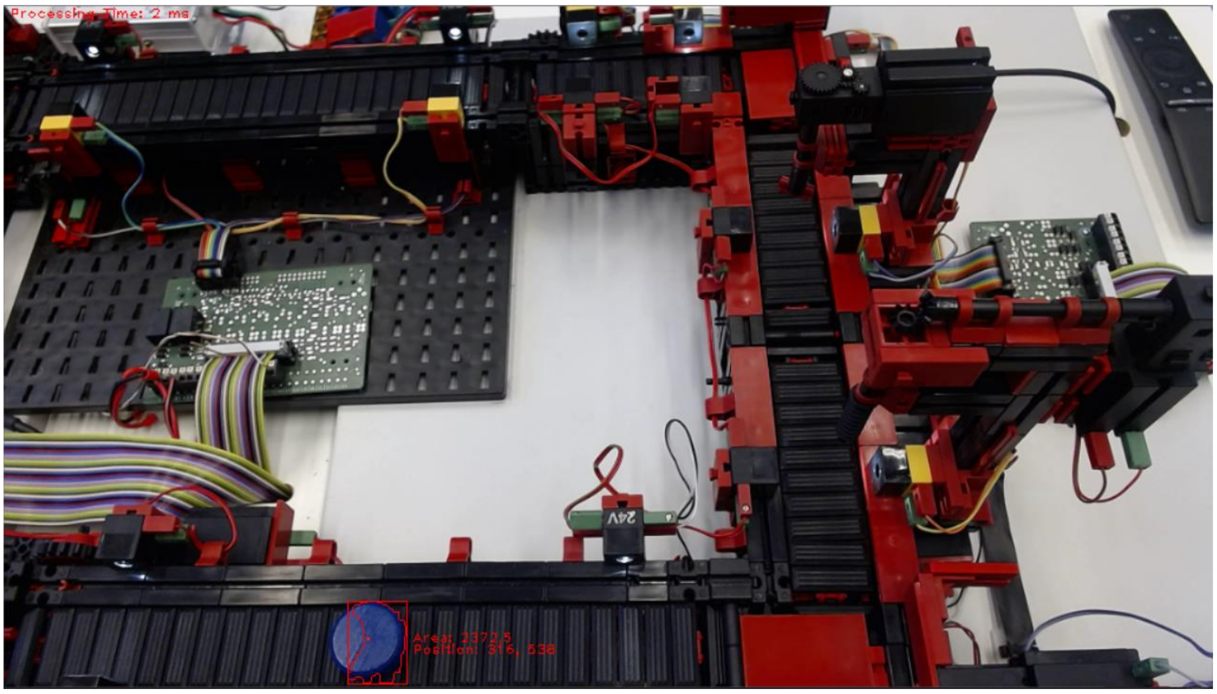


Figure 4.6: Subtraction Method – Accuracy / Position-3

At the third position, the performance of the Subtraction Method was quite unacceptable, which is demonstrated in Figure 4.6. It could not detect almost half of the object. The center is detected at (316,538) whereas the actual position is (306,532). The total error can be rounded to 11.66 pixels.

Color Filter Method – Accuracy

The second method in the list is the Color Filter. As it has been pointed out before, Color Filter has more potential than the Subtraction Method since it can simply erase unwanted colors. A blue product has been used in the experiments and luckily, there are not any other blue parts in the vision of the camera.

At the first position, the object center is detected at the pixels of (283,61) and it corresponds to a total error of 4.24 pixels. Figure 4.7 also shows that the product is well-captured concerning its shape compared to the Subtraction method.

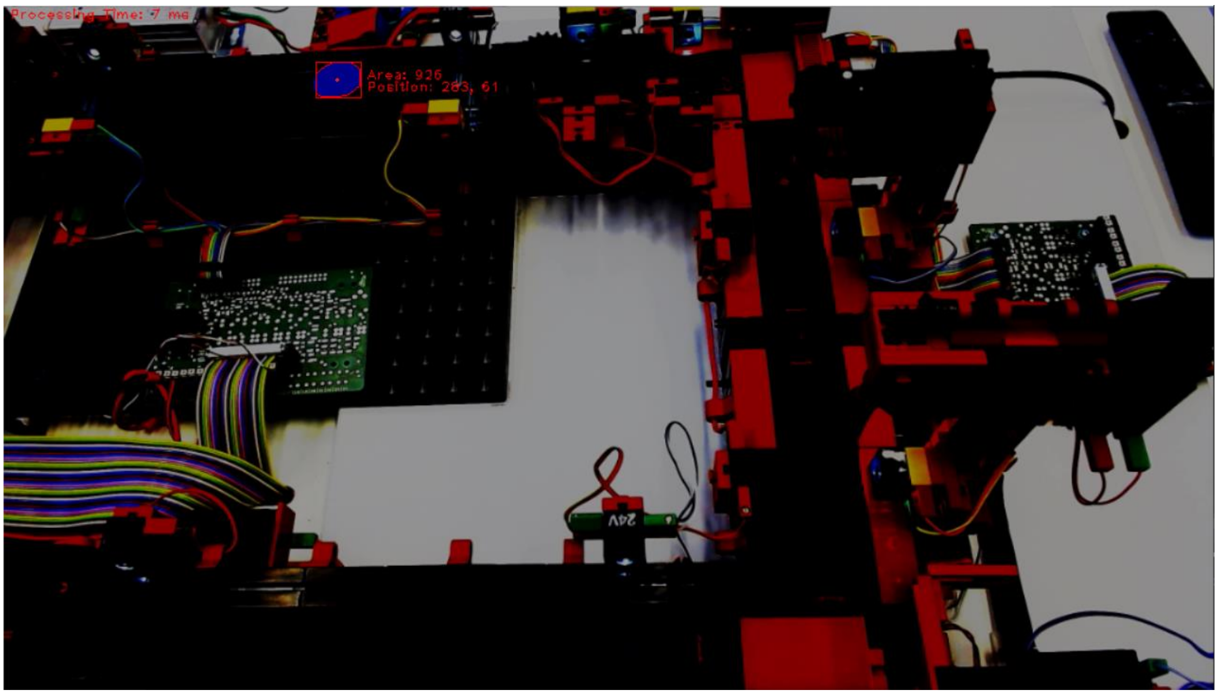


Figure 4.7: Color Filter Method – Accuracy / Position-1

At the second position, the Color Filter struggles to fully capture the product since a part is casting its shadow on the top-right corner of the product and that is interpreted as black color in the algorithm as illustrated in the figure below. The object center is detected at (676,275) and the total error sums up to 4.12 pixels.

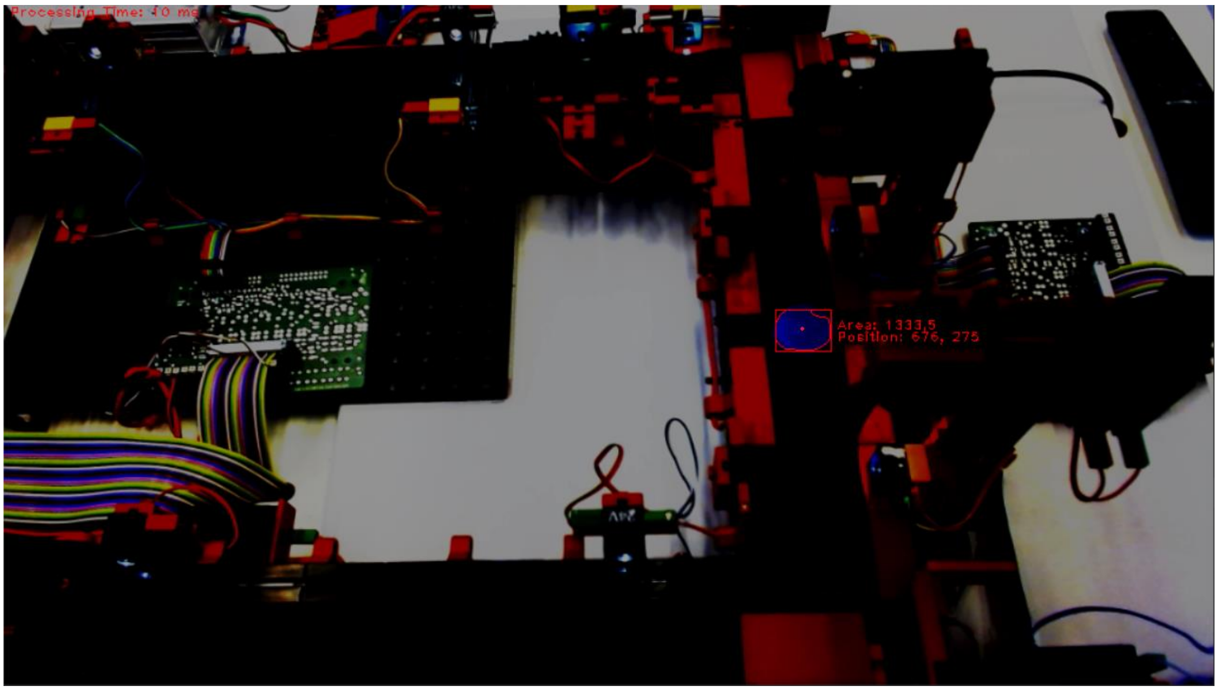


Figure 4.8: Color Filter Method – Accuracy / Position-2

At the third position, the performance of the detection is very well indeed. The shape of the product is perfectly captured, and the error is estimated to be around 2.23 pixels.

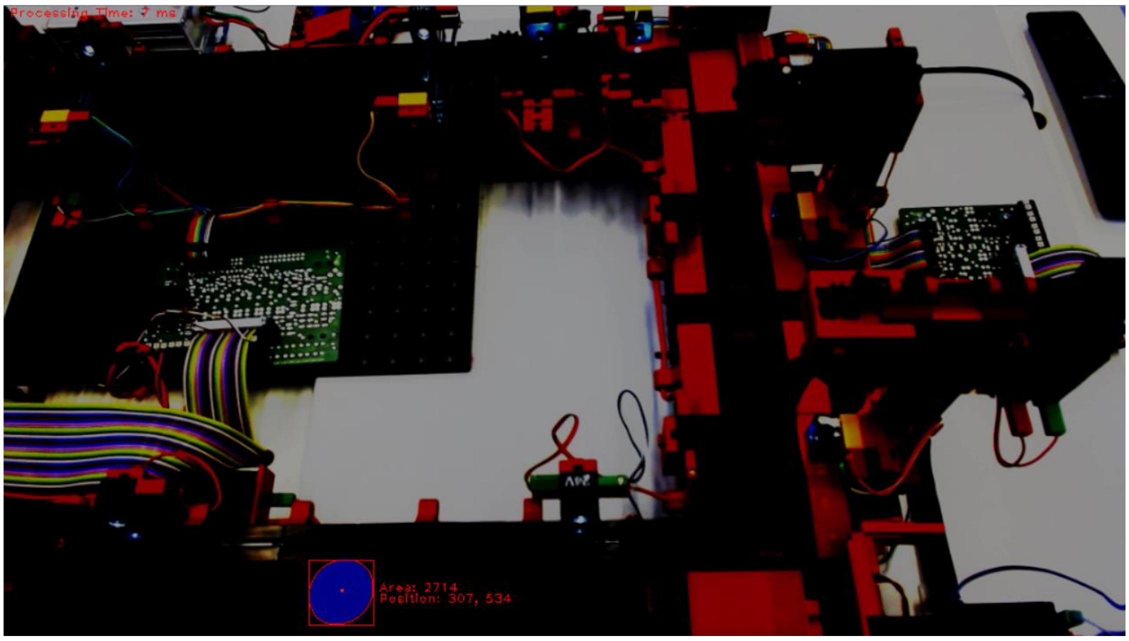


Figure 4.9: Color Filter Method – Accuracy / Position-3

Subtraction&Color Filter Method – Accuracy

Finally, the Subtraction&Color Filter was subjected to the experiments. At the first position, the total error is around 4.12 pixels. As Figure 4.10 shows, the given method is indeed very accurate.

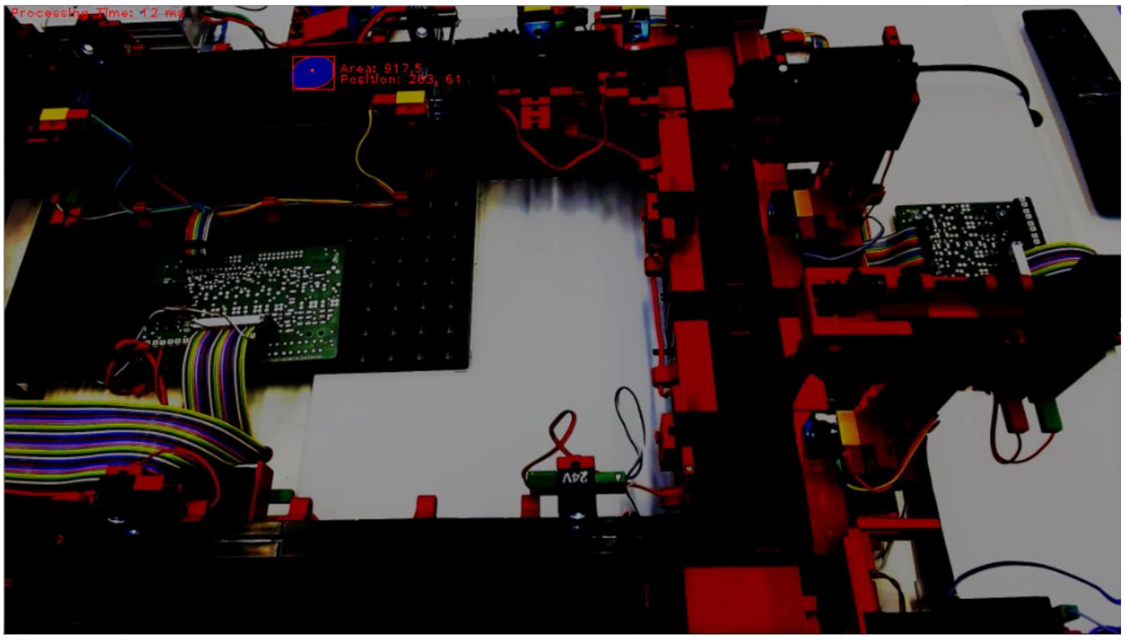


Figure 4.10: Subtraction&Color Filter Method – Accuracy / Position-1

At the second position, the total error is estimated as 2.23 pixels which is a very accurate object detection result.

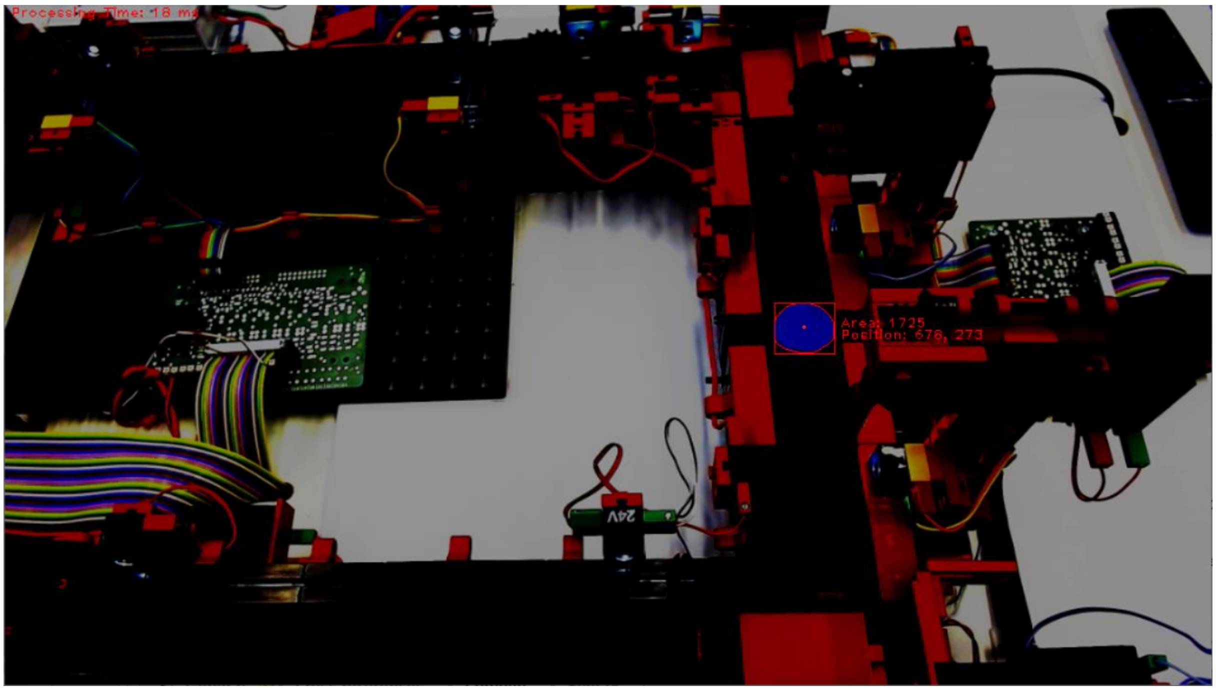


Figure 4.11: Subtraction&Color Filter Method – Accuracy / Position-2

Finally, at the last position, the total error is around 2 pixels. Similar to the Color Filter, the shape of the object is caught nicely, which you can review from the figure below.

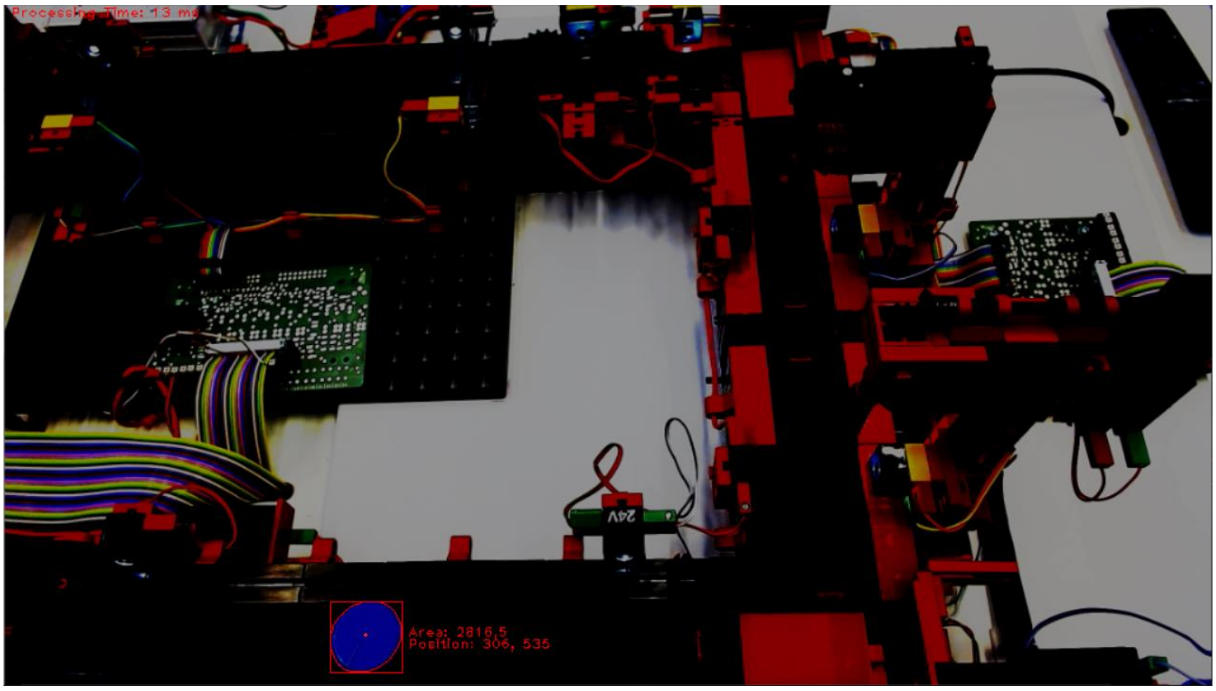
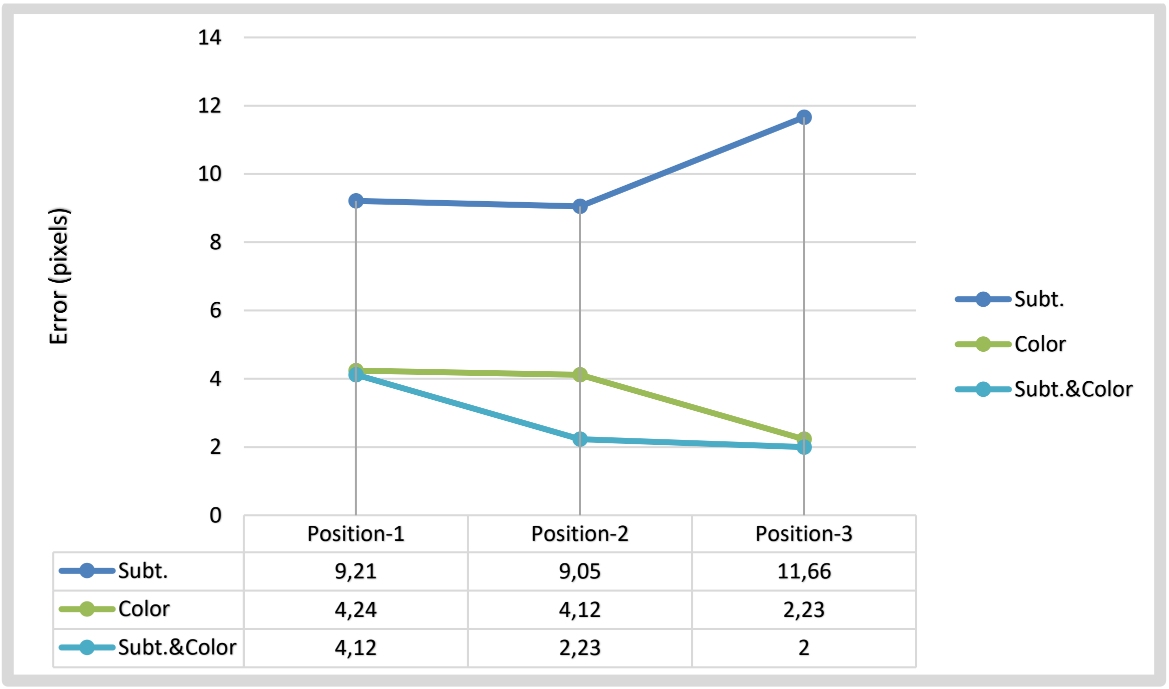


Figure 4.12: Subtraction&Color Filter Method – Accuracy / Position-3

Summary – Accuracy of Object Detection Methods

Overall, it is safe to say that Subtraction&Color Filter is the best performing object detection algorithm in this research. It is followed by Color Filter and at last, Subtraction Method comes. More details are given in Figure 4.13.

Each of these algorithms has its advantages and drawbacks. While color-based methods perform decently in our case, they would fail if a product does not have a distinctive color, or the environment is not bright enough. On the contrary, Subtraction Method does not depend on brightness levels or the color of the products. Regardless, it is prone to vibrations and not suitable if there are other moving parts in a plant. In that case, it may be too hard to differentiate the product from the rest of the assembly.

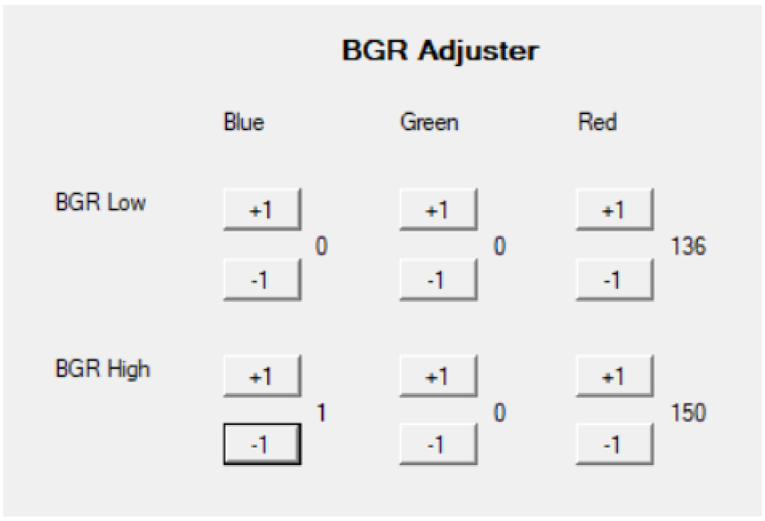


	Position-1	Position-2	Position-3
Subt.	9,21	9,05	11,66
Color	4,24	4,12	2,23
Subt.&Color	4,12	2,23	2

Figure 4.13: Error Rates of Object Detection Methods (in pixels)

4.1.3 Color Variation Test

In the FischerTechnik model plant, there are three different kinds of products which are colored in red, blue, and white. Due to the settings in the automation, only blue products are processed in ILM, where the camera system is located for calibration. Fortunately, most of the parts in ILM are colored in white, red, or black. This makes it easier to detect blue products with color-based object detection methods but since a generalized approach is sought, it should be expected that a product may have a similar color to its environment. Because of that, it is decided to test the performance of the Subtraction&Color Filter with the white and red products at two positions. Furthermore, a dedicated function has been used to precisely set the BGR - color code of the products. The interface of the function is given in the figure below.



BGR Adjuster

	Blue	Green	Red
BGR Low	0	0	136
BGR High	1	0	150

Figure 4.14: BGR Adjuster for precise color code setting

Subtraction&Color Filter Method – Color Variation Test / Position-1

After carefully calibrating the color code, the red product is successfully captured at the first position using Subtraction&Color Filter. However, it should be pointed out that a part, that has a very similar color to the red product, appears lightly in the object detection which can be seen in Figure 4.15.

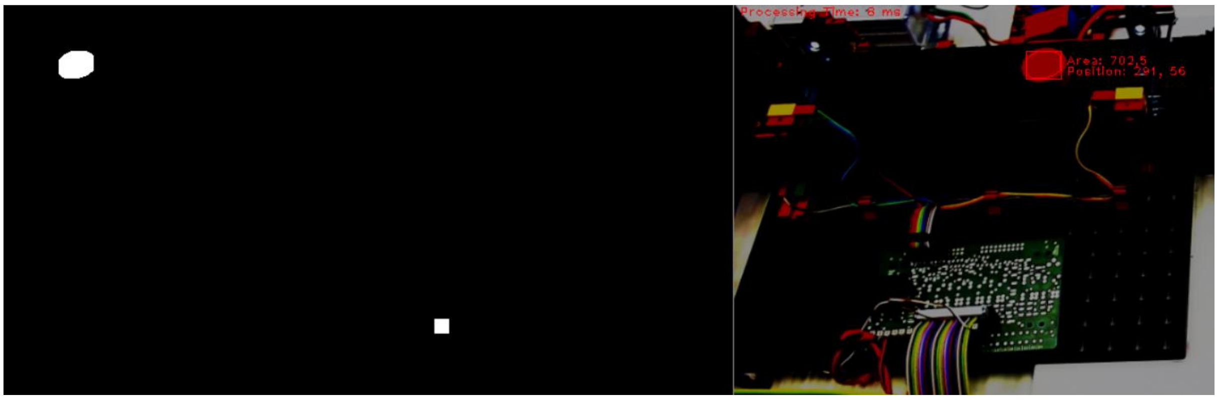


Figure 4.15: Subtraction&Color Filter Method – Color Variation Test / Red Product / Position-1

Subtraction&Color Filter Method – Color Variation Test / Position-2

At the second position, due to reflections from the light sensors, the algorithm fails while capturing the object. As it can be seen from Figure 4.16, almost two-thirds of the product could not be detected.

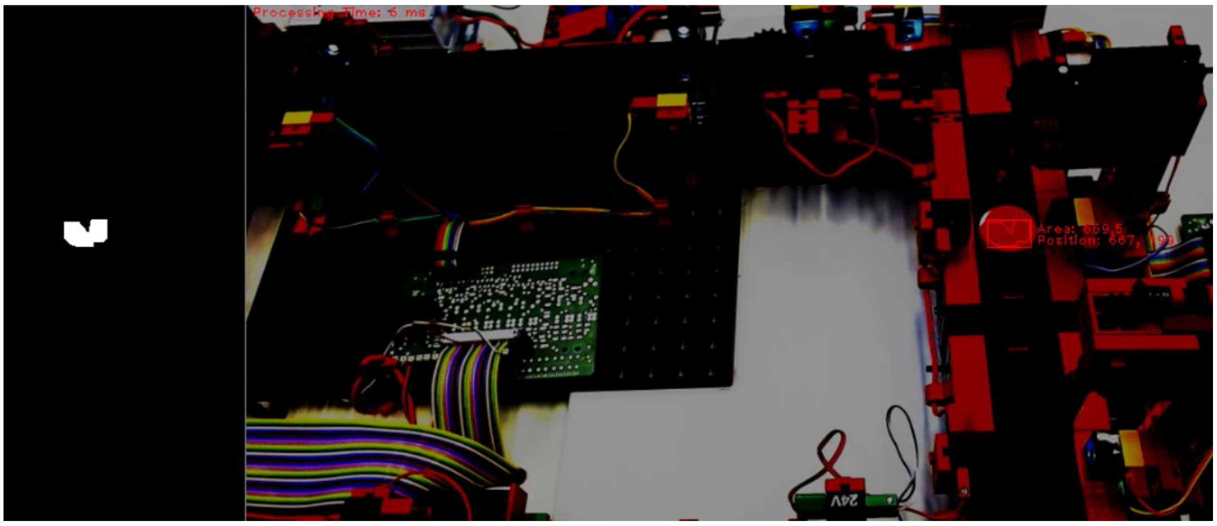


Figure 4.16: Subtraction&Color Filter Method – Color Variation Test / Red Product / Position-2

Figure 4.17 describes the experiment when the range of the red color is increased just a little to capture the rest of the product, other red parts in the plant are detected instead of the product.

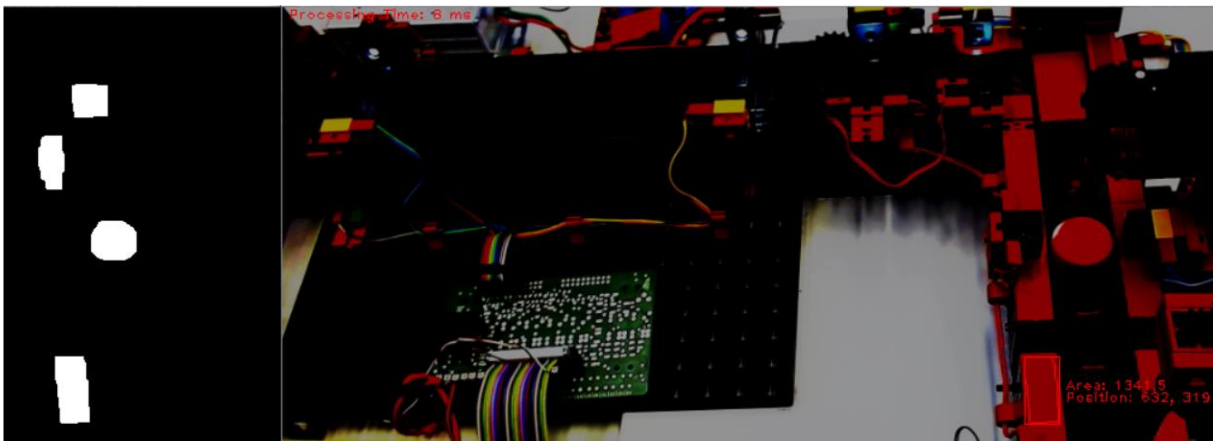


Figure 4.17: Subtraction&Color Filter Method – Color Variation Test / Red Product / Position-2 / Extended Range

Figure 4.18 shows the final experiment, that is carried out using a white product at the second position. It has been observed that white products can be captured much better than the red products because of the external conditions such as color of the irrelevant parts, lightning, etc.

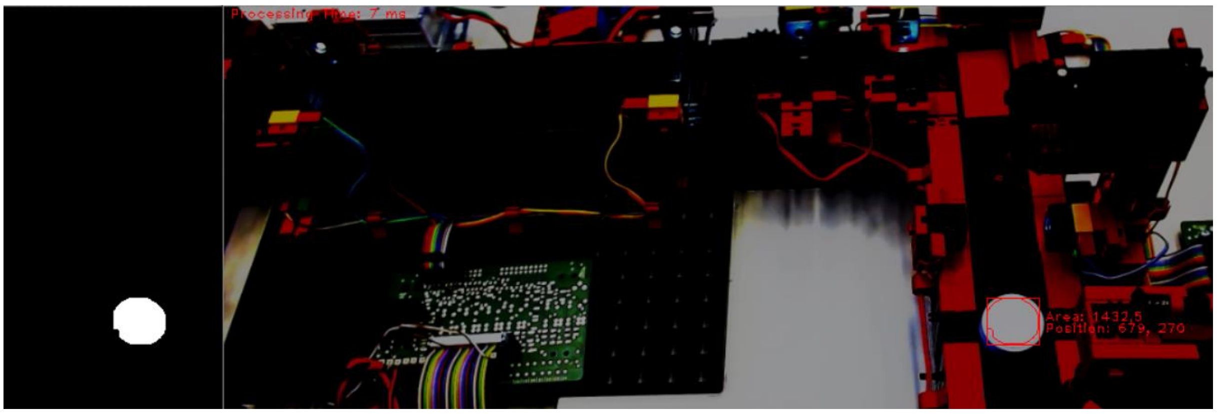


Figure 4.18: Subtraction&Color Filter Method – Color Variation Test / White Product / Position-2

4.1.4 Summary – Performance of the Object Detection Methods

In total, three different methods have been implemented in this research, to detect the products in a plant. They all have their upsides and downsides regarding accuracy, speed, and flexibility.

Firstly, Subtraction Method is a very fast approach compared to the color-based methods. It is also not vulnerable to environmental conditions such as brightness, reflections, background colors, and so on. But the current version of the Subtraction Method cannot accurately detect the products and has not been used frequently consequently.

On the other hand, if the color codes are carefully introduced to the algorithm, then the accuracy of color-based methods can be very high. But their performance degrades when there is not enough brightness and color divergence between the background and the products.

4.2 Performance of the Calibration Software

Besides object detection, the calibration software is also responsible for generating timing results and updating the digital twins. In fact, these tasks take more time to be completed than object detection. In this section, the performance evaluation of these tasks has been experimented with.

4.2.1 Result Generation Performance

The generation of the timing results is not as straightforward as it at first seems. Firstly, a timer constantly works in the background to save time. Secondly, the user enters the detection coordinates for the products, but it is assumed that the order of the coordinates may not be the same with the product's path. Therefore, the distance between detection coordinates and product's location is calculated each cycle. Thirdly, a frame is grabbed from the camera and adjusted depending on the object detection method. Therefore, generation of the results requires some time, which will be investigated.

The loop of the calibration software starts with frame grabbing. To evaluate the performance, the time required between consecutive frame capturing events is calculated in the experiment. Additionally, Color Filter is used as an object detection method for this purpose. Finally, the experiment is performed on three different sections of the model plant.

Section 1 – Result Generation Performance

In the first section, it is observed that the average processing time, including the object detection method, is turned out to be around 66 ms, as it is demonstrated in the figure below. Considering that Color Filter requires 7 ms on average, the rest of the algorithm requires 59 ms.

Also, it can be concluded that the calibration software offers 15 Fps while detecting products, running User Interfaces, and processing the data.

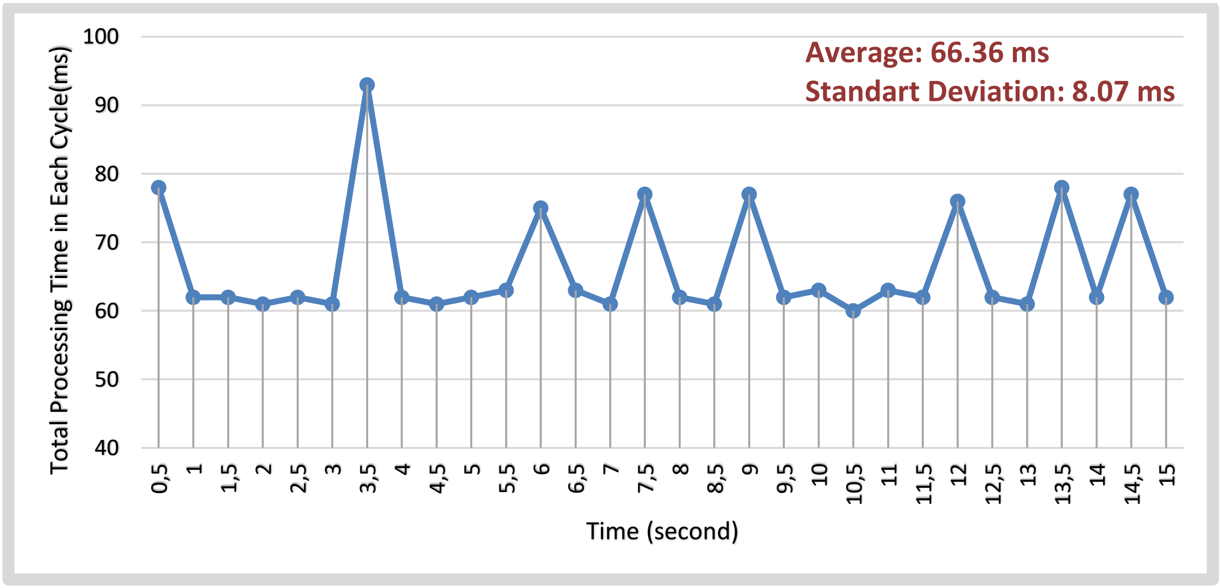


Figure 4.19: Result Generation Performance – Section 1

Section 2&3 – Result Generation Performance

The figure below describes the total processing time in the second section of the model plant. The average time is very similar to the first section, but the data is more fluctuated.

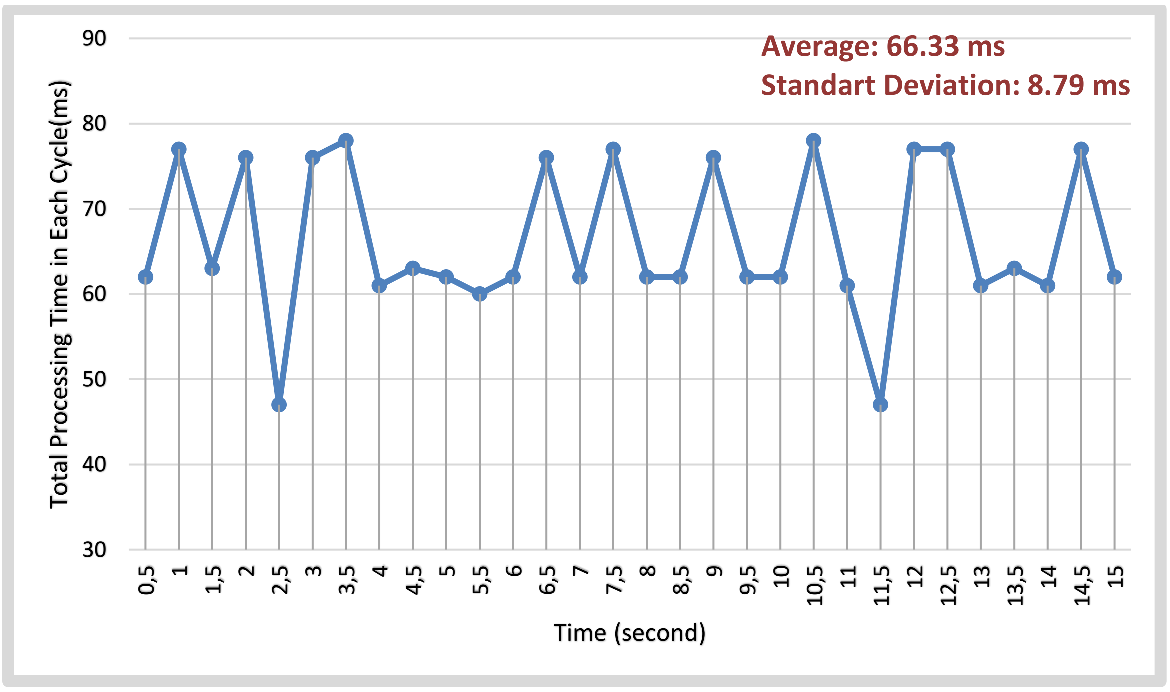


Figure 4.20: Result Generation Performance – Section 2

In the third section, the average processing time has dropped 1 ms while having the same amount of fluctuations as the second section which can be seen in Figure 4.21. From the previous experiments, we know that the standard deviation of the Color Filter is around 1.32 ms. The rest of the contributions to the standard deviation is believed to come from frame grabbing which is not too stable.

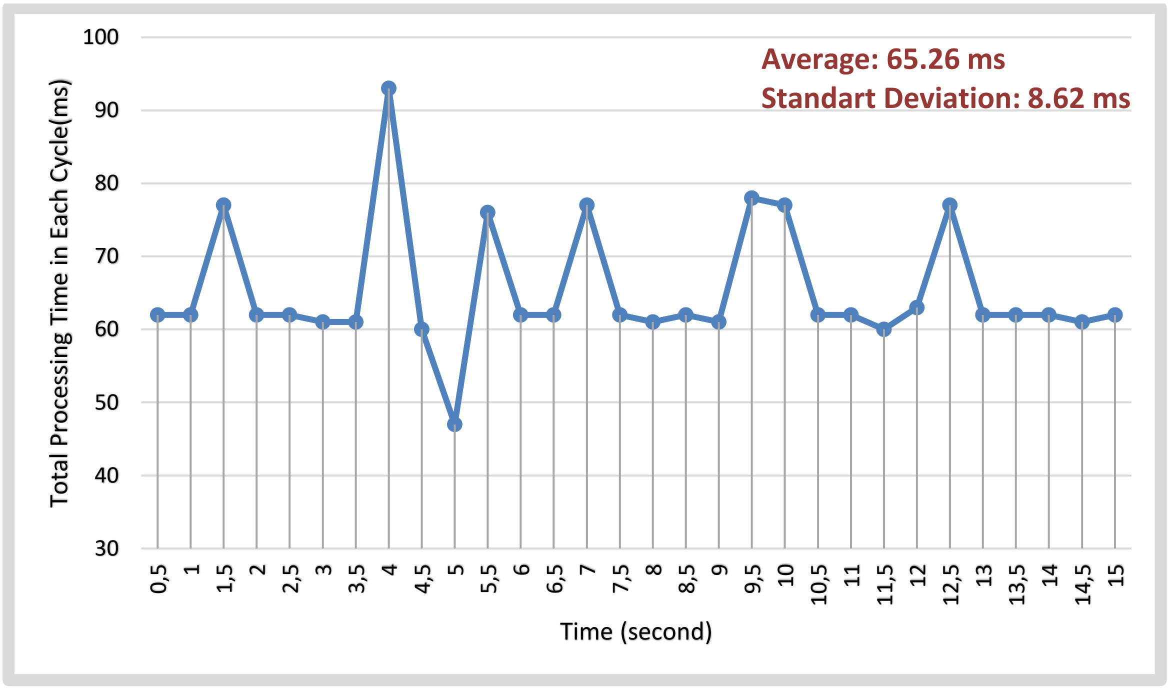
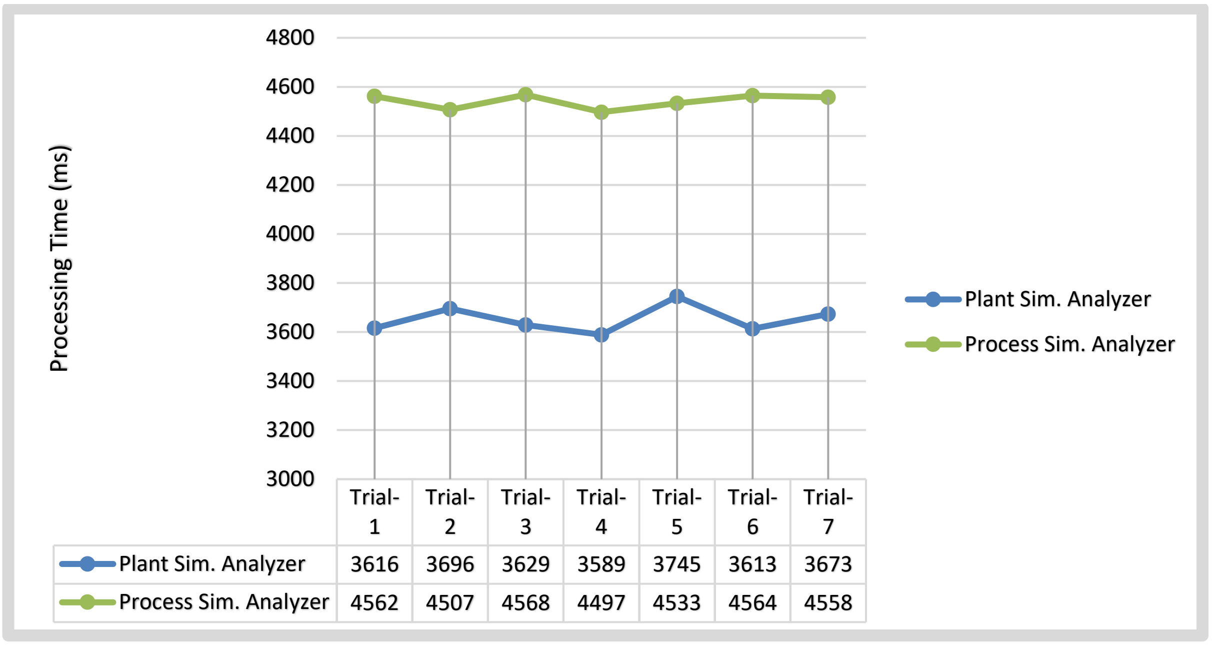


Figure 4.21: Result Generation Performance – Section 3

4.2.2 Updating the Digital Twins

When the collection of the timing results is performed by the algorithm, they are forwarded to Process Simulate Analyzer and Plant Simulation Analyzer if automatic calibration is turned on. After that, the methods in these analyzers update the digital twins. In this section, the total required time for these methods is studied by updating the digital twins seven times.

The results of the experiment can be checked in Figure 4.22. The average time required to update the Plant Simulation model is around 3651 ms, whereas 890 ms more is needed for updating the Process Simulate model, which stands at 4541 ms. Even though Plant Simulation Analyzer cannot set the parameters without opening the model, it is still much faster than Process Simulate Analyzer since the model file of Process Simulate contains many unrelated information and it takes time to extract relevant information from it. In any case, the processing speed of these functions is not so critical owing to the threaded structure of the software.



	Trial-1	Trial-2	Trial-3	Trial-4	Trial-5	Trial-6	Trial-7
Plant Sim. Analyzer	3616	3696	3629	3589	3745	3613	3673
Process Sim. Analyzer	4562	4507	4568	4497	4533	4564	4558

Figure 4.22: Total Required Time for Updating the Digital Twins

4.3 Resource Consumption of the Calibration Software

In this section, the total resource consumption of the calibration software will be discussed in the context of RAM, space, and processing usage.

Disk Usage – the Calibration Software

The software consists of four main forms, two assistant forms, and the main program. In total, around 3000 lines of code are written, and three external libraries are imported which are Emgu.CV, UnifiedAutomation, and eMPlantLib. They are used for the methods of object detection, OPC UA, and Plant Simulation Analyzer. All of them take disk space of 110 MB. Most of the contribution comes from Emgu.CV, which is around 88 MB.

RAM & Processing Usage – the Calibration Software

To test RAM and processing usage, an assistant UI is implemented which collects the relevant usage of the workstation. Therefore, all of the background applications are closed beforehand to prevent any miscalculation of the results. Two experiments are conducted for each of the object detection methods and average usages are gathered for two minutes.

Figure 4.23 explains the resource consumption of the calibration software when Subtraction Method is used as an object detection method for two minutes. The average CPU consumption of the trials is 13.66% and RAM consumption is 33.37%.

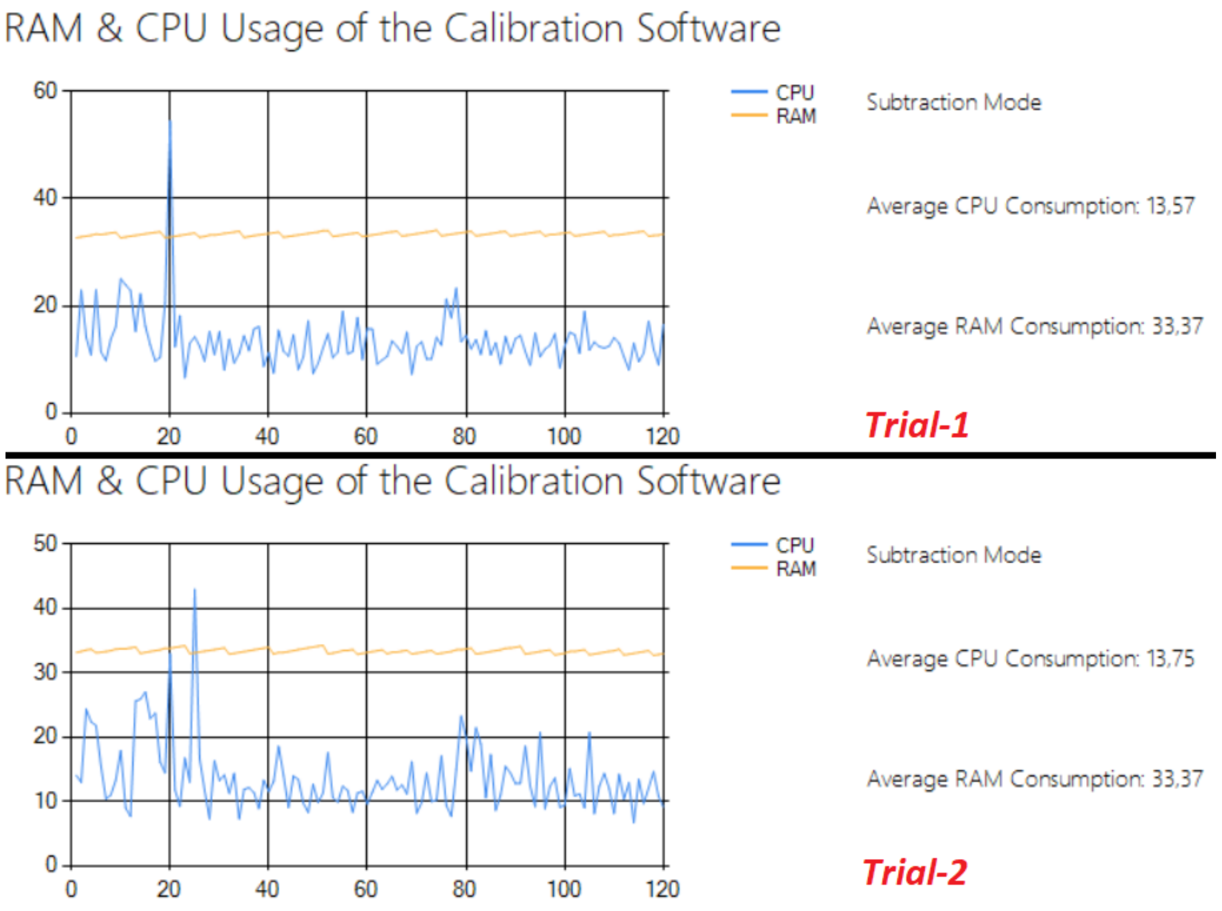


Figure 4.23: Calibration Software / Subtraction Method – Resource Consumption

Similarly, Figure 4.24 shows the resource consumption of the calibration software when the Color Filter is used as an object detection method for two minutes. The average CPU consumption of the trials is 14.81% and RAM consumption is 37.105%. Even though there is only a 1% difference between Color Filter and Subtraction in terms of CPU consumption, it has been noticed that there is almost a 4% difference in terms of RAM usage.

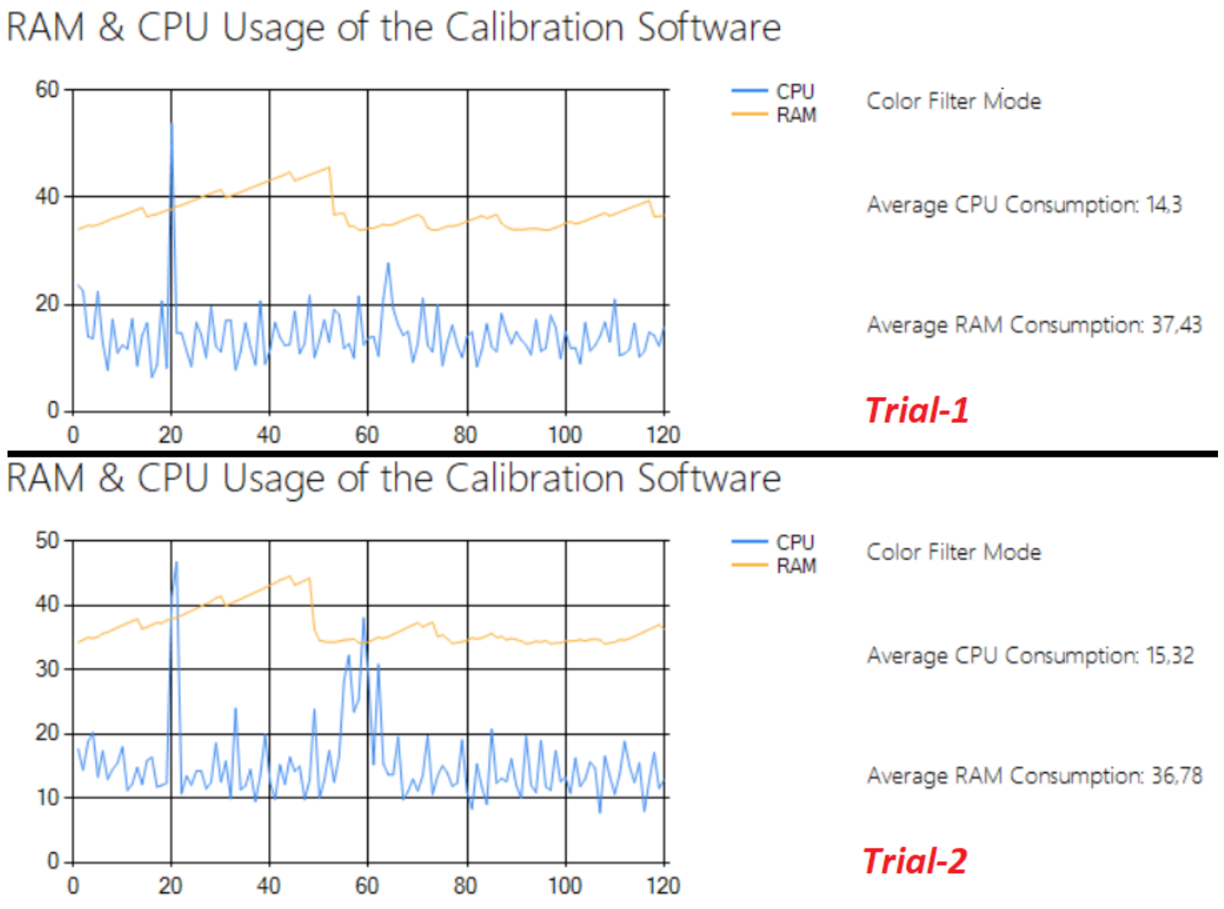


Figure 4.24: Calibration Software / Color Filter Method – Resource Consumption

This time, Figure 4.25 displays the resource consumption of the calibration software when Subtraction&Color Filter Method is used as an object detection method for two minutes. The average CPU consumption of the trials is 16.11% and RAM consumption is 37.825%. It is fair to say that even the most resource-consuming method of the calibration software can be run easily with the workstation.

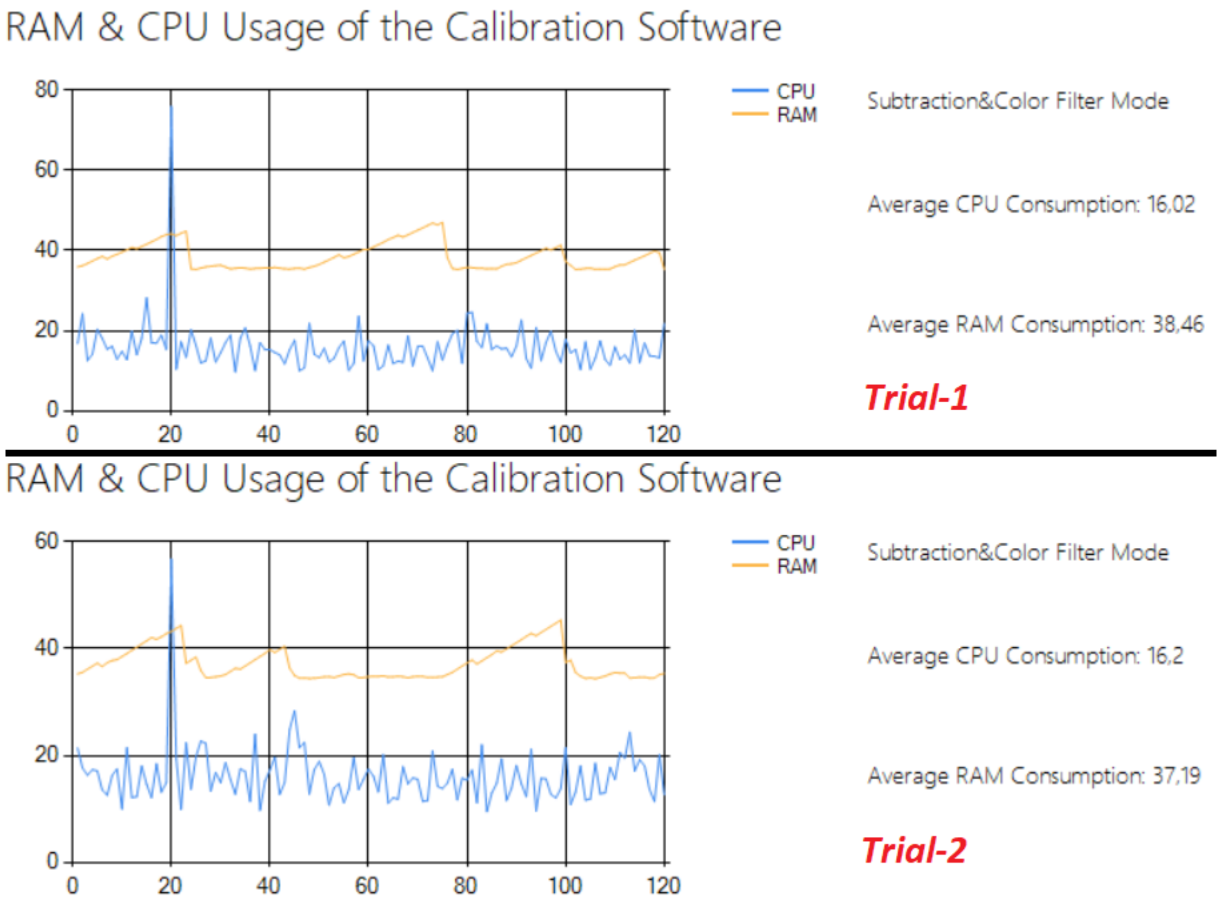


Figure 4.25: Calibration Software / Subtraction&Color Filter – Resource Consumption

4.4 Sample Scenarios for the Calibration Software

The main purpose of the calibration software is to update digital twins of a plant. This can be done either for an initial calibration or as regular maintenance. To test the capabilities of the calibration software in accordance with its purpose, unsteady conditions must be introduced to the model plant so that the reaction of the software can be evaluated.

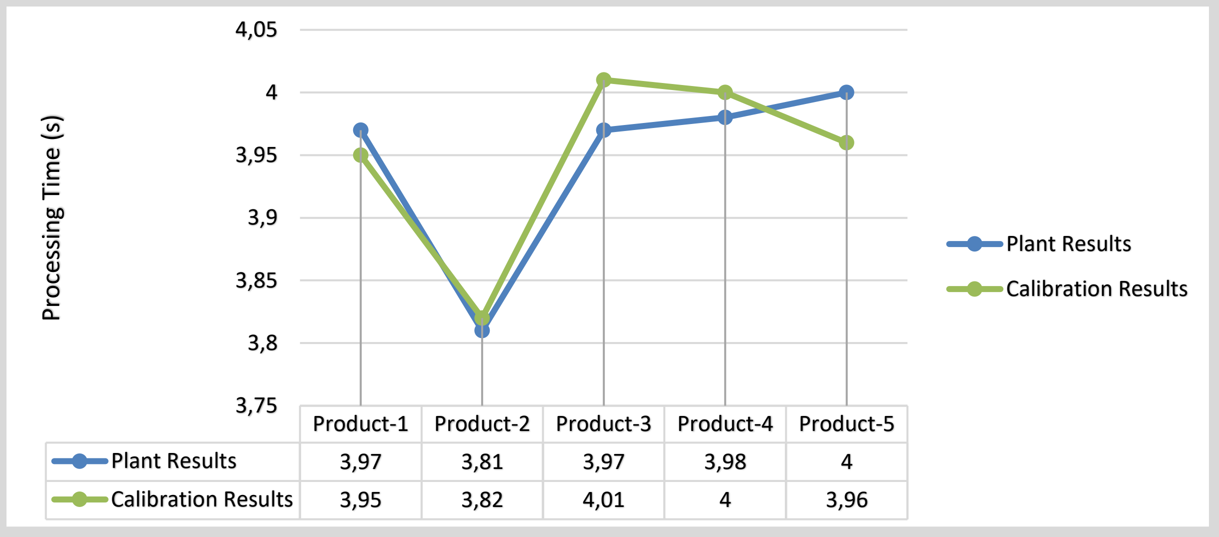
Under the current circumstances, it was decided to integrate a potentiometer into the model plant. A potentiometer is a device that can be used to manually control electrical current by sliding contact. It is connected to the first conveyor of ILM so that the voltage of the conveyor would be adjusted.

With the potentiometer, it is now possible to introduce variable processing time. It means that some sample scenarios can be developed in which the calibration software is tested under different conditions. For the experiments, Color Filter is selected as the main object detection method.

4.4.1 Sample Scenario – 1

The first scenario depicts a situation in which there is a physical problem with a conveyor and the transportation speed is decreased consequently. This may cause synchronization issues between digital twins and a plant. To avoid it, the calibration software should update digital twins accurately.

Given scenario is simulated with the help of the potentiometer. Normally, the motor of the conveyor works at 24 Volts and the digital twins are calibrated according to the speed at that voltage level. In the first experiment, the voltage is lowered to 17.5 – 18.5 Volts, and then five workpieces are processed there sequentially. The result of the calibration and the actual plant values are compared in the figure below.

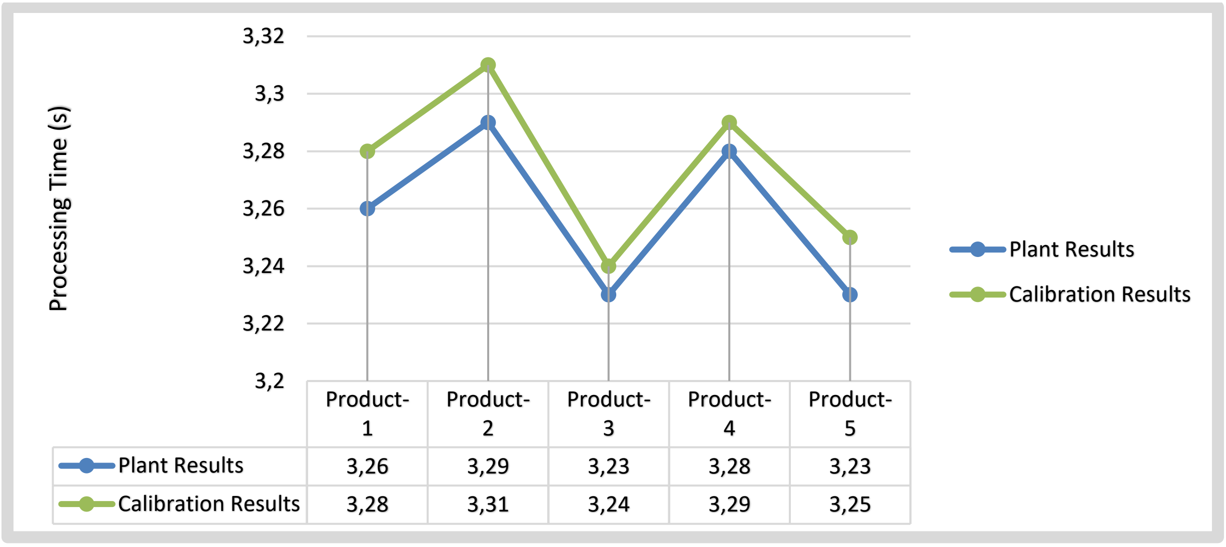


	Product-1	Product-2	Product-3	Product-4	Product-5
Plant Results	3,97	3,81	3,97	3,98	4
Calibration Results	3,95	3,82	4,01	4	3,96

Figure 4.26: Voltage Level 17.5 – 18.5 V / Comparison of Plant and Calibration Results

The variations in the graph mostly occur due to fluctuations in the voltage levels which could not set to a specific value because of the potentiometer's working principle. If we put that aside, it can be seen that the biggest difference between plant and calibration values is 40 ms. This value can even converge to zero if the detection points are carefully selected. In this case, they are chosen quickly without spending too much effort.

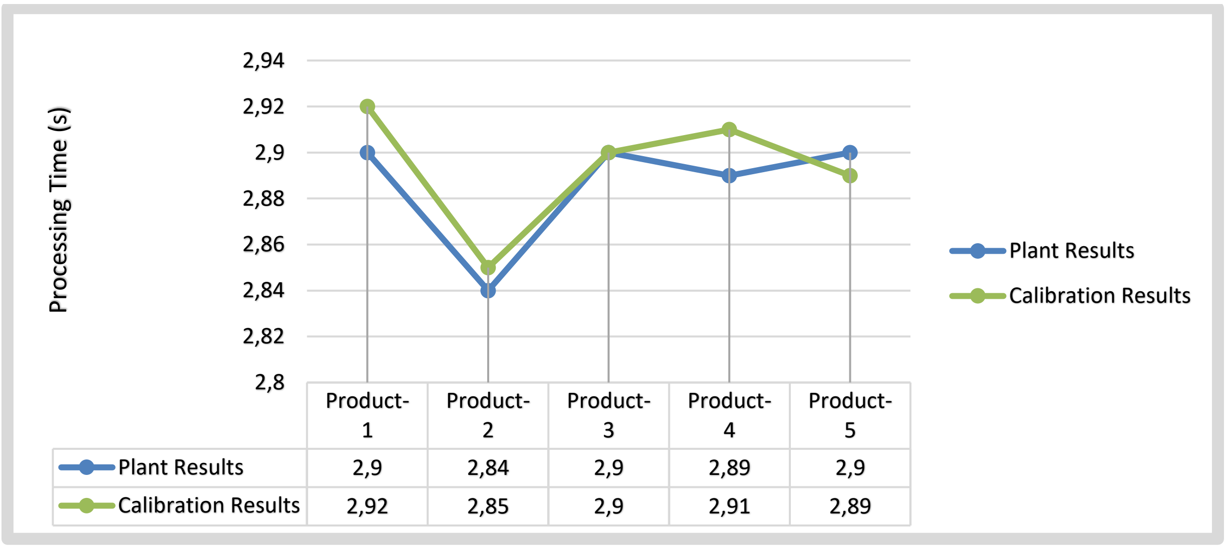
In the second experiment, the voltage level is increased to a band of 20.75 – 21.20 Volts. Similarly, five products are processed in ILM consecutively and the corresponding data is gathered in the figure below. Once again, it has been observed that the calibration results are successfully following actual results.



	Product-1	Product-2	Product-3	Product-4	Product-5
Plant Results	3,26	3,29	3,23	3,28	3,23
Calibration Results	3,28	3,31	3,24	3,29	3,25

Figure 4.27: Voltage Level 20.75 – 21.2 Volts / Comparison of Plant and Calibration Results

In the final experiment, the voltage level is increased to its maximum capacity, which is exactly 23.93 Volts. The same methods are applied to collect the results and they can be seen in the figure below.



	Product-1	Product-2	Product-3	Product-4	Product-5
Plant Results	2,9	2,84	2,9	2,89	2,9
Calibration Results	2,92	2,85	2,9	2,91	2,89

Figure 4.28: Voltage Level 23.93 Volts / Comparison of Plant and Calibration Results

The given experiments confirmed that the calibration software will react to changing variables as quickly as the plant itself. Considering that the software processes each frame 65 ms on average, the total error cannot pass this number theoretically.

4.4.2 Sample Scenario – 2

Like the first one, the second scenario also depicts a situation in which there is a physical problem with a conveyor and the transportation speed is decreased remarkably. However, this time, the feedback of Statistical Inspection UI will be investigated. The UI is designed to make basic statistical deductions and analyses based on the average results of the products. In case there is too much deviation between the processing time of a new product and the average results, a warning should be displayed on Statistical Inspection, which will be checked by this scenario.

The scenario is simulated with the help of the potentiometer likewise. In ideal conditions, the motor of the conveyor works at 24 Volts and the former results are collected based on that voltage level. A product generally needs 2.9 seconds to pass the conveyor, as illustrated below.

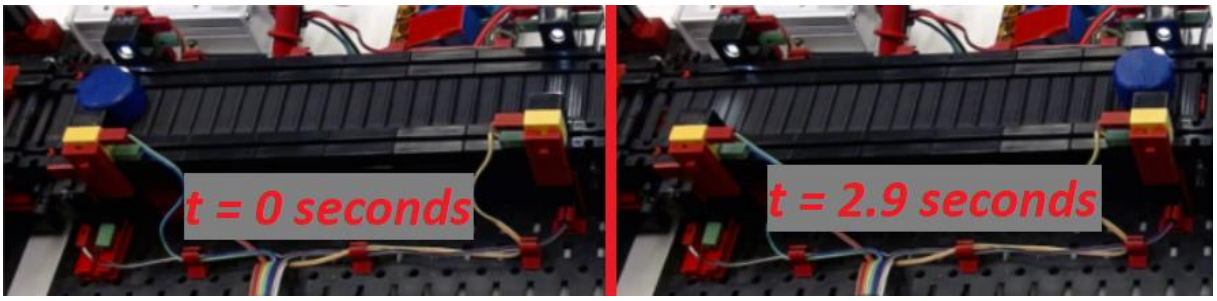


Figure 4.29: 24 Volts / Product Motion on ILM Conveyor

However, a voltage-drop to 12 – 14 Volts will cause a longer material motion that is at least two seconds more than the previous configuration, which can be seen in Figure 4.30. At the same time, Statistical Inspection is prepared in such a way that an alarm would be set in case the elapsed time difference is more than two seconds.

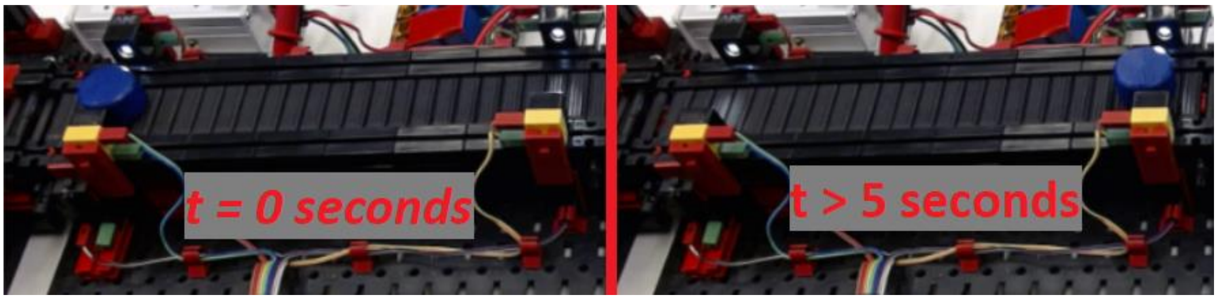
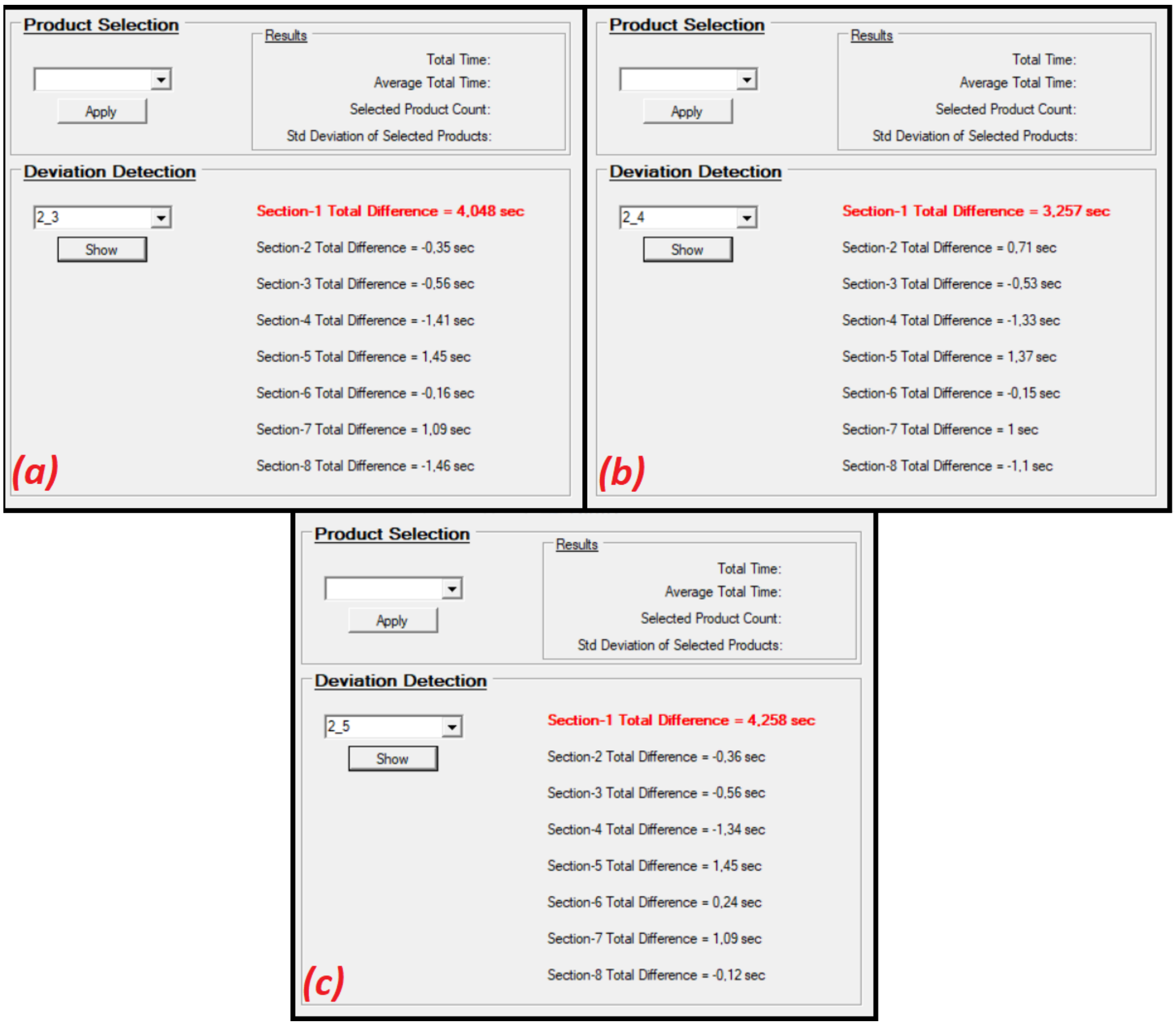


Figure 4.30: 12 – 14 Volts / Product Motion on ILM Conveyor

In the experiments, three products are processed right after the voltage is dropped. The results of the Statistical Inspection are shown in the figure below.



(a)

Product Selection

Apply

Results

Total Time:
Average Total Time:
Selected Product Count:
Std Deviation of Selected Products:

Deviation Detection

2_3

Show

Section-1 Total Difference = 4,048 sec
Section-2 Total Difference = -0,35 sec
Section-3 Total Difference = -0,56 sec
Section-4 Total Difference = -1,41 sec
Section-5 Total Difference = 1,45 sec
Section-6 Total Difference = -0,16 sec
Section-7 Total Difference = 1,09 sec
Section-8 Total Difference = -1,46 sec

(b)

Product Selection

Apply

Results

Total Time:
Average Total Time:
Selected Product Count:
Std Deviation of Selected Products:

Deviation Detection

2_4

Show

Section-1 Total Difference = 3,257 sec
Section-2 Total Difference = 0,71 sec
Section-3 Total Difference = -0,53 sec
Section-4 Total Difference = -1,33 sec
Section-5 Total Difference = 1,37 sec
Section-6 Total Difference = -0,15 sec
Section-7 Total Difference = 1 sec
Section-8 Total Difference = -1,1 sec

(c)

Product Selection

Apply

Results

Total Time:
Average Total Time:
Selected Product Count:
Std Deviation of Selected Products:

Deviation Detection

2_5

Show

Section-1 Total Difference = 4,258 sec
Section-2 Total Difference = -0,36 sec
Section-3 Total Difference = -0,56 sec
Section-4 Total Difference = -1,34 sec
Section-5 Total Difference = 1,45 sec
Section-6 Total Difference = 0,24 sec
Section-7 Total Difference = 1,09 sec
Section-8 Total Difference = -0,12 sec

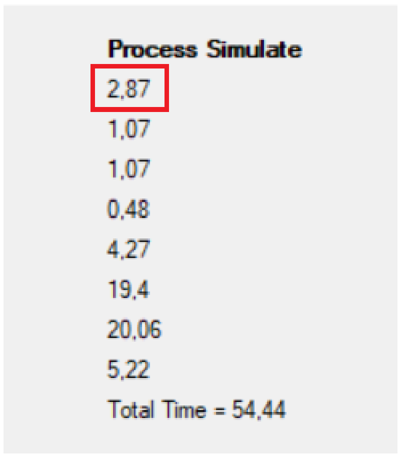
Figure 4.31: Voltage Drop Results of Statistical Inspection, (a) Product ID: 2_3, (b) Product ID: 2_4, (c) Product ID: 2_5

It can be seen that Statistical Inspection UI has given a warning for Section-1, which corresponds to the conveyor that has a lower voltage. Since the difference does not cross two seconds threshold on the other sections, no warning is published.

4.4.3 Sample Scenario – 3

In the last scenario, a similar situation is depicted in which there is a physical problem with a conveyor, and it is working slower than expected. Because of that, digital twins will no longer be synchronized with the actual system. This would create inaccurate results in digital twin simulations. Since the main purpose of the research is to prevent this situation from happening, it is decided to simulate the given scenario by using Process Simulate based digital twin of the model plant.

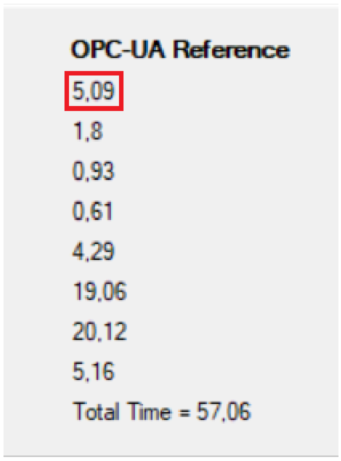
At the beginning of the scenario, the digital twin is calibrated manually according to the data obtained from the model plant when the conveyor is set to 24 Volts. Total processing time reaches 54.44 seconds and the time spent on the conveyor resulted in 2.87 seconds as illustrated below.



```
Process Simulate
2,87
1,07
1,07
0,48
4,27
19,4
20,06
5,22
Total Time = 54,44
```

Figure 4.32: Timing Data / 24 Volts on the Conveyor

After that, the voltage is dropped to 15 Volts. On that level, the conveyor is barely moving, and the processing time is increased notably which can be seen in Figure 4.33.



```
OPC-UA Reference
5,09
1,8
0,93
0,61
4,29
19,06
20,12
5,16
Total Time = 57,06
```

Figure 4.33: Timing Data / 15 Volts on the Conveyor

The problem in the conveyor costs 2.22 seconds in each cycle of the product. It may be insignificant, but the accumulation of total missing time is going to be crucial if there were hundreds of products and they are being processed sequentially. Therefore, an updated digital twin would be useful to figure out a solution.

To update the digital twin, automatic update mode is turned on and the calibration software is run. The updated digital twin can be seen in Figure 4.34. We can see that the conveyor in the digital twin is updated exactly to 5.09 seconds.

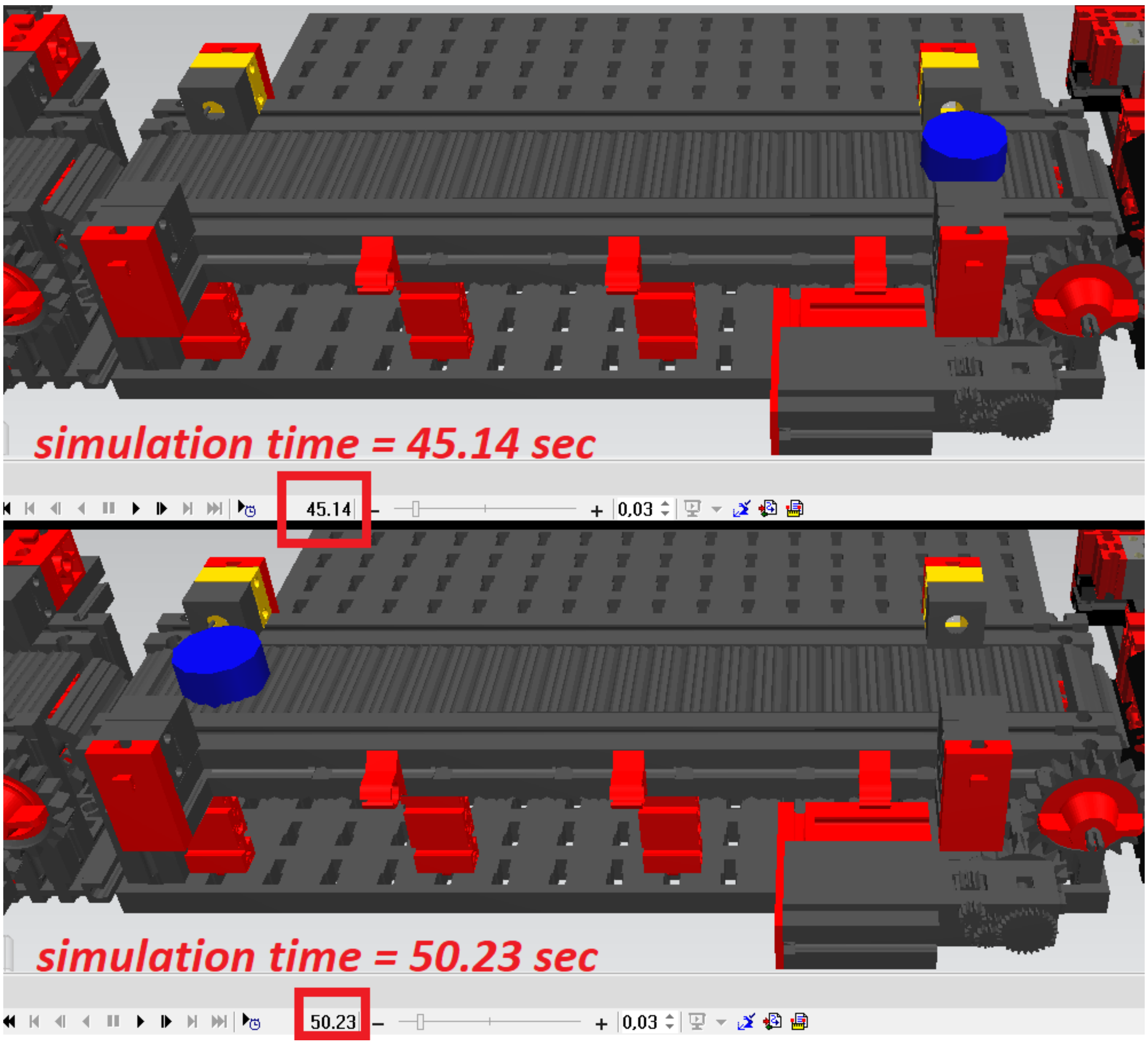


Figure 4.34: Process Simulate Model / Updated using the Calibration Software

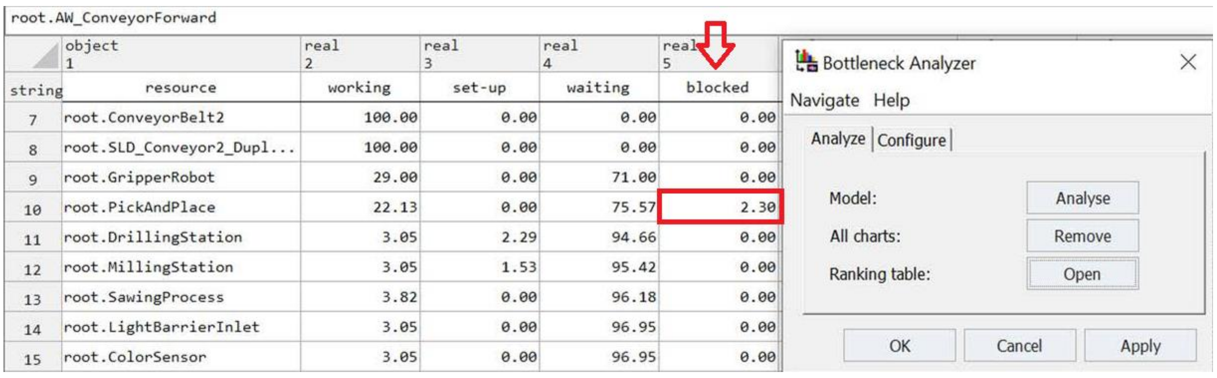
5 Applications of the Calibration Software

In this chapter, two digital twin applications, that are feasible owing to the Calibration Software, will be discussed.

As it has been pointed out before, digital twins are very useful tools for developing additional operational support for flexible manufacturing plants. Because, vast amounts of data and complex production networks requires smart cyber-physical solutions, such as digital twins. For example, production lines may face bottlenecks depending on the physical setup and order of the products during production. A similar problem is simulated in the FischerTechnik model and it is decided to use the digital twin implemented in Plant Simulation to monitor and eliminate the problem.

Firstly, the model plant and Calibration Software are run for once so that the digital twin is calibrated. Otherwise inaccuracies would decrease the quality of the results. After that, the digital twin is ready to analyze the model plant more deeply by using the Bottleneck Analyzer. It is a built-in tool that shows the user if there exists any bottleneck in the current configuration or not. It has also some statistical features which display the amount of working time, disrupted time, and paused time of every element in the simulation. It is also possible to create a chart and rank bottleneck percentages that are sorted according to the selected criteria.

The figure below demonstrates a sample application of the Bottleneck Analyzer. Because of the bottleneck, a robot named PickAndPlace is blocked for 2.3%, in terms of the operation time.



root.AW_ConveyorForward

	object 1	real 2	real 3	real 4	real 5
string	resource	working	set-up	waiting	blocked
7	root.ConveyorBelt2	100.00	0.00	0.00	0.00
8	root.SLD_Conveyor2_Dupl...	100.00	0.00	0.00	0.00
9	root.GripperRobot	29.00	0.00	71.00	0.00
10	root.PickAndPlace	22.13	0.00	75.57	2.30
11	root.DrillingStation	3.05	2.29	94.66	0.00
12	root.MillingStation	3.05	1.53	95.42	0.00
13	root.SawingProcess	3.82	0.00	96.18	0.00
14	root.LightBarrierInlet	3.05	0.00	96.95	0.00
15	root.ColorSensor	3.05	0.00	96.95	0.00

Figure 5.1: Sample Bottleneck Analyzer Results from Plant Simulation

Similar to the first application, second application also make use of a digital twin, particularly based on Process Simulate. However, before working with the digital twin, the Calibration Software should be utilized to update the digital twin so that the best performance can be achieved.

One of the advantages of having such a fully functional digital twin on Process Simulate is the ability to implement many kinds of additional logic methods. To illustrate the point, the blockage detection algorithm can be given as an example. Because of any sort of equipment failure or unexpected obstacle occurrence on the path of material flow, the production process may be interrupted. To overcome the given challenge with Process Simulate, a logical control block has been implemented which continuously compares the actual processing location of the workpieces with the expected processing location. If the difference is greater than a predefined threshold, Process Simulate sets an alarm signal in PLCs and HMI so that operator would be informed about the emergency in the model plant, which is presented in Figure 5.2.

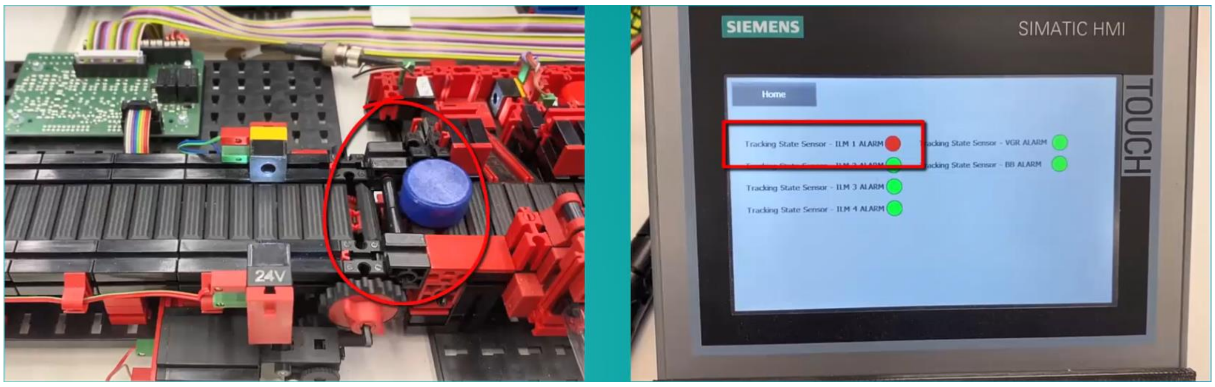

24V
SIEMENS
SIMATIC HMI
TOUCH
Home
Tracking State Sensor - ILM 1 ALARM
Tracking State Sensor - ILM 3 ALARM
Tracking State Sensor - ILM 4 ALARM
Tracking State Sensor - VGR ALARM
Tracking State Sensor - BB ALARM

Figure 5.2: An alarm is triggered by Process Simulate due to a stuck product

6 Conclusion and Future Work

It can be concluded that updating a digital twin is a very costly option in terms of labor and time. Even in most cases, an experienced operator is required since digital twins are generally complicated. Our research has offered a new way to solve this problem: A calibration software that integrates field data with a computer vision-aided camera system for automatic calibration of digital twins.

Different types of experiments are conducted which mainly focused on the performance and speed of the calibration software. It is safe to say that, under some particular circumstances, the calibration software works very accurately and consistently. In terms of precision, the calibration software can estimate the total processing time at an error rate of up to 60 milliseconds. In terms of speed, the threaded structure offers high processing power if a processor is multi-core. Besides that, the drawbacks of the software are mostly related to the computer vision system. If there is not enough brightness or the background has the same color code as a product, color-based object detection methods struggle. Otherwise, it is a perfect tool for the automatic calibration of digital twins. Because, by minimizing manual efforts with this method, cost of many labor hours and possible inaccuracies are correspondingly reduced and prevented.

However, like all other systems, there are many places for improvement. For example, a neural network-based object detection algorithm can be applied to increase detection quality and accuracy in case a product has a similar color code with the background. But this demands a powerful GPU to run smoothly. Furthermore, a more advanced search method can be implemented in Process Simulate Analyzer since it takes a very long time to find and update the parameters in a simulation file. Finally, the calibration software is currently designed to work with only one camera system. But a real plant can be very big, and it would require many cameras to track the motion of a workpiece during the production stage. To overcome this, a cloud-based approach can be utilized in which the camera images are loaded online, and the calibration software would pull the images for object detection.

References

Agapaki, E., Miatt, G. and Brilakis, I. (2018) ‘Prioritizing object types for modelling existing industrial facilities’, *Automation in Construction*, 96, pp. 211–223. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0926580518300670. Retrieved 12.01.2021

AMFG / Autonomous Manufacturing (2019) *Industry 4.0: 7 Real-World Examples of Digital Manufacturing in Action*. Available at: https://amfg.ai/2019/03/28/industry-4-0-7-real-world-examples-of-digital-manufacturing-in-action/. Retrieved 10.01.2021

Biba, E. (2017) *The jet engines with ‘digital twins’*, *BBC Autos*. Available at: http://www.bbc.com/autos/story/20170214-how-jet-engines-are-made. Retrieved 10.01.2021

Björnsson, B., Borrebaeck, C. and Elander, N. (2020) ‘Digital twins to personalize medicine’, *Genome Med*, 12(4). Available at: https://doi.org/10.1186/s13073-019-0701-3.

Breuer, H. (2020a) *Clever assistant for production systems*, *Siemens*. Available at: https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/clever-assistant-for-production-systems.html. Retrieved 12.01.2021

Breuer, H. (2020b) *Digital Twins Everywhere*. Available at: https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/outlook2030.html. Retrieved 12.01.2021

Brilakis, I. *et al.* (2020) *Built Environment Digital Twinning*. Munich.

Cai, Y. *et al.* (2017) ‘Sensor Data and Information Fusion to Construct Digital-twins Virtual Machine Tools for Cyber-physical Manufacturing’, *Procedia Manufacturing*, 10, pp. 1031–1042. Available at: https://www.sciencedirect.com/science/article/pii/S2351978917302767?via%3Dihub.

Cohen, J. (2020) *Computer Vision at Tesla*. Available at: https://heartbeat.fritz.ai/computer-vision-at-tesla-cd5e88074376. Retrieved 02.03.2021

Errandonea, I., Beltrán, S. and Arrizabalaga, S. (2020) ‘Digital Twin for maintenance: A literature review’, *Computers in Industry*, 123. Available at: https://doi.org/10.1016/j.compind.2020.103316.

Fischertechnik GmbH © (2021a) *Automated High-Bay Warehouse 9V - Education*. Available at: https://www.fischertechnik.de/en/products/teaching/training-models/536626-edu-automated-high-bay-warehouse-9v-education. Retrieved 11.04.2021

Fischertechnik GmbH © (2021b) *Indexed Line with two Machining Stations 24V - Education*. Available at: https://www.fischertechnik.de/en/products/teaching/training-models/96790-edu-indexed-line-with-two-machining-stations-24v-education. Retrieved 11.04.2021

Fischertechnik GmbH © (2021c) *Multi Processing Station With Oven 9V - Simulation*. Available at: https://www.fischertechnik.de/en/products/simulating/training-models/536627-sim-multi-processing-station-with-oven-9v-simulation. Retrieved 11.04.2021

Fischertechnik GmbH © (2021d) *Sorting Line With Color Detection 24V - Education*. Available at: https://www.fischertechnik.de/en/products/teaching/training-models/536633-edu-sorting-line-with-color-detection-24v-education. Retrieved 11.04.2021

Fuller, A. *et al.* (2020) ‘Digital Twin: Enabling Technologies, Challenges and Open Research’, *IEEE Access*, 8. doi: 10.1109/ACCESS.2020.2998358.

GE Digital (2021) *Digital Twin*. Available at: https://www.ge.com/digital/applications/digital-twin. Retrieved 16.01.2021

Grieves, M. and Vickers, J. (2017) ‘Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems’, *Transdisciplinary Perspectives on Complex Systems, Springer*, pp. 85–113. Available at: https://link.springer.com/chapter/10.1007/978-3-319-38756-7_4#citeas.

Huang, T. S. (1996) *Computer Vision: Evolution and Promise*. Urbana, IL. Available at: https://cds.cern.ch/record/400313/files/p21.pdf.

i-SCOOP (2021) *Industry 4.0: the fourth industrial revolution – guide to Industrie 4.0*. Available at: https://www.i-scoop.eu/industry-4-0/. Retrieved 16.01.2021

Joordens, M. and Jamshidi, M. (2018) ‘On The Development of Robot Fish Swarms in Virtual Reality with Digital Twins’, in *13th Annual Conference on System of Systems Engineering (SoSE)*. Paris: IEEE, pp. 411–416. doi: 10.1109/SYSOSE.2018.8428748.

Kagermann, H., Wahlster, W. and Helbig, J. (2013) *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0*, *acatech*. Available at: https://www.acatech.de/wp-content/uploads/2018/03/Abschlussbericht_Industrie4.0_barrierefrei.pdf. Retrieved 20.02.2021

Kristie, L. (2017) *Discover fischertechnik for Industrial Simulation and Training*. Available at: https://www.studica.com/blog/fischertechnik-training-simulation. Retrieved 11.04.2021

LandTech (2021) *3D Laser Scanning*. Available at: https://www.landtechinc.com/services/3d-laser-scanning/. Retrieved 02.02.2021

Mihajlovic, I. (2019) *Everything You Ever Wanted To Know About Computer Vision.* Available at: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e. Retrieved 05.02.2021

Parrott, A. and Warshaw, L. (2017) ‘Industry 4.0 and the digital twin’, *Deloitte University Press*, p. 20. Available at: https://www2.deloitte.com/content/dam/Deloitte/kr/Documents/insights/deloitte-newsletter/2017/26_201706/kr_insights_deloitte-newsletter-26_report_02_en.pdf. Retrieved 11.01.2021

Pedamkar, P. (2020) *Difference between Winforms and WPF*. Available at: https://www.educba.com/winforms-vs-wpf/. Retrieved 19.04.2021

Photonics Media (2021) *Machine Vision Cameras: Making the Right Selection*. Available at: https://www.photonics.com/Articles/Machine_Vision_Cameras_Making_the_Right_Selection/a58427. Retrieved 11.03.2021

RF Editorial Team (2018) *Ultra High Frequency (UHF) RFID Tags & Systems*. Available at: https://www.everythingrf.com/community/ultra-high-frequency-uhf-rfid-tags-systems. Retrieved 10.03.2021

Rosen, R. *et al.* (2015) ‘About The Importance of Autonomy and Digital Twins for the Future of Manufacturing’, *IFAC-PapersOnLine*, 48(3), pp. 567–572. Available at:

https://doi.org/10.1016/j.ifacol.2015.06.141.

Russakovsky, O. *et al.* (2015) ‘ImageNet Large Scale Visual Recognition Challenge’, *International Journal of Computer Vision*, 115, pp. 211–252. Available at: https://doi.org/10.1007/s11263-015-0816-y.

Shykora, B. (2020) *Your eyes have a resolution of 576 megapixels*. Available at: https://www.keremeosreview.com/trending-now/morning-start-your-eyes-have-a-resolution-of-576-megapixels/. Retrieved 11.03.2021

Siemens (2021) *Concept to realization: Digital Twin in Manufacturing*. Available at: https://www.plm.automation.siemens.com/global/en/webinar/digital-twin-in-manufacturing/68561. Retrieved 17.01.2021

Siemens Digital Industries (2021a) *Plant Simulation & Throughput Optimization*. Available at: https://www.plm.automation.siemens.com/global/en/products/manufacturing-planning/plant-simulation-throughput-optimization.html. Retrieved 01.02.2021

Siemens Digital Industries (2021b) *Process Simulate Tecnomatix: realise innovation of your products and manufacturing processes*. Available at: https://oneplm.com/tecnomatix-2/. Retrieved 01.02.2021

Siemens Industry Software (2011) *Process Simulate*. Available at: https://www.plm.automation.siemens.com/en_gb/Images/7457_tcm642-80351.pdf. Retrieved 01.02.2021

Siemens Industry Software (2020) *Process Simulate Human*. Available at: https://www.dex.siemens.com/plm/tecnomatix/process-simulate-human. Retrieved 01.02.2021

Siemens RT (2021) *Digital Twins / Simulation at Siemens*. Available at: https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/digital-twin.html. Retrieved 01.02.2021

Szeliski, R. (2010) *Computer Vision: Algorithms and Applications*. Springer. Available at: https://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf. 11.02.2021

Thomas Publishing Company (2021) *Material Handling Equipment (Types, Applications and Suppliers)*. Available at: https://www.thomasnet.com/articles/materials-handling/material-handling-equipment/.

Retrieved 01.03.2021

Timilsina, B. (2012) *Removing Bottleneck From a Manufacturing Unit*. Central Ostrobothnia University. Available at: https://core.ac.uk/download/pdf/38062633.pdf. Retrieved 05.03.2021

W. Cearley, D. *et al.* (2017) *Top 10 Strategic Technology Trends for 2018*. Available at: http://brilliantdude.com/solves/content/GartnerTrends2018.pdf. Retrieved 07.01.2021

Wehrstedt, J. C. *et al.* (2020) *A Framework for Operator Assist Apps of Automated Systems*.

Zarchan, P. (2012) *Tactical and Strategic Missile Guidance*. 6th edn. American Institute of Aeronautics and Astronautics, Inc. Available at: https://doi.org/10.2514/4.868948. Retrieved 05.02.2021

Zhou, Y. *et al.* (2019) 'Flexible Architecture to integrate Simulation in Run-Time Environment', in *AUTOMATION 2019*, pp. 1–13. Available at: https://www.researchgate.net/publication/334635916_Flexible_Architecture_to_integrate_Simulation_in_Run-Time_Environment. Retrieved 20.02.2021