TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Maksim Moissejev 192525IASM

# 3D crane control using ABB PLC and Computer Vision

Master's thesis

Supervisor: Kristina Vassiljeva
PhD

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Maksim Moissejev 192525IASM

# 3D kraana juhtimine ABB PLC ja masinnägemise abil

Magistritöö

Juhendaja: Kristina Vassiljeva
PhD

Tallinn 2022

# Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Maksim Moissejev

02.01.2022

# Abstract

Humanity is on the verge of the fourth industrial revolution, which will be centered on using a data-driven approach for system control and composing networks of sophisticated machinery. The problem is that replacing the current machinery requires significant investments. Cranes play an important role in the industry because there is a constant need to move heavy and bulky objects back and forth. Computer Vision is regarded as the most effective technology for acquiring information about the crane workspace. The aim of this thesis is to augment the existing crane control system with Computer Vision technology in a cost-effective way. To evaluate the viability of the proposed solution, an overhead crane laboratory model provided by Inteco is used.

In this thesis, the proposed system architecture and the implemented control loop are discussed. The Computer Vision part of the system, as well as the underlying algorithms, are thoroughly explained. The Computer Vision component was implemented using two different approaches, and the results were evaluated and compared. An interactive and user-friendly application was developed to simplify the crane operation process. The proposed system was tested in the laboratory environment, and the performance analysis is presented at the end of the paper.

This thesis is written in English and is 87 pages long, including 6 chapters, 29 figures, 7 tables, and 2 appendices.

# List of abbreviations and terms

| | |
|---|---|
| ADU | Application Data Unit |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| CRC | Cyclic Redundancy Check |
| CV | Computer Vision |
| DC | Direct Current |
| DL | Deep Learning |
| DNN | Deep-Neural Network |
| FOV | Field of View |
| GAN | Generative Adversarial Network |
| GPU | Graphics Processing Unit |
| GUI | Graphical User Interface |
| HMI | Human Machine Interface |
| HSV | Hue, Saturation, Value |
| I/O | Input/Output |
| I4.0 | Industry 4.0 |
| ID | Identifier |
| IIoT | Industrial Internet of Things |
| IP | Internet Protocol |
| KB | Kilobyte |
| MBAP Header | Modbus Application Header |
| OCR | Optical Character Recognition |
| OS | Operating System |
| PAC | Programmable Automation Controller |
| PC | Personal Computer |
| PDU | Protocol Data Unit |

| | |
|---|---|
| PID | Proportional-Integral-Derivative |
| PLC | Programmable Logic Controller |
| PWM | Pulse Width Modulation |
| RAM | Random Access Memory |
| RGB | Red-Green-Blue |
| RoI | Region of Interest |
| SCADA | Supervisory Control and Data Acquisition |
| ST | Structured Text |
| TPU | Tensor Processing Unit |

# Table of contents

# List of figures

# List of tables

# 1 Introduction

As technology advances, more and more processes are becoming automated. In industry, cranes are of a vital role since there is a huge need to move heavy and bulky objects in different directions. Previously, the crane operator had to be highly qualified and experienced. The reason for this is that the work is hazardous to both the environment and the people in the crane's operating area. However, modern cranes are equipped with a variety of electronic components that assist the operator and alert him in case of abnormal scenarios. Furthermore, due to software algorithms commonly referred to as “Artificial Intelligence” (AI) and significantly increased computational power, the process of operating the crane begins to resemble a gaming process. More broadly, the industry as a whole is seeing a shift toward control gamification.

AI is an umbrella term that encompasses a wide range of research activities, including computer vision (CV), natural language processing, machine learning, deep learning (DL), etc [1]. In this paper, the author attempts to use CV technology to automate crane control operations. An overhead crane laboratory model provided by Inteco is used to evaluate the feasibility of the proposed solution. Section 3.1 provides a more detailed description of the experimental setup. Overhead cranes are typically used in indoor environments such as warehouses or assembly lines. However, there are also overhead cranes (with slightly different construction) that are used in outdoor environments, such as harbors or agriculture. The proposed solution is intended to be used indoors, where the environment is more predictable in terms of background noise, lighting conditions, camera viewpoint, and visual obstructions.

One of the most significant advantages of CV technology is its ability to augment existing systems and make them more automated or even autonomous at very low cost. It means that by incorporating a visual system into the crane control system, the degree of its automation can be increased. Cranes can be classified into three types based on their degree of automation: manual, semi-automatic, and fully automated. The majority of cranes used in the industry today are semi-automatic. They assist the operator with various features such as payload oscillation control, but still require manual control actions from the operator. However, fully autonomous cranes are also available today. To name a few solutions:

- Integrated Autonomous Crane System (IACS) – developed by Schneider Electric [2]
- Optilift Autonomous Crane Control System – developed by VOCA [3]
- ForeSite100™, AutoSite™ technologies – developed by Intsite [4]
- Konecranes offers a CV-based solution as an optional feature on its cranes [5]

Relying on AI, the operator can delegate crane control to the computer system to handle repetitive or difficult tasks. The main advantages of this approach are that the settings can be changed remotely and that the crane's operation can be monitored remotely. It is especially important in demanding and hazardous environments.

Previously, as part of his Bachelor's thesis [6], the author developed an application to control the Inteco 3D crane [7] with the ABB AC500 Programmable Logic Controller (PLC). The main task was to replace the original RT-DAC/PCI multipurpose digital Input/Output (I/O) board, which is managed by the 3DCrane Toolbox (integrated into the MATLAB®/Simulink® environment) – with the PLC (the device that is more common in industrial environments). The secondary task was to configure Proportional-Integral-Derivative (PID) controllers to reduce the payload oscillations and make the cart movement smooth.

In this work, the author attempts to augment the existing crane control system with CV technology. The main motives are:

- Crane control simplification
- Routine actions automation
- Remote control possibility

The resulting system aimed to be compatible with the Industry 4.0 (I4.0) standards. The proposed implementation should be cost-effective.

The paper is organized as follows. Section 2 discusses the phenomenon of Industry 4.0 and the applicability of CV technology for crane automation. Section 3 provides an overview of the proposed system and describes the implemented control loop. The protocol used for communication between the Personal Computer (PC) and PLC, as well as the configuration process for both nodes, are also covered in this Section. Section 4 discusses the advantages of both DL-based and traditional CV approaches. The proposed solutions, which are implemented using these two methods, are also described here. Section 5 is devoted to the developed crane control application. The communication logic between the PC and PLC, as well as their responsibilities, are described in detail. Finally, Section 6 summarizes the results provided by the proposed system and formulates the goals for future development.

## 1.1 Motivation

CV is becoming a part of our everyday lives as it finds an application in a wide range of areas. The author believes that AI in general, and CV in particular, have huge potential for automation of industrial processes. This is a relatively new and in-demand research area with a lot of challenging and practical problems. Recent breakthroughs in DL give reason to believe that success in this field will bring humanity into the new era of autonomous manufacturing.

## 1.2 Thesis contributions

This thesis makes the following contributions:

- Computer Vision – corner points extraction algorithm for Inteco 3D crane frame is developed. Two methods are evaluated and compared. The first method employs the traditional CV approach, whereas the second uses the trained Mask R-CNN model. The parameters used in the classic CV implementation are designed to be adjustable via the Graphical User Interface (GUI). The parameter sets can be stored in a database.

- Control – the communication logic between the PLC and PC is established, taking into account their significantly different program execution speeds. The room for future development is preserved. The proposed control loop follows the Industry 4.0 operating principle.

- PLC programming – the program code developed within the author's Bachelor thesis is completely rewritten. The Modbus TCP protocol is now supported for communication.

- GUI – all of the information and controls required to operate the crane are contained within the GUI. The information is displayed over the camera's captured frames. It is presented in an unambiguous, legible, and redundant manner.

The CV-based system for controlling the Inteco 3D crane is implemented. The proposed architecture is suitable for managing an indoor warehouse.

# 2 Background

As evidenced by consumer electronics, technologies are evolving at a rapid pace. Terms such as “smart home”, “self-driving car”, or “virtual reality” are becoming more common in our daily lives. Today’s smart gadgets come with software that allows swapping faces, augmenting reality, and there is a lot more entertainment available in the Play or App Store. The hardware required to run the software is becoming more affordable and sophisticated. The part of the story that is usually left out of the media is that industrial processes are also evolving due to advancements in the same research area. Similar technologies, however, are hotly debated in industry-specific resources, and at various conferences and forums, but in a slightly different context. This chapter discusses the ideas behind the term Industry 4.0 and the applicability of CV for crane control automation.

## 2.1 Industry 4.0

A growing number of researchers are discussing so-called Fourth industrial revolution, or Industry 4.0. The term is well accepted among both academics and industrial society [8]. However, to the best of the author's knowledge, there is no clear and comprehensive definition of what this buzzword means. This term is defined differently by various authors. Some of the notable ones, in the author’s opinion, are presented in Table 1. According to the definitions provided, most authors associate I4.0 with sophisticated machinery with a high level of autonomy. The majority also highlights the importance of effective, flexible, and standardized data exchange methods. Some authors stated that it is important to plan, predict, and control the entire process within the entire business, and that AI could be remarkably useful in this case. I4.0 is also associated with higher quality products, which have a high degree of customization potential. One author even stated that the lines between the real world and virtual reality will become blurrier.

Table 1. Definitions of Industry 4.0

| **Year of publication** | **Definition** |
|---|---|
| 2020 | I4.0 is a manufacturing philosophy comprised of complex automation systems with a high level of autonomy and flexible data exchange methods that will lead to the next level of manufacturing technology. This will allow for more agile and personalized production [8]. |
| 2017 | I4.0 will be achieved through the integration of sophisticated machinery and devices with advanced software and networked sensors. The result of the integration could be used to improve planning, predicting, and control, resulting in better business and societal outcomes. It will improve the value chain organization and simplify the management throughout the product lifecycle. The optimization has to be achieved through autonomous control and dynamic production [9]. |
| 2016 | The central idea of I4.0 is to use emerging technologies and the rapid development of machines and tools to keep up with the progressive improvement of human life quality by providing customers with customized, higher-quality products. Production can now run faster and with less downtime thanks to the use of information technology. The production systems will be more efficient and easier to maintain, resulting in cost savings for the company [10]. |
| 2016 | I4.0 will be achieved by forming a new kind of intelligent, networked, and agile value chain. The technologies, on which the integration process will rely are service automation, AI, robotics, internet of things, and additive manufacturing. The lines between the real world and virtual reality will become blurrier over time, resulting in a phenomenon known as cyber-physical production systems [11]. |
| 2016 | I4.0 is based upon two key design principles: interoperability and consciousness. The interoperability consists of digitalization, communication, standardization, flexibility, real-time responsibility, and customizability. It serves as a "connecting bridge" to ensure a reliable manufacturing environment. The consciousness is composed of predictive maintenance, decision making, intelligent presentation, self-awareness, self-configuration, and self-optimization. I4.0 is about intelligent manufacturing, which means that it discovers knowledge, makes decisions, and performs actions independently and intelligently. It is achieved by gathering raw data from manufacturing networks and intelligently analyzing it [12]. |

In the simplest terms, the basic idea of I4.0 is to collect data throughout the organization, to have access to that data from anywhere, and, most importantly, to use the data. It implies the presence of built-in sensing devices in almost all manufacturing components, equipment, and, in some cases, in products. The key difference between the Fourth industrial revolution and the previous one is that the focus is shifting from individual computerized machines to the entire network of them. It is assumed that the autonomous smart machines, each equipped with different sensors, will exchange data with one another, forming an intelligent or smart factory. However, the idea of I4.0 goes beyond separate smart factories. The fourth industrial revolution is supposed to improve the entire supply chain by interconnecting its components. In practice, this could mean that ships exchange data with warehouses, which are then linked to autonomous trucks, and so on. The ultimate goal, as with all previous industrial revolutions, is increased efficiency, lower production costs, and higher product quality [12].

In summary, the Fourth Industrial Revolution is a very complex and broad phenomenon that can be defined and understood differently depending on the perspective from which it is viewed. However, it is a fact that it will have a significant impact on the entire society, and the way people live and think. The main source of concern about the ongoing revolution is economic factors. The problem is that the industrial machinery that is currently in use is very expensive, and large investments are required to renovate the entire factories (in contrast to consumer electronics, which initially assumes a relatively short service life). Such heavy investments come with the risk of going bankrupt, which is especially crucial for small and medium enterprises.

As previously stated, one of the advantages of CV technology is that it allows for the low-cost enhancement of existing systems, as demonstrated in [13]. The authors of this paper proposed a solution to monitor tool wear in process. They used a camera and CV to inspect the drill of a CNC machine. Another CV-based solution is intended to monitor and collect data from the Human Machine Interface (HMI) of any device with a proper display. This project is notable for its use of only open-source solutions and low-cost equipment [14]. These two examples show how I4.0 cornerstones such as predictive maintenance and continuous data collection could be realized by simply adding a camera and the corresponding software. It is true that CV cannot guarantee one hundred percent correct results and that errors may occur. However, such an approach could be viewed as a good short-term strategy for small and medium enterprises trying to mitigate the risks associated with heavy and single-entry investments.

## 2.2 Application of CV for crane control

As previously noted, cranes are widely used in industry. They can be found in almost any sector that requires transportation. Cranes are used on land and in the water. Furthermore, certain activities, such as skyscraper construction, cannot be carried out without the use of a crane. Cranes come in a variety of shapes and sizes, each with its own intent [15]. In the scope of this thesis, overhead cranes are considered. To be more specific, overhead cranes with an incorporated runway system (see Figure 1). These cranes are typically used indoors, and the runway beams and support columns are tied to the building's support structure. The crane construction is quite simple, consisting of parallel runways connected by a traveling bridge. The bridge holds one or more hoists that move along it.

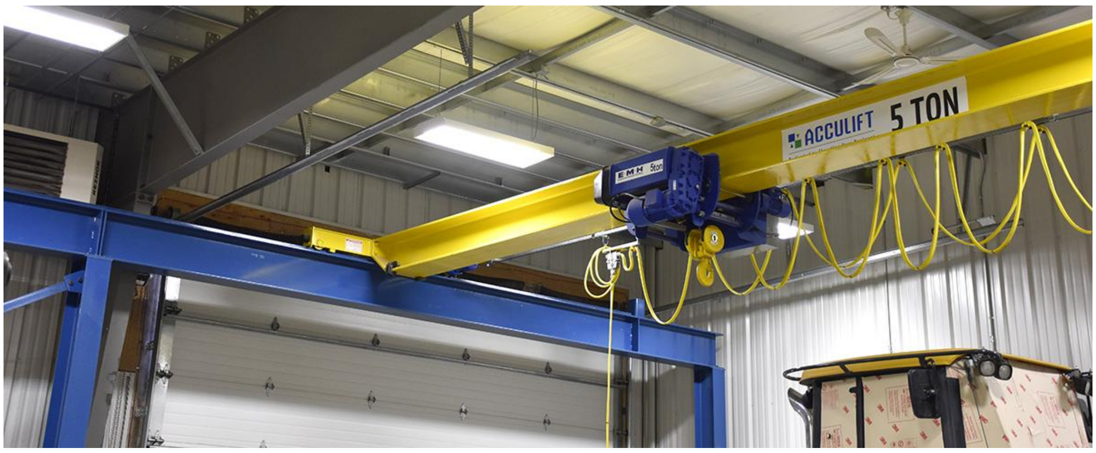


Figure 1. Industrial overhead crane [16]

The idea of automating crane operations is not new. Hyla defined the requirements for automating crane operations in his survey in 2012. According to Hyla, the crane control system should be capable of [17]:

- Precise positioning of a load from its origin to its destination.
- Motion speed control should be used to reduce the sway of a payload during movement.
- Crane skew angle reduction.
- Real-time obstacle detection and identification of the operating workspace.
- Calculation of a safe and time-optimal path (avoiding detected obstacles).
- The selected path tracking for detecting obstacles in the next real-time.

- Synchronization of tasks with other devices in the same operating area.

He also stated that vision systems are perfect for gathering information about the crane workspace and potential obstacle dimensions. However, CV technology was still in its infancy at that time, and other researchers discussed CV-related drawbacks. They mentioned issues such as sensitivity to background noise and lighting conditions. Furthermore, discussed issues included crowded, harsh, changing environments, etc [18].

In 2016, researchers in the field discussed already the issues associated with the rapid positioning and tracking of small and key targets [19]. In this work, the authors attempted to detect container corner castings precisely. They stated that among researchers, CV-based crane control is the most popular way to automatically load and unload port containers, and that vision systems outperform earlier encoder-based positioning solutions in terms of efficiency and flexibility. In [20], two CV-based methods for crane workspace mapping were proposed. The first relies entirely on CV algorithms, while the second relies on QR codes. Both methods work in near-real-time and could be combined. According to the author's findings, QR codes have a high potential for widespread use in crane workspace mapping.

All of the preceding examples are based on so-called traditional CV (algorithmic approach in CV). However, due to a recent huge leap in the development of Convolutional Neural Networks (CNN), combined with significantly increased computing power, more and more researchers are trying to adopt different Deep-Neural Network (DNN) architectures for various tasks. For instance, in [21], an assistive early warning system for crane operator was proposed. It employs the Mask R-CNN model. The vision system is capable of calculating the safe distance to any specified object with high precision. The system has the potential to increase safety during crane operations.

When compared to systems powered by traditional CV techniques, vision systems powered by CNNs have higher accuracy and are more flexible. Moreover, they are more tolerant to lighting conditions, camera viewpoint, background noise, visual obstructions, and intra-class variations. In terms of performance and reliability, the new approach allows for the same results to be obtained using a regular camera and middle-level hardware rather than a specialized system. However, it cannot be claimed that DNN is a panacea for all problems since traditional CV has its own advantages, which will be discussed in Section 4.1.

For quite a long time, researchers have been looking for ways to make the crane operator's job easier. In 2009, a crane control method that uses a wand as a controller was implemented [22]. The proposed solution allows an operator to use a wand to set the desired position for a crane hook. According to the authors, the proposed solution significantly increases operating efficiency when compared to standard pendant control. It is intended to assist the human operator while also improving workplace safety.

In general, the modern approach to crane control systems considers the operator as a subject of the weak link. The reason for this is that the operator is exposed to unfavorable effects such as vibrations and a high level of noise. Furthermore, his work becomes more difficult in some cases due to high (or low) temperatures, toxic fumes, high pollution, or air dustiness. To the foregoing is added the nature of crane operation itself, as well as the fact that the operator's daily work requires continuous concentration. These negative factors make it important to find preventive solutions aimed at improving the operator's concentration, assisting him in making correct decisions, and avoiding so-called sensory deprivation. A vision system might help a lot with this by gathering and presenting information in an unambiguous, legible, and redundant manner [23].

# 3 System description

This chapter discusses the proposed system architecture. The hardware components and overall control loop are described first. Following that, the rationale for selecting the communication protocol is presented. Finally, the process of configuring two key nodes in the proposed architecture is described.

## 3.1 Experimental setup

The experimental setup used to evaluate the feasibility of the proposed solution consists of six nodes. Figure 2 shows a graphical representation that illustrates the system architecture.

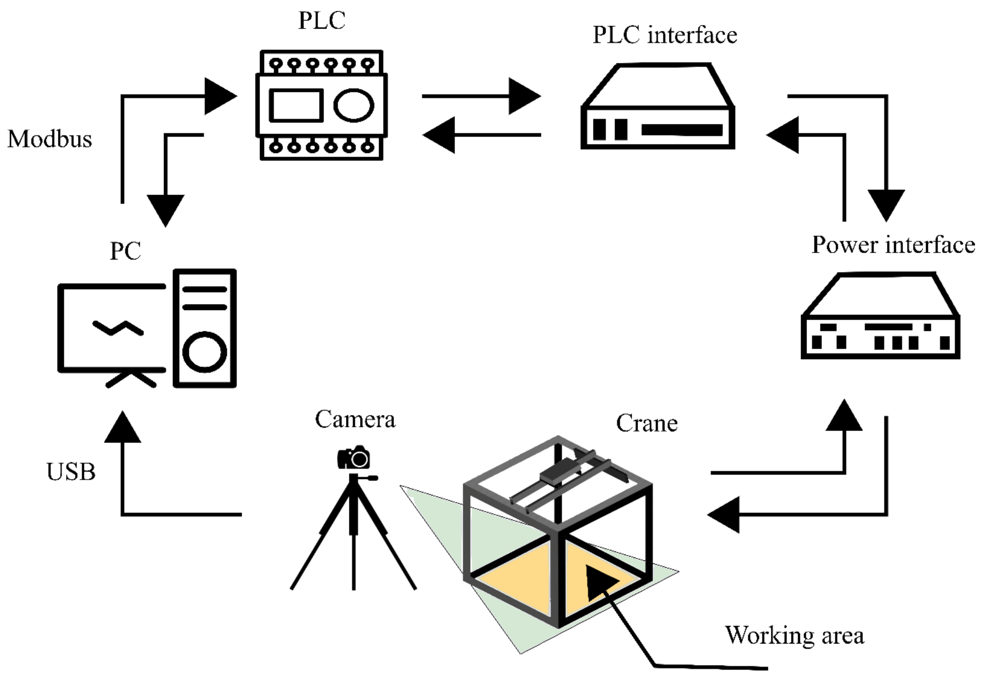


Figure 2. System architecture

The proposed architecture assumes that the camera is installed near the crane's frame. The entire crane's working area should be visible in the camera's field of view (FOV). For simplicity, the placement of the camera was limited to one side. It should be installed against the side with the crane's "home" position. Figure 3 shows a possible camera's FOV.

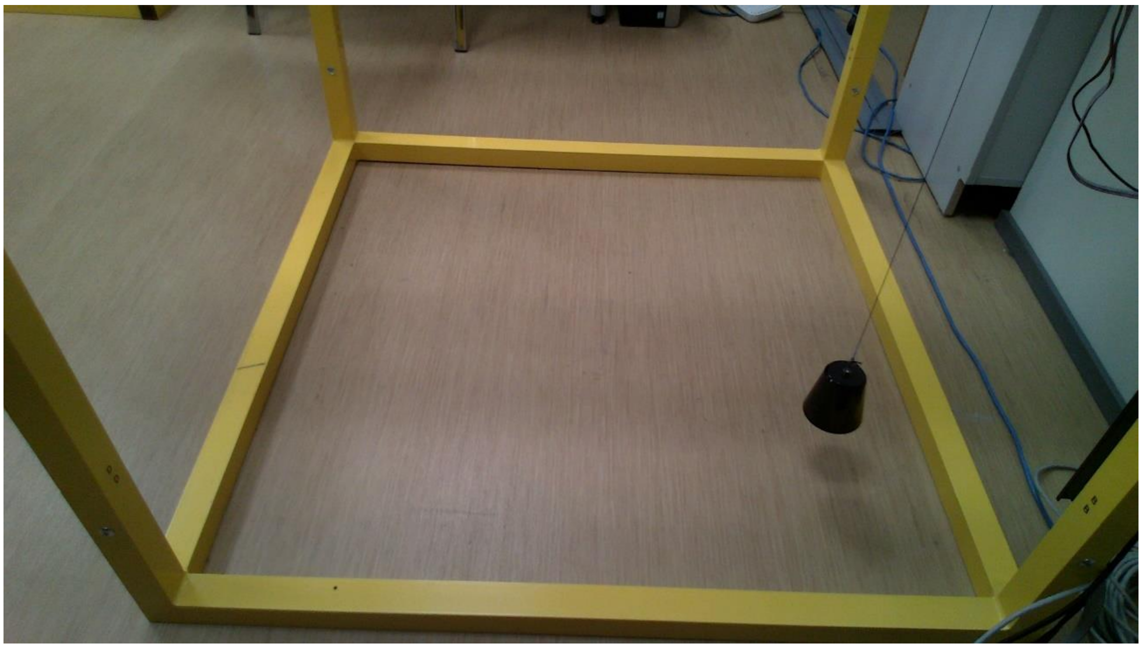


Figure 3. Possible camera FOV

The proposed system operates according to the following principle. The color images captured by the camera are sent to the PC via the USB cable for processing. The PC executes the algorithms required to extract all useful information from the image, display it, and send corresponding control signals to the PLC, which controls the crane's motors. For communication between the PC and PLC, the Modbus TCP protocol is used. The PLC is programmed as a secondary controller and is intended to receive and respond to commands. All time- and safety-critical functionality is implemented on the PLC since it is more reliable and has a higher speed of program execution. The connection between the rest of the nodes is established via electrical wires and cables. The detailed description can be found in the author’s Bachelor thesis [6].

Table 2 provides a summary of the devices used in the proposed system. The column unit reflects the name used in the overall system architecture (see Figure 2). It is important to note that, despite the fact that the camera used in the architecture has built-in LiDAR, it was decided to rely on the Red-Green-Blue (RGB) sensor only, as the author's interest is in finding a relatively cheap solution.

Table 2. Summary of devices used in the experimental setup

| Unit | Description |
|---|---|
| Camera | **RGB camera** (Intel® RealSense™ LiDAR Camera L515 [24])<br>– *Frame resolution*: Up to 1920 × 1080<br>– *Frame rate*: 30 fps (at max. resolution)<br>– *FOV*: 70° × 43° (±3°)<br>– *Focus*: Fixed |
| PC | – *GPU*: NVIDIA GeForce GTX 980 Ti<br>– *CPU*: Intel Core i7-6700K @ 4.00GHz<br>– *Memory*: 32GB RAM<br>– *OS*: Windows 10 64-bit |
| PLC | **ABB AC500** [25], with the following modules:<br>– *PM590-ETH* [26]: Processor module (Memory 2MB, Serial and Ethernet interfaces)<br>– *TA524* [27]: Dummy coupler module<br>– *DA501* [28]:Digital/Analog I/O Module (16 digital inputs, 8 configurable digital I/O, 4 analog outputs, integrated fast counter)<br>– (2) *CD522* [29]: Encoder and PWM module (2 encoder inputs, 2 PWM outputs, 2 digital inputs, 8 configurable digital I/O) |
| Crane | **Inteco 3D Crane** [30]:<br>– *Dimensions*: frame (1000 × 1000 × 1000 [mm]), working area (920 × 920 [mm]), lift-line length 920 [mm]<br>– *Hardware*: Safety button, 3 DC motors, 3 position limit switches (indicate „home“ position), 5 incremental encoders (measuring the cart coordinates on the x, y plane, the load's lift-line length, and the load deviation angle in the x, y directions) |
| PLC interface | The interface that provides conversion between the digital I/O voltage standard accepted by the PLC and the voltage standard of the power interface [31]. |
| Power interface | The interface that amplifies the control signals sent from the PLC to the DC motors and converts encoder pulse signals to a digital 16-bit format that the PLC can read [7]. |

## 3.2 Control loop

The overall control loop of the system is meant to be running on two controllers. These controllers are the PC and PLC (see Table 2). As mentioned in Section 3.1, the PC receives color images captured by the camera, processes them to extract useful data, provides all necessary information to the operator, and sends parameters to the PLC in near real-time. The corresponding loop is referred to as the “Outside loop” in the scheme presented in Figure 4. The iteration time of this loop is much longer compared to the second loop, which is titled “Inside loop”, which is meant to run on the PLC. During the "Inside loop" loop, all received parameters are processed and new ones for controlling the crane's motors are generated. The PLC also sends feedback to the PC. It contains the current load position, as measured by the crane's incremental encoders (see Table 2), and the bits that indicate the current crane status. A more detailed description of these bits and communication logic can be found in Section 5.1.

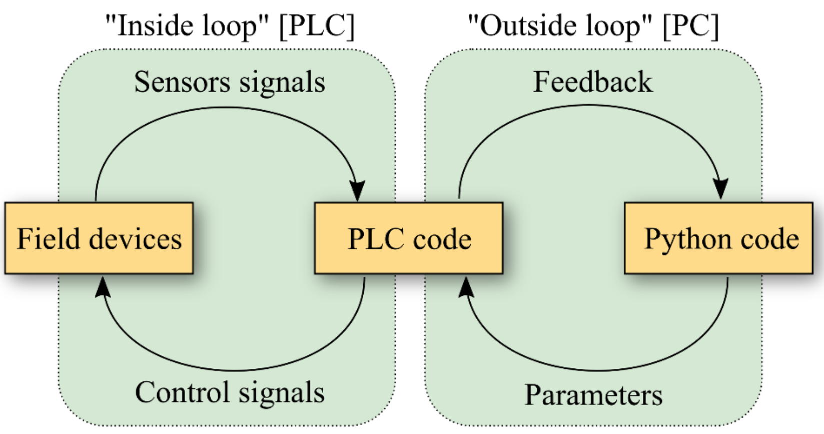


Figure 4. Control loop

### 3.2.1 “Inside loop”

The inside loop is implemented in Structured Text (ST). ST is a PLC programming language, supported by IEC-61131. This standard describes in detail, what is a PLC and what are the requirements that PLC must satisfy. The third part of the IEC-61131-3 specifies five programming languages [32]. Among them, ST is the most widely used in Europe [33]. All these languages allow for the programming of PLC to operate in real-time. This means that the PLC will provide fast, reliable, and deterministic results, or it will respond to the command within a specified time window. The execution time of each instruction within the code is always almost the same, with a small amount of variation. This is very important, especially if the PLC is performing safety-critical duties. However, it is also important in common cases. For example, while performing his duties, the crane operator expects an immediate response to his commands. These are the main reasons why PLCs are still widely used in the industry. Such behavior is largely determined by the PLC operating principle. It is quite simple, straightforward and consists of three main steps [34]:

- First, the PLC checks the status of all inputs and saves the values to the Random Access Memory (RAM). When the binary value representing all of the inputs is in the RAM, the user program starts to run.
- Second, the PLC executes the program and prepares the output status based on the logic of the program.
- Finally, the PLC stores the binary number representing the output status in a special location in memory. This location always keeps the PLC's output status. As a result, the outputs, which are typically connected to field devices, are updated with the new values.

These three steps are executed cyclically and an iteration is called a scan cycle. The time the scan cycle takes is called scan time and it is typically measured in milliseconds, depending on the complexity of the user program. Such an operating principle makes the PLC especially good for handling an interface with sensors and actuators, as well as performing time- or safety-critical tasks.

Considering the experimental setup (see Figure 2) during the “inside loop” the PLC communicates with all the sensors and actuators. More specifically, it reads encoder signals, position limit switches, and, if necessary, controls corresponding Direct Current (DC) motors by generating an appropriate Pulse Width Modulation (PWM) signal. Because PLCs are good for doing time-critical jobs, the PID controllers that control the load oscillations while the cart is moving are also implemented on PLC. All of this functionality is safety-critical. In the case of an experimental setup, unexpected behavior may damage the equipment, in the real-world scenario, it may be dangerous to any object within the crane's operating area, including people. Another duty of the PLC is to maintain the list of Modbus registers.

### 3.2.2 “Outside loop”

The outside loop is implemented in Python. Python is a high-level, dynamic, and interpreted programming language. It is general-purpose and it focuses on code readability, making it especially good for prototyping [35]. Python has a great community and it is actively used by tech giants such as Facebook, Alphabet, Apple, Dropbox [36]. Due to its popularity, it has a great amount of diverse third-party libraries. This significantly reduces the time, required for project development, since most of the highly used programming tasks are already scripted. Among the reasons, why Python was selected to program the “outside loop”, the most important one is its ecosystem, with the modern tools and methods for effective data analysis, machine learning, and so on. However, Python is not a panacea and it has its own drawbacks. The most important ones are that Python has a relatively slow speed of execution and that it is considered as not a memory-efficient language [37]. Therefore, to implement a real system, it is better to use more favorable in these terms language, for instance, C++.

Considering the experimental setup (see Figure 2), during the "outside loop", the PC performs the following duties. It runs the algorithms required to extract all of the useful information from the video stream captured by the camera. Within this loop, the PC interprets the feedback from the PLC. It displays all the relevant information on the GUI. As a primary controller, the PC wraps and sends user commands to the PLC.

### 3.2.3 Relevance of the control logic under consideration

The control loop, discussed in the Sections above is becoming increasingly common in factory control systems. Typically, during the “inside loop” all the time- and safety-critical functionality is performed and the interaction with all the peripheral devices happens. The “outside loop” is mostly intended only to read out certain values or parameters at specific time intervals, store them, and analyze them. However, in some cases, the “outside loop” also gets to tweak certain parameters. Such an operating principle is what are the buzzwords “Industrial Internet of Things” (IIoT) or “Industry 4.0” are practically all about.

Continuous data collection with its further analysis has a great number of benefits and the industrial applications can be categorized into three domains: monitoring, optimization, and control [38]. An example from the monitoring domain is the concept of predictive maintenance that is growing rapidly right now in the factories. For instance, it is possible to continuously monitor the vibration of a motor and then based on that predict when it is going to fail. Another type of problem that might be solved effectively by taking a data-driven approach is scheduling or process planning. They belong to the optimization domain. Modern factories are complex networks of goods, data, and labor. To make them show up in the right place at the right time is a holistic and multilayered problem. The right use of data might optimize the entire workflow. Examples from the control domain are robotics, autonomous vehicles, or smart grids. Even the entire factories or warehouses, might be automated. This approach is known as lights-out manufacturing or “smart factory”, where different types of robots move materials and assemble goods autonomously, under the control of a supervisory system.

A growing amount of data makes the selection of a communication protocol especially relevant. However, it is not an easy task, since in the “field” many factors must be considered. To the common problems, such as vulnerabilities, reliability, or speed of communication, industry-specific challenges are added. Very often engineers have to consider that in parallel with the brand-new machines work their distant predecessors. Moreover, when it comes to the industry, there are a lot of various policies, regulations, acts, etc.

In the context of this project, there is more freedom of choice. The major constraints for the protocol came out from the PM590-ETH combability (see Table 2) and its implementation possibility in Python. Other requirements are that the selected protocol should provide sufficient reliability and speed of communication. Preferably, the protocol should be free of charge, since the author’s interest lies in the finding of a cheap solution. The future-looking requirement is that the protocol should be widely used. It will make the proposed system more flexible and friendly with other systems.

## 3.3 Communication protocol selection

A huge number of industrial communication protocols exist. They all fall into two groups. The first category is the ones that require specialized hardware, and the second category uses standard ethernet networking hardware. Modbus is the one that does not require any specialized hardware. It is the oldest and one of the most popular automation protocols in the field of process automation [39]. Modbus is an open protocol, meaning that manufacturers can deploy it into their equipment without the need to pay any royalties. Since the official Modbus specification is publicly available, it is supported with libraries in many programming languages, including Python [40].

Modbus is a master/slave protocol, meaning that it is unidirectional. The master (or client) is the device that sends requests over the network in order to read or write the values stored in the “slave” device. The slave (or server) is the unit that keeps and maintains a list of values and allows other devices to read or change them through the network requests. The server does not initiate requests on its own and only responds to the requests transmitted from the client. Typically, the client is a PLC, Supervisory Control and Data Acquisition (SCADA) system, or accordingly programmed PC. The role of the server might be a sensor, another PLC, or programmable automation controller (PAC) [41]. In other words, Modbus is used to transmit signals from the field and control devices back to the main controller or data gathering system.

There are many versions of the Modbus protocol and the most common are Modbus RTU, Modbus ASCII, Modbus TCP, and Modbus Plus [39]. Modbus communicates over the Ethernet or Serial (RS-232, RS-485, RS-422) physical media.

### 3.3.1 Modbus TCP

Considering the experimental setup (see Figure 2), it was decided to use the Modbus TCP version of the protocol. ABB AC500 supports Modbus RTU and Modbus TCP and there were several reasons for selecting the latter. The first one is related to the speed of communication. Since Modbus TCP uses Ethernet, it provides a faster communication speed. Moreover, Ethernet is considered to be a more reliable and flexible communication technology [42]. The cost factor, that could have been decisive in the past is now less relevant since the required hardware is ubiquitous and became more affordable [43]. Another reason is the future looking. Compared to Modbus RTU, Modbus TCP is not limited to a single master device. In Modbus TCP additional master devices will not destroy the whole network, which is the case with Modbus RTU [44]. And the last but not the least reason is that during the system development PLC was programmed through the Ethernet cable, and the use of a single cable for both purposes seemed to be more convenient.

### 3.3.2 Description of Modbus TCP

Modbus TCP is essentially Modbus RTU, packaged with a TCP/IP interface that runs on Ethernet. When comparing the formats of telegrams (see Figure 5), it can be seen that the Slave Identifier (ID) address at the beginning and the Cyclic Redundancy Check (CRC) checksum at the end are omitted. Instead, the Modbus Application Header (MBAP Header) is added to each message. It contains all of the identifying information required to route data to the addressed device. The checksum calculation is not needed, because lower layers already provide mechanisms to verify an accurate delivery of a packet. Modbus TCP uses port 502 [45], [46].

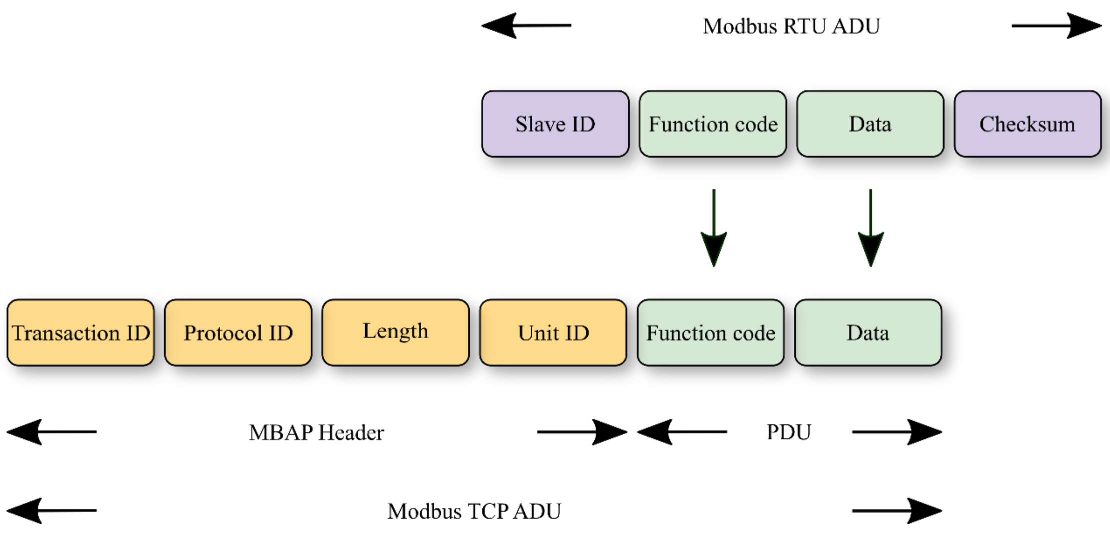


Figure 5. Modbus TCP telegram

The Modbus TCP message consists of Protocol Data Unit (PDU) and MBAP Header. As previously stated, the MBAP Header contains all of the information required to route data. It is 7 bytes long and keeps the following data [46], [47]:

- 2 bytes: Transaction ID – These bytes are set by the client and are required for synchronization between the server and client messages. The server repeats these bytes in the response because the order of the server's responses is not always the same as the order of the client's requests.
- 2 bytes: Protocol ID – These bytes are set by the client, and according to Modbus TCP protocol, should always be 00 00.
- 2 bytes: Length field – These bytes are set by the client. They indicate the number of bytes that will follow in this message, starting with the Unit ID.
- 1 byte: Unit ID – This byte is set by the client. It is used to identify the server to which the request is directed. In its response, the server repeats this byte.

The second part of the message is the PDU. The PDU consists of the function code, followed by an associated set of data. The size and contents of this field vary and depend on the function code. The function code itself is one byte long and tells the server what kind of action to perform. The size of PDU is limited to 253 bytes [41]. This constraint was inherited from the first Modbus implementation on the Serial line network. According to the Modbus specification v1.1-b3 [48], the maximum size of the RS485 Application Data Unit (ADU) is 256 bytes. One byte takes the Slave ID, and two bytes represent the CRC field. Therefore, the maximum size of Modbus TCP ADU is 260 bytes.

### 3.3.3 Modbus TCP vulnerabilities

Despite all of the advantages of the Modbus TCP protocol and its widespread use in the field of process automation, it has several drawbacks. Modbus TCP protocol contains multiple vulnerabilities that could result in data leakage or allow an attacker to change certain system parameters. The following list of vulnerabilities is based on [49]:

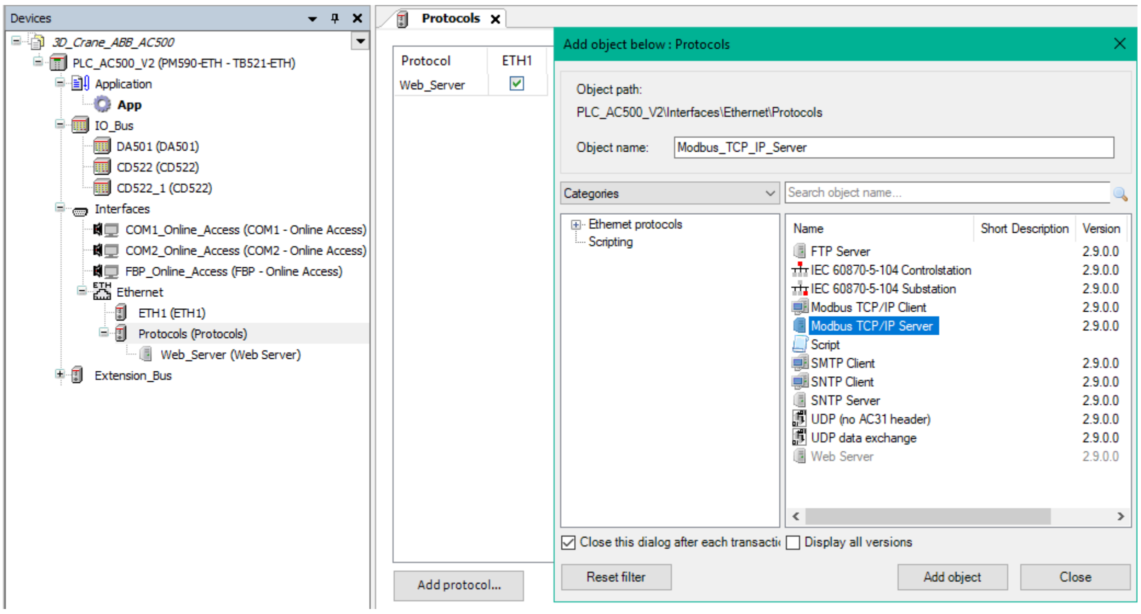
- Lack of Authentication – at any level of Modbus protocol, there is no authentication. For example, when a PC is programmed to send and receive Modbus messages, the PLC considers it as another PLC.
- Simplistic Framing – Modbus TCP messages are sent via TCP connections. These connections are usually reliable, but a TCP connection does not guarantee the message delivery completely.
- Lack of Confidentiality – all Modbus messages are transmitted in clear text over the communication channel.
- Lack of Integrity – in the Modbus application protocol, there are no integrity checks. Hence, the lower layers should ensure the accuracy and consistency of the transferred data.
- Lack of Session Structure – Modbus TCP is made up of short-lived transactions. The client initiates a request to the server, and the server responds, resulting in a single action.

The most straightforward way to attack Modbus devices on the network is to send harmful commands. For instance, an attacker might craft a packet that is longer than 260 bytes and send it to the client or server. As it was discussed in Section 3.3.2, the maximum size of Modbus TCP ADU is 260 bytes. If the attacked device was incorrectly programmed, it might result in a buffer overflow or denial-of-service [49].

### 3.3.4 AC500 configuration as Modbus TCP server

The AC500 PLC can be configured to operate as both a server and a client. The server operating mode should be set, according to the proposed architecture (see Figure 2). The configuration takes place in the vendor-specific PLC programming tool ABB Automation Builder [50]. During the project development, version 2.3.0.847 of the software suite was used.

The first step is to create and set up a new project, as described in the author's Bachelor's thesis [6] (Section 3.1). All of the configurations needed to interact with the crane's hardware are complete at this point. The next step is to create a new object to configure the Modbus TCP server. The object can be found by double-clicking on *Protocols* in the project tree. To open a window containing the list of available objects, click the *Add protocol* button in the opened tab. In this window, after selecting the *Modbus TCP/IP Server*, the selection should be confirmed by clicking the *Add object* button (See Figure 6). After that, a new item, called *Modbus_TCP_IP_Server* is added to the *Protocols*.



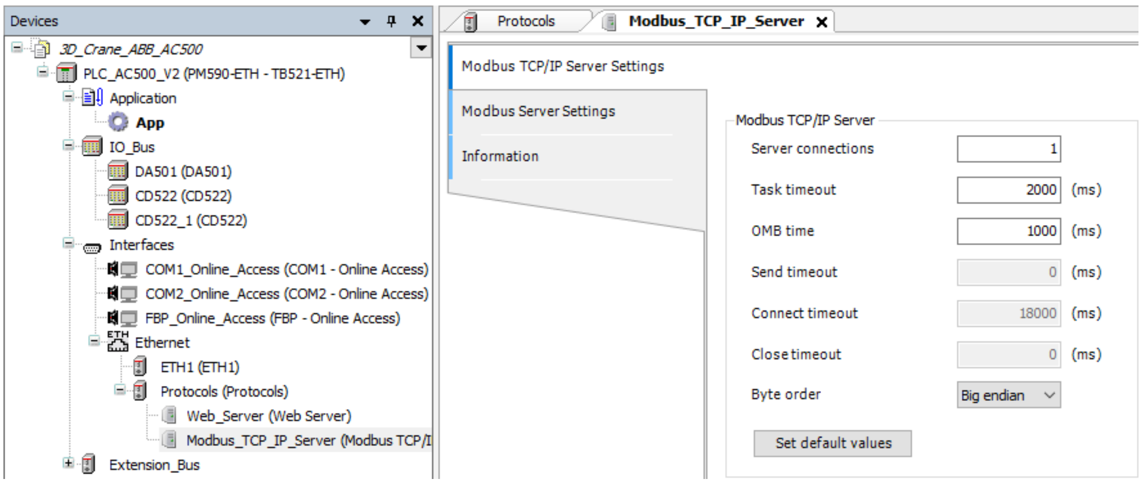
Object path:
PLC_AC500_V2\Interfaces\Ethernet\Protocols

Object name: Modbus_TCP_IP_Server

| Name | Short Description | Version |
|---|---|---|
| FTP Server | | 2.9.0.0 |
| IEC 60870-5-104 Controlstation | | 2.9.0.0 |
| IEC 60870-5-104 Substation | | 2.9.0.0 |
| Modbus TCP/IP Client | | 2.9.0.0 |
| Modbus TCP/IP Server | | 2.9.0.0 |
| Script | | |
| SMTP Client | | 2.9.0.0 |
| SNTP Client | | 2.9.0.0 |
| SNTP Server | | 2.9.0.0 |
| UDP (no AC31 header) | | 2.9.0.0 |
| UDP data exchange | | 2.9.0.0 |
| Web Server | | 2.9.0.0 |

Figure 6. Adding Modbus server object

The server should now be configured. It can be done by clicking the newly added entry. Set the maximum number of parallel connections allowed in the opened tab by entering the desired number into the *Server connections* field, as shown in Figure 7. Save the project to apply the new value (*File* → *Save Project*).

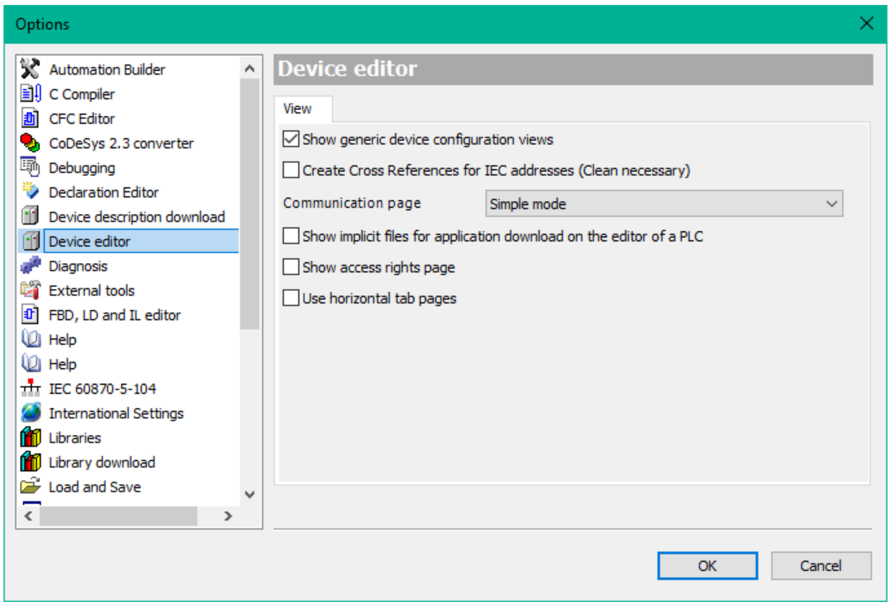


Modbus TCP/IP Server

| | | |
|---|---|---|
| Server connections | 1 | |
| Task timeout | 2000 | (ms) |
| OMB time | 1000 | (ms) |
| Send timeout | 0 | (ms) |
| Connect timeout | 18000 | (ms) |
| Close timeout | 0 | (ms) |
| Byte order | Big endian | |

Figure 7. Configuring Modbus server

As can be seen, not all of the parameters are configurable. If these settings are needed, they can be accessed through the generic device configuration view. This view can be enabled in the *Options* window (*Tools* → *Options*). Make sure the corresponding checkbox is selected in the *Device editor* (See Figure 8).



Device editor

View

☑ Show generic device configuration views

☐ Create Cross References for IEC addresses (Clean necessary)

Communication page: Simple mode

☐ Show implicit files for application download on the editor of a PLC

☐ Show access rights page

☐ Use horizontal tab pages

Figure 8. Activating generic device configuration view

After the Modbus TCP server has been configured, variables must be created and made available via Modbus. Modbus uses register addresses to identify variables in its list. Hence, the Modbus variables should be stored in the special memory area.

Since default Modbus TCP server settings were previously applied, the addressable flag area (%M area) is now accessible via Modbus. By default, all registers have read/write permissions. The read/write permissions for the register ranges can be set in the *Modbus Server Settings* tab (see Figure 7). The addressable flag area can also be changed in this tab.

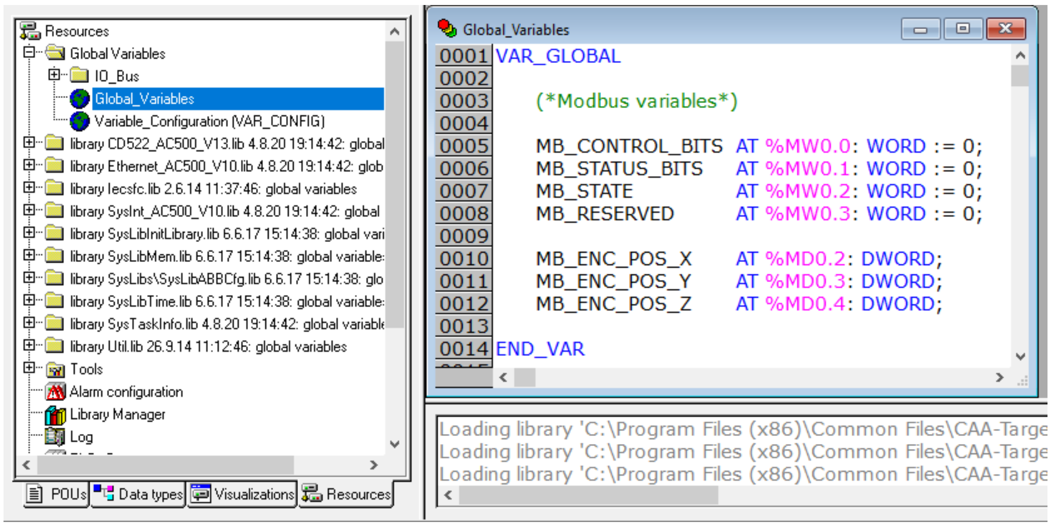
The local %M area is 128 kilobytes (KB), and the memory is divided into two segments (with the numbers 0 and 1), 64 KB each. The Modbus addresses start from 0. Each address points to a two-byte value in memory. Thus, Modbus address range for segment 0 is from *0* to *32767* (*0x0000 – 0x7FFF*), and for segment 1, the range is from *32768* to *65535* (*0x8000 – 0xFFFF*). The address assignment for variables is done in accordance with the IEC 61131 standard. It has the following format:

“ %M Data type ” . ” Segment ” . ” Number ”, where

- Data type – the register data type (B – byte, X – bit, W – word, D – double word)
- Segment – the segment number in the controller’s memory
- Number – the register number in decimal

It is recommended that Modbus variables be defined as global variables. It can be done in the Codesys environment. The *Global_Variables* entry can be found under the project tree, on the *Resources* tab. The global variables are defined here, as shown in Figure 9. The Modbus variables are defined as follows. The first part of the expression is the variable name that is accessible in the code. The second part is the assignment of the Modbus address, which is formatted as described above. The following is the variable type and the initial value (used at PLC start-up).

It is strongly recommended to use the variable type that corresponds to the register data type (for a variable of type Byte - use *%MBx.x*, for type Bool - *%MXx.x.x*). If a variable of type REAL should be declared, the four-byte DWORD should be used instead. Since Modbus registers are 16-bit, the variable of type DWORD is stored in two registers. It should be considered by the client when decoding the received data.


Resources
Global Variables
IO_Bus
Global_Variables
Variable_Configuration (VAR_CONFIG)
library CD522_AC500_V13.lib 4.8.20 19:14:42: global
library Ethernet_AC500_V10.lib 4.8.20 19:14:42: glob
library Iecsfc.lib 2.6.14 11:37:46: global variables
library SysInt_AC500_V10.lib 4.8.20 19:14:42: global
library SysLibInitLibrary.lib 6.6.17 15:14:38: global vari
library SysLibMem.lib 6.6.17 15:14:38: global variable
library SysLibs\SysLibABBCfg.lib 6.6.17 15:14:38: glo
library SysLibTime.lib 6.6.17 15:14:38: global variable
library SysTaskInfo.lib 4.8.20 19:14:42: global variabl
library Util.lib 26.9.14 11:12:46: global variables
Tools
Alarm configuration
Library Manager
Log
POUs
Data types
Visualizations
Resources
Global_Variables

```
VAR_GLOBAL

    (*Modbus variables*)

    MB_CONTROL_BITS  AT %MW0.0: WORD := 0;
    MB_STATUS_BITS   AT %MW0.1: WORD := 0;
    MB_STATE         AT %MW0.2: WORD := 0;
    MB_RESERVED      AT %MW0.3: WORD := 0;

    MB_ENC_POS_X     AT %MD0.2: DWORD;
    MB_ENC_POS_Y     AT %MD0.3: DWORD;
    MB_ENC_POS_Z     AT %MD0.4: DWORD;

END_VAR
```

```
Loading library 'C:\Program Files (x86)\Common Files\CAA-Targe
Loading library 'C:\Program Files (x86)\Common Files\CAA-Targe
Loading library 'C:\Program Files (x86)\Common Files\CAA-Targe
```

Figure 9. Creating Modbus variables

It is important to note that since Modbus addresses are set manually, the development environment allows the intersection of memory addresses. This means that several variables might share a single register, which might corrupt data. The development environment does not identify this scenario as an error. Hence, it must be considered when declaring variables. Another requirement to consider is that PLC inputs and outputs cannot be directly assigned to Modbus registers.

In ABB controllers, the WORD data type Modbus register corresponds to the Holding register. In “Modbus lingo”, the Holding register is a 16-bits register with read and write permissions. Thereby, the WORD data type registers can be read using the function code *03* (*0x03*). Both, Write Single/Multiple Holding register commands are supported (the corresponding function codes – *06* (*0x06*) and *16* (*0x10*), respectively). If necessary, single bits (or “Coils”) can be accessed using the corresponding standard function codes (can be found in [46]). However, it was decided to pack the required single bits into the variables of type WORD and then unpack them on the PC. It increases the data transfer rate and gives the client more general access to the Modbus registers, which in turn simplifies the overall system architecture. As mentioned, the 32-bit DWORD data type takes 2 Modbus registers. The most significant bits are kept in the *n* register and the remaining bits are in the following (*n+1*) register. It should be taken into account on the client side when accessing particular Modbus registers. Recommendations and the overall concept are based on [51].

After the Modbus variables have been defined, they can be accessed in the program code. The content of these variables is available over the network while the PLC is running. No additional actions are required in the Modbus server operating mode since the ABB AC500 handles all network traffic automatically (i.e., sends and receives Modbus telegrams) [47].

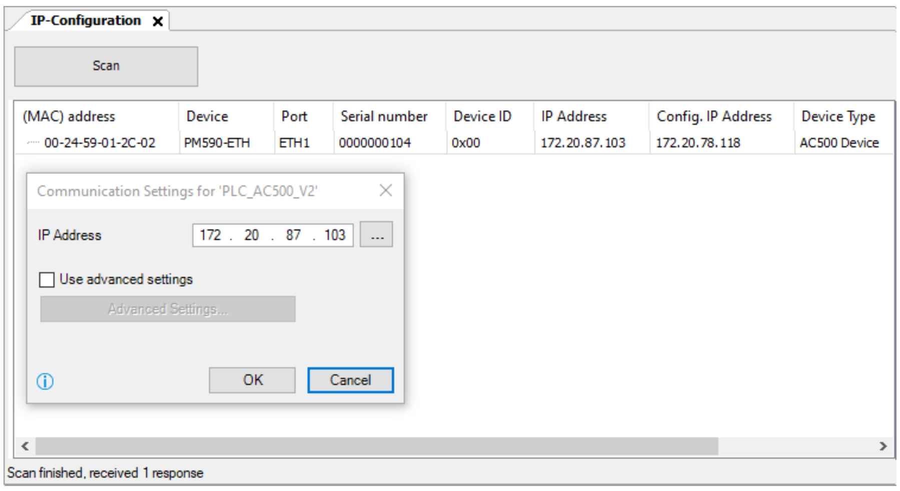
### 3.3.5 PC configuration as Modbus TCP client

Modbus was developed primarily to transfer data between PLCs and field devices. When a PC (or server, cloud, RaspberryPi, or another device) is programmed to send and receive Modbus telegrams, it essentially emulates the second PLC. As mentioned in Section 3.3.3, one of the issues with the Modbus protocol is the lack of authentication [49].

As stated in Section 3.2.2, Python is used to implement the "outside loop", which is meant to run on the PC. There are several Python implementations of the Modbus protocol. PyModbus is by far the most popular implementation (considering the number of stars and downloads on GitHub [52]). It is an actively maintained library distributed under the BSD License. It allows for the implementation of various Modbus protocol versions, including Modbus TCP. The library supports "emulating" both the client and the server [52]. PyModbus is regarded as the implementation with the best performance / Central Processing Unit (CPU) load ratio [53]. Considering the benefits listed above, it was decided to stick with this library.

Other implementations, however, were also being considered. For example, pyModbusTCP, which only supports the Modbus TCP protocol [54]. It has benefits such as a simple Application Programming Interface (API) and a good debug core. Another library under consideration was modbus-tk [55]. It is very similar to PyModbus and provides similar features for implementing the Modbus object. In terms of performance, modbus-tk is approximately two times faster than PyModbus, but it loads a CPU much harder [53]. Since the amount of data traveling between the PC and the PLC in the proposed architecture is relatively low, there was no need to try to achieve better performance. Hence, it was decided to reject the modbus-tk library.

The use of an existing Python library significantly reduces the time required for project development because all Modbus communication functions are already implemented. The following is an overview of configuring and using the PyModbus library as a client. After installing the library, the synchronous Modbus TCP client implementation should be imported (line 1 in Figure 11). Following that, a client instance that connects to the PLC using the Modbus TCP protocol should be created (line 2 in Figure 11). As can be seen, the only parameter that should be set is the Internet Protocol (IP) address of the PLC. All the remaining parameters stay with their default values. The IP address of the PLC can be checked in ABB Automation Builder by clicking *Tools* → *IP-Configuration*. In the opened tab the column *IP Address* shows the current IP address of the PLC. As can be seen in Figure 10, the IP address is 172.20.87.103.



| (MAC) address | Device | Port | Serial number | Device ID | IP Address | Config. IP Address | Device Type |
|---|---|---|---|---|---|---|---|
| 00-24-59-01-2C-02 | PM590-ETH | ETH1 | 0000000104 | 0x00 | 172.20.87.103 | 172.20.78.118 | AC500 Device |

Communication Settings for 'PLC_AC500_V2'

IP Address 172 . 20 . 87 . 103

Use advanced settings

Scan finished, received 1 response

Figure 10. IP configuration tab

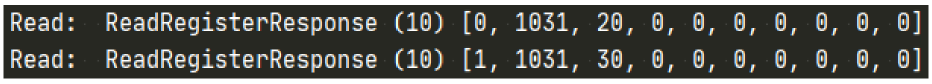
After defining the host to connect to, the corresponding command can be used to connect to the PLC (line 3 in Figure 11). Once the PC and PLC are connected, the Modbus registers of the PLC that were previously configured (discussed in Section 3.3.3) can be read and set. It was noted that it was decided not to use Coils, but rather to pack single bits into variables of type WORD (or Holding registers in "Modbus lingo"). Thereby, using the corresponding functions, all Modbus registers can be set and read (lines 4 and 5 in Figure 11, accordingly).

```
from pymodbus.client.sync import ModbusTcpClient
client = ModbusTcpClient(host="172.20.87.103")
client.connect()
write_reg = client.write_registers(address=0, values=1)
read_reg = client.read_holding_registers(address=0, count=10)
print("Read: ", read_reg, read_reg.registers)
```

Figure 11. PyModbus basic functions

As can be seen in Figure 9, after four variables of type WORD, there are three variables of type DWORD (that take two Holding registers, each). Therefore, there are ten Holding registers in use. Control bits are at address 0, status bits are at address 1, and so on. The last register keeps the lower bits that represent the load lift-line length. Hence, in order to request the contents of all of the used registers, the parameters should be set as shown in Figure 11, line 5. Line 4 in Figure 11, assigns the value 1 (which sets the “Start” flag) to the register with the address 0 (that keeps the Control bits). A more detailed description of the flags and commands can be found in Section 5.1.

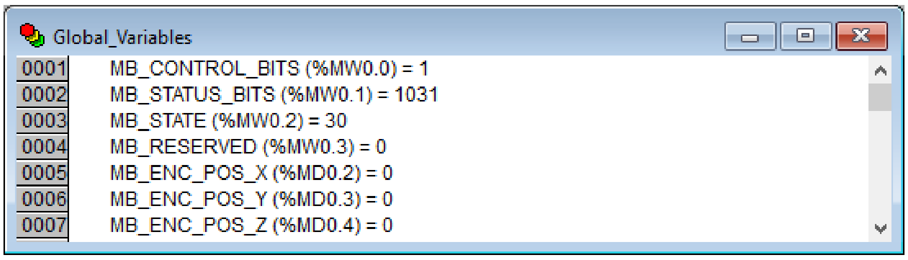
If we run the script, shown in Figure 11 (after reading the registers with the same commands), then the output on the client side (PC) will be the following (see Figure 12).



```
Read:  ReadRegisterResponse (10) [0, 1031, 20, 0, 0, 0, 0, 0, 0, 0]
Read:  ReadRegisterResponse (10) [1, 1031, 30, 0, 0, 0, 0, 0, 0, 0]
```

Figure 12. Modbus registers, PC side

If we look at the PLC values in the Modbus registers, we can see that the value of MB_CONTROL_BITS is equal to one in decimal, which the client just set (see Figure 13). It can also be seen that PLC changed its state to 30 (or the "Main" state). A more detailed description of the states of the PLC can be found in Section 5.2.2.



```
Global_Variables
0001  MB_CONTROL_BITS (%MW0.0) = 1
0002  MB_STATUS_BITS (%MW0.1) = 1031
0003  MB_STATE (%MW0.2) = 30
0004  MB_RESERVED (%MW0.3) = 0
0005  MB_ENC_POS_X (%MD0.2) = 0
0006  MB_ENC_POS_Y (%MD0.3) = 0
0007  MB_ENC_POS_Z (%MD0.4) = 0
```

Figure 13. Modbus registers, PLC side

The maximum length of data that can be read out with a single command is 123 WORDs (or 61 DWORDs). The same restriction applies to the write registers command, i.e., 123 WORDs can be set at once. These restrictions are imposed by the PLC [47].

Similarly, any computer that can run Python code can be configured to act as a Modbus client.

# 4 Computer Vision

As previously stated, vision systems are regarded as the most efficient method for crane workspace mapping. It was also noted that, despite the fact that DL has dominated the domain, classic CV has its own advantages. Since the proposed system (see Figure 2) is intended to be applied indoors, where the environment is more predictable, it was decided to try both conventional and DL-based CV approaches. The first part of this chapter is devoted to the traditional CV implementation, while the second part describes the DL-based CV implementation. The advantages of both approaches are thoroughly discussed and the obtained results are presented.

## 4.1 Traditional CV advantages

Today algorithmic approach in CV is mostly used when it is not possible to deploy a large amount of computation power. For example, if the system is driven by a microcontroller or energy efficiency is a critical factor. Even more in some cases, it is much easier to apply simple color thresholding, instead of gathering large amounts of data, annotating it, and finding the right parameters for training the DNN. Based on [56], [57], [58], [59] the following list of classic CV advantages was compiled:

**Generality** – after the features have been extracted, they will work in the same way for any image (in contrast to DNN, which will work only if the required object of interest was in the training data set). This might be especially useful for color segmentation task. For example, if it is required to detect some instances that have the same color (especially if the lighting is also stable), conventional CV is far more effective. Moreover, it is much easier for an engineer to find suitable parameters for the underlying algorithms, instead of gathering data, labelling it, and so on. Furthermore, if some global features are changed or something goes wrong, the engineer can easily tweak certain parameters so that algorithms work for a broader range of problems.

**Transparency** – each image conversion is observed step by step, hence controlled and configured. This makes the system more flexible and allows the engineer to have better insights into a problem. DNN in this sense, in contrast, is often a black box, especially for an inexperienced engineer. After the training data is prepared, it is simply loaded into the trainer, “magic happens”, and the solution is ready. It is very hard to understand, what happens under the hood since it is related to the complex mathematical operations with sophisticated interconnections between them. Hence in the case of classic CV, the learning curve is smoother. In addition to that, since state-of-the-art models are implemented within various frameworks, the inference possibilities are limited.

**Power efficiency** – traditional CV uses far less computation resources for each frame processing when compared to DNN. Undoubtedly, modern-day Graphics Processing Units (GPUs) are very efficient and could perform millions of math operations in parallel, but they require an appropriate cooling system due to the heat produced during this process. The more sophisticated the cooling system, the more energy it consumes (or the more service it requires). Since all the world is striving for green energy and reasonable energy consumption, the use of DNN in some cases might be an overkill in terms of eco-friendliness. Moreover, in some areas, it might be simply impossible to provide sufficient cooling.

**Reconfigurability** – the implemented system could be adapted to the new environment. This is closely related to the transparency point and means that the image processing pipeline may be changed. Some algorithms might be included, while others excluded. When using the DNN, the whole network should be re-trained (that might take quite a lot of time). Of course, the weights of an existing model could be re-used. In other words, transfer learning technique applied, however, it is not enough just to additionally train the net. This cannot guarantee sustainable performance, which is very important in the industry. Moreover, re-training often requires several iterations as it entails trial and error with various training parameters.

**No data requirement** – despite the fact that camera systems are now ubiquitous and there is no lack of data, the collected data should be sorted and labeled. This is a time-consuming process. Moreover, at the beginning of the complex system implementation, it may not be immediately clear what the objects of interest are (in the sense that it is impossible to consider everything at once). Furthermore, if the idea is to augment the existing system by adding a CV module, there will be no data at all in the beginning.

**Image resolution sensitivity** – conventional CV is less sensitive to the RGB sensor resolution. As shown in Section 4.2.2, various image conversion techniques, such as blur, might be used throughout the pipeline. Some of these techniques reduce the initial image quality. In contrast, the DNN is highly dependent on the RGB sensor resolution.

**Priority** – classic CV algorithms are “programmed”, which means that engineers can specify which features are more important for a specific task. In contrast to DNNs, which are trained. As previously stated, “the magic” that happens under the hood during training is related to complex mathematical operations, and configuring the priorities is a rather complex process.

Some of the points raised above are interconnected and could have been combined. However, it was decided to split the items in this way in order to highlight the most important, in the author's opinion, points. It should also be noted that traditional CV techniques could be used to semi-automate the data labeling process. Since dataset annotation is a major bottleneck in the DL workflow, the solution implemented on conventional CV may be regarded as a significant investment for future system development.

In general, an understanding of classic CV technology is still necessary for the engineer. The reason for this is that it is heavily used to artificially enlarge the available dataset. The process is known as dataset augmentation. It involves the generation of additional training data by cropping, scaling, rotating, and applying various “effects” to existing images. Modern tools and frameworks allow these transformations to be carried out while preserving the labels. Data augmentation is a common method for overcoming limited datasets and reducing overfitting during model training.

## 4.2 Traditional CV implementation

It was decided to use first the classic CV to implement the CV module. This module is intended to collect information about the crane's working area. The key motivator was the fact that there was no training data available at the beginning of the project development. The data labeling process is quite time-consuming and for the sustainable performance of

CNN, the amount of data should be significant. Since classic CV might semi-automate this routine work, the idea of the corresponding implementation seemed more appealing. Moreover, the proposed system is intended to be applied indoors, where the environment (i.e., lighting conditions) can be controlled, and the DL-based approach felt to be overkill. The traditional CV implementation is primarily based on OpenCV (Open Source Computer Vision) library. This is the most popular library in the field, with over 2500 algorithms for various CV tasks. It is written natively in C++ and well optimized to accelerate the calculations at the hardware level. It provides a solid foundation for using the library in real-time vision applications [60].

### 4.2.1 Proposed approach

Classic CV requires an engineer to understand the problem. It is necessary to extract the relevant features in order to find an appropriate solution. Since the goal is to detect a load and move the crane's cart to the point above, the load coordinates should be extracted. The found coordinates must then be mapped to the coordinate plane of the crane. It means that the pixels within the crane's working area should be matched with the crane's encoder-based positioning system. Since it was not possible to limit the camera’s FOV to only the working area, it was decided to use the crane’s frame as a basis to find the origin that would correspond to both coordinate planes (i.e., to match the “pixel coordinates” with “encoder coordinates”). Figure 14 illustrates the task at hand.

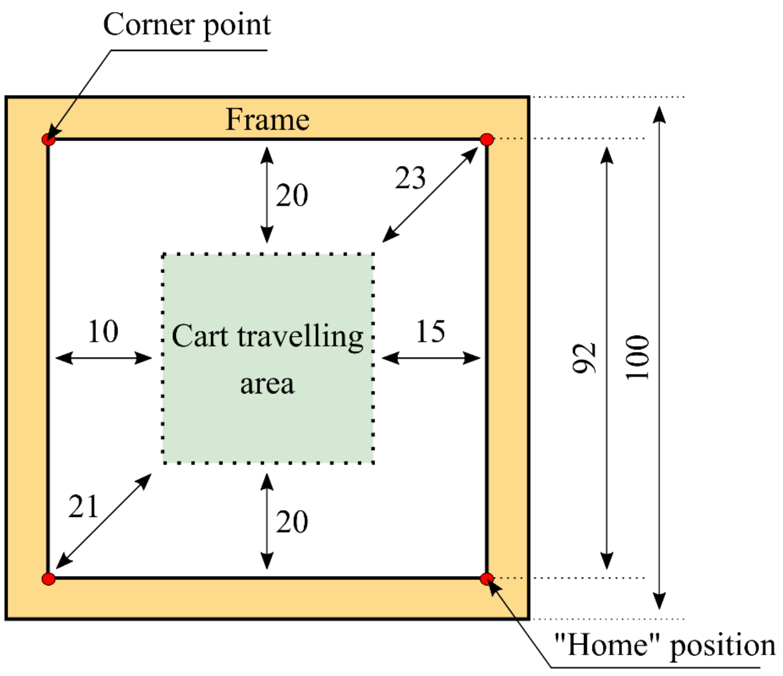


Figure 14. Inteco 3D crane, illustrative cart travelling area (units – centimeters)

The problem is complicated by the fact that the dimensions of the frame are greater than the actual working area. In other words, the cart is travelling only within a limited zone. Moreover, the traveling zone is not centered.

### 4.2.2 Description of the corner points extraction algorithm

To extract the corner points, the following algorithm was developed.

**The first step is to extract color.** During observation, it was noticed that the key feature that separates the crane’s frame from the background is the yellow color (See Figure 15, 1. Original). Color filtering is a common task in a CV that consists of two steps. First, an image should be converted to the Hue, Saturation, Value (HSV) color space, so that the color of interest can be later filtered out by specifying the boundaries that best describe the target color (i.e., apply "Mask"). Although color space conversion is not a necessary step, the HSV color model allows an engineer to obtain more robust results. The reason for this is that in HSV, the color information (Hue and Saturation) is separated from the brightness, or “luma” (Value). Hence, in theory, thresholding rules could be set regardless of lighting changes. However, in practice, it is just a good improvement [61].

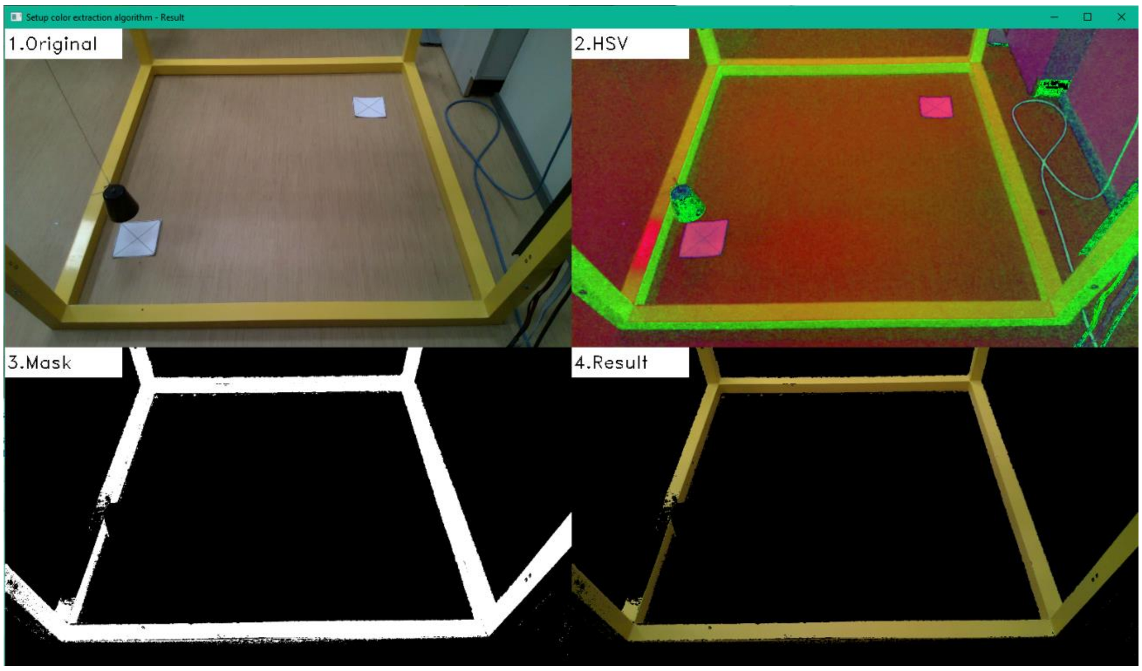


Figure 15. Pipeline description, color extraction

As can be seen in Figure 15, the result is far from ideal. The resulting mask (where white regions represent yellow color and black represents the rest of the colors) contains black spots. One of the spots is caused by the crane’s frame surface that reflects the light. An illuminant is located right above the crane, hence part of the frame has a highly different shade of yellow (i.e., “white spot” in the original frame). The second spot is caused by the load. When the load is at its lowest point, it obscures the view of a portion of the frame. These problems should be considered in the following steps.

**The second step is to find edges.** Since it is required to find the coordinates of internal corner points (see Figure 14), the inner edges of the frame should receive the most attention – while finding parameters to hard code. As can be seen in Figure 16, the proposed method includes four image conversions to find the edges. In the beginning, the resulting mask from the previous step (see Figure 15, 3. Mask) is blurred. This is necessary in order to make the edges smoother. In image processing, the blurring technique is typically used to reduce image noise and reduce detail. Gaussian function was selected for this purpose because it allows preserving more of the edges when compared to other methods (available in OpenCV). Bilateral blurring was rejected since it is considerably slower than Gaussian blur [62].

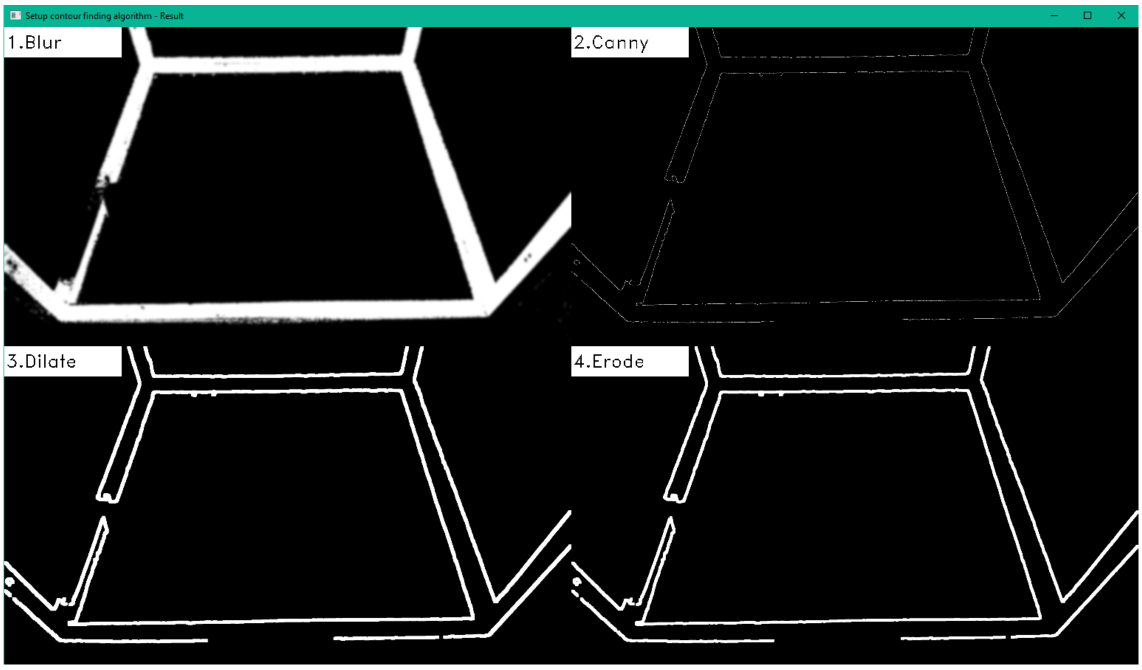


Figure 16. Pipeline description, edges finding

After the frame has been blurred, the algorithm is able to find the edges. For this purpose, a Canny edge detector was applied. This method was selected because of the results it provides, which are more suitable for reaching the goals of the project. When compared to another popular technique widely used in the field – Sobel edge detector, Canny detector makes the resulting edges more precise (i.e., sharp). According to [63], Canny takes the output of Sobel Operator and makes the edges thinner so that they are one pixel wide. Moreover, the Canny detector is considered to be more flexible, as it allows setting two threshold rules. Hence, in theory, an engineer can get rid of the edges, he is not interested in and keep only those, which are of interest.

Analyzing the resulting image, it was noticed that the found edges (See Figure 16, 2. Canny) are highly discontinuous. Moreover, the problems formed during the color extraction made the frame's contour inconsistent. At this stage, it was clear that the best way to find corner points is to artificially draw lines over the extracted contour and then estimate their intersection points. The main reason for such a decision is that the load may obscure one of the top corner points (in the cart's top- rightmost and leftmost positions). As a result, it will be impossible to extract the coordinates of the obscured point since the contour will be inconsistent in that place. Considering the peculiarities of the algorithm, used for drawing the lines, it was decided to apply a combination of dilation and erosion on the "canny frame", to overcome the discontinuity of contour lines.

The basics of dilation are that it increases the area of the features. Hence, by using the right parameters, it is possible to "glue" the discontinuity in contour lines. The only thing to keep in mind is that the inner contour does not intersect with the outer contour. The drawback of using dilation is that by increasing the area of pixels, the inner contour decreases. It reduces the accuracy of estimating the coordinates of corner points. The erode operation is used to return the pixel's area. Erosion works in reverse to dilation. The "magic" of this method is that the contour remains "glued", preserving the inner contour area.

**The third step is to draw lines that correspond to the frame's inner contour.** To find the lines, the Probabilistic Hough line transform method was applied. Hough transform is a popular technique used to find a geometrical representation of any shape, even if the shape is broken or distorted a little bit (i.e., an image has some noise). In its simplest form, the Hough transform is a method for detecting straight lines [64]. OpenCV provides two line detection implementations based on this algorithm. The probabilistic implementation used in this project is an optimization of the classic Hough transform. It has a simpler inference and uses less computation resources. The general idea behind the Hough line transform method is that based on the specified angle deviation accuracy, the pixels that are within – accumulate the score or votes. The number of votes required for the algorithm to conclude that these pixels form a line can be specified. Hence, based on the parameters, the Hough line transform method allows for drawing of the lines that correspond to the specified distribution of pixels. Moreover, the algorithm accepts the presence of gaps in potentially detected lines [65].

The frame, obtained in the previous step (see Figure 16, 4. Erode) is processed using this technique and the result can be seen in Figure 17, 1. Raw Lines. To estimate accuracy, the resulting lines are placed over the original frame (See Figure 17, 2. Combined). Despite the fact that visually result looks satisfying, each of the lines (that form the frame’s contour) is made up of many separate lines, which may cause problems in the next step. Hence, it was decided to blur the frame, shown in Figure 17, 1. Raw Lines. To ensure the quality of this approach, the same image conversion sequence, namely Canny, Dilate, and Erode is performed (see Figure 16). This sharpens the blurred edges, which is required for greater accuracy.

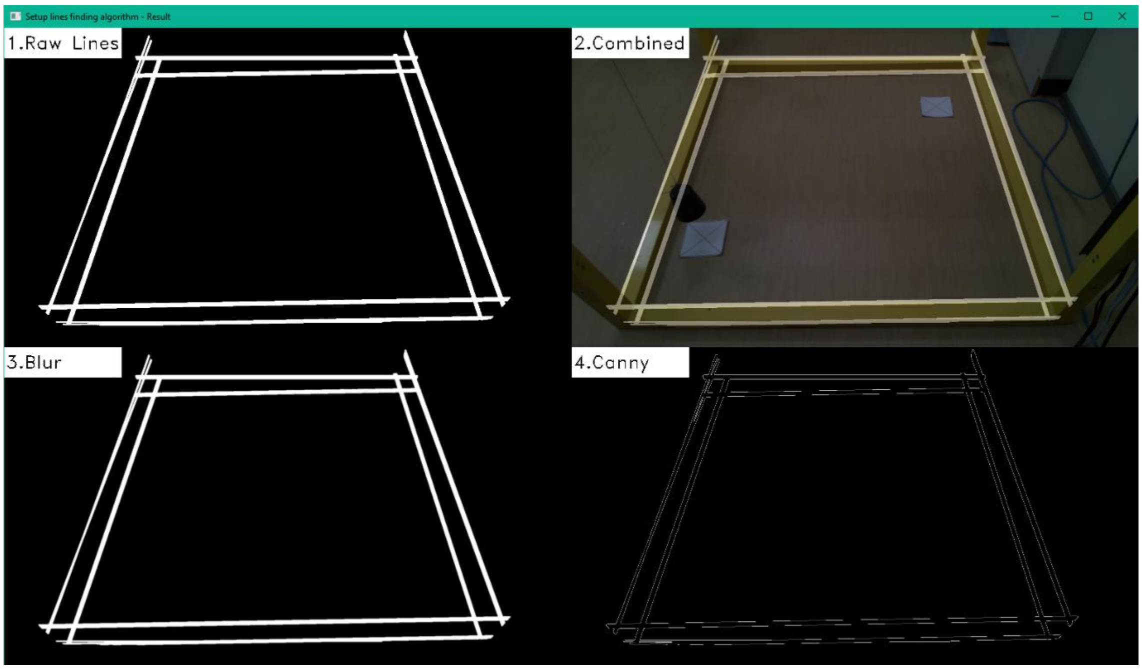


Figure 17. Pipeline description, artificially drawn lines

**The final step is to find the inner contour and its corner points.** Identifying shape contours is a fairly common task in CV. OpenCV includes an implementation for extracting contours from images. The corresponding function is based on the “Suzuki85” algorithm [66]. This is a border following algorithm with a topological analysis capability. In general, the algorithm allows extracting the information about surroundings between two types of borders: the outer- and hole borders (i.e., inner borders, followed by background or hole). In other words, it can estimate the surrounding relations between different contours. Hence, the extracted data can be used to reconstruct the entire original frame, as the "hierarchy" of contours is preserved. The authors state that the proposed method is an effective way of storing binary image data [67]. Using this algorithm, all contours from the last resulting image are retrieved (i.e., the image obtained after applying the Dilate and Erode operations over the frame shown in Figure 17, 4. Canny).

Since the image contains a lot of contours, it was required to find a feature that best distinguishes the sought-for one from others. During the search for such a feature, it was observed that specifying the upper- and lower boundaries, which limit the range of contours by their area, yielded the best results. The reason for this is that the sought-for contour is the only one with an area that differs significantly from the rest of the contours. As a result, there is no need to specify the boundaries precisely. This provides more freedom in selecting the distance between the camera and the crane’s working area. It was also discovered that the proposed approach is quite tolerant to the angle at which the working area is captured.

The only noticed problem is that in some cases, the “Suzuki85” algorithm sees multiple contours in the same place (i.e., overlapped contours). To address this issue, the contour with the greatest area (within the boundaries discussed above) is chosen as the origin. After the required contour is found, its corner points can be extracted. The problem, which may result in an incorrect estimation of the points of interest, is that the pixels that constitute the contour boards are not precisely aligned (i.e., the lines are polygonal). As a result, the algorithm may detect more than four corners or extract wrong coordinates. To overcome this issue, the lines that form contour are slightly approximated. If the contour was successfully detected and none of the problems mentioned in this Section occurred, the function used for approximation [68] returns four points (x, y coordinates) that describe resampled contour.

### 4.2.3 Results and discussion

Based on the results shown in Figures 18 and 19, it can be concluded that the algorithm discussed in Section 4.2.2 performs well in most cases. As can be seen, the algorithm is quite tolerant to the angle at which the working area is captured. Furthermore, the distance between the camera and the crane’s frame may vary. Figure 18 shows that corner points are extracted with reasonable accuracy. Moreover, the algorithm considers environment-specific problems, such as color inconsistency in the crane’s frame surface, and the possibility of obscuring corner points.

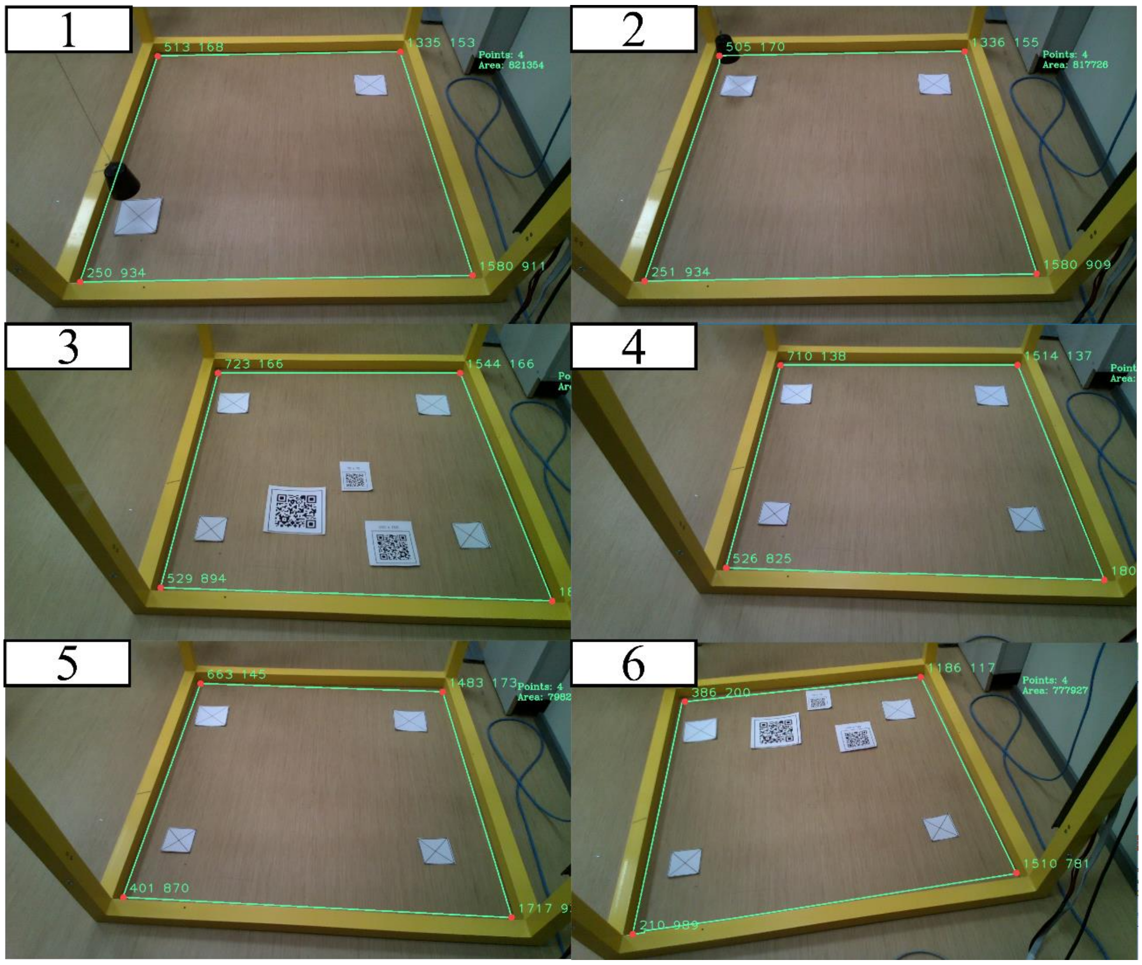


Figure 18. Successful corner points extraction

However, when we compare the results in Figure 18 (second and third image) with the results in Figure 19 (first and second image) – accordingly, we can see that, despite the same camera layout (in both cases), the algorithm may make a mistake when determining the inner contour. Analyzing the root of the problem, the only meaningful reason for such behavior is that the illuminant located directly above the crane was flickering. It caused a change in color, which was detected by the camera's RGB sensor, and as a result, the corresponding system input parameter was unstable. One of the most significant problems with conventional CV, as discussed in Section 4.3, is that it is highly dependent on environmental conditions in general, and lighting in particular. The reason for this is that changing environments disrupts color consistency. The proposed algorithm contains a lot of hard-coded parameters, including the ones used for color extraction. Since illuminant flickering caused too high variations in the RGB values provided by the camera (see Table 2), the algorithm was unable to guarantee stable results in all cases.

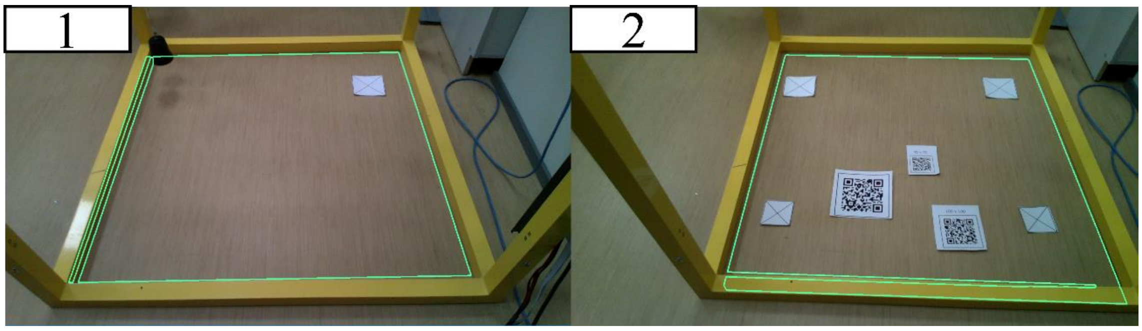


Figure 19. Unsuccessful corner points extraction

The time required for single frame processing is the most significant benefit of the proposed method. On average, the pipeline used for corner points extraction takes 110 milliseconds (PC configuration can be seen in Table 2). Furthermore, since OpenCV uses CPU for calculations by default (this setting has not been changed), there is no need for a high-end GPU. As previously stated, the author's interest lies in finding of a relatively cheap solution, and because GPUs are highly overpriced at the time of this writing, the ability to use only the CPU can be regarded as a significant advantage.

The main disadvantage of this method is that the proposed algorithm contains a lot of hard-coded parameters. Each image conversion, discussed in Section 4.2.2, requires a set of specified parameters. As a result, the corresponding system part configuration is quite complex. Hence, in order to properly adjust the system, the crane operator must have specific skills (in case of a change in the environment). An inappropriate algorithm configuration may not only make the operator's work more difficult, but it may also set off a dangerous scenario. Furthermore, as previously mentioned, certain parameters are overly sensitive to changes in the input signals.

To simplify the configuration process, it was designed to be interactive. The parameters are set using sliders, and the effect of each parameter in the corner extraction algorithm is displayed in real-time. Section 4.2.4 provides a more detailed description of the configuration process.

### 4.2.4 Hard-coded parameters configuration

The crane control application allows the parameters used in the corner extraction algorithm to be configured. The parameters could be adjusted both offline and online, i.e., using previously captured stationary images or frames captured from a live video stream. The configuration procedure is divided into four steps, as shown in Figure 20. These steps correspond to those described in Section 4.2.2.

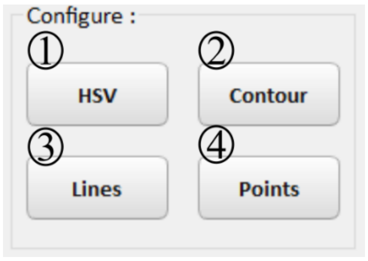
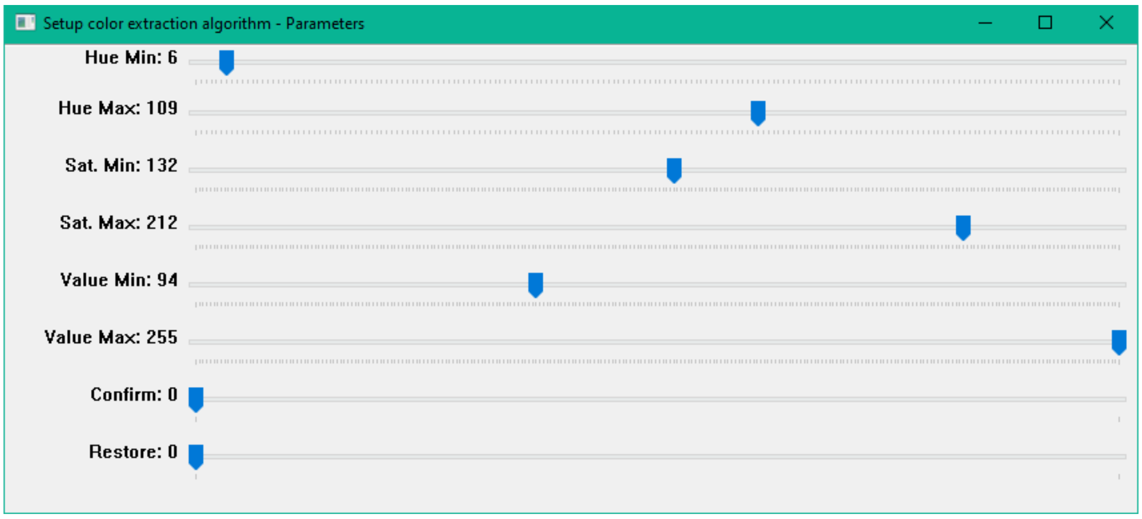


Figure 20. Configuration procedure

A click on one of the configuration buttons brings up two windows. The first one contains sliders for configuring parameters related to the corresponding part of the corner extraction algorithm. Figure 21 shows the HSV values configuration window. The second window contains a series of images that show how the given parameters affect the corresponding image conversion. Figure 15 shows the image set that corresponds to the HSV values configuration window. After configuring the parameters, the user should confirm them by dragging the corresponding slider (see Figure 21). It is also possible to restore the initial values that were used at the start of the setup.


Setup color extraction algorithm - Parameters

Hue Min: 6
Hue Max: 109
Sat. Min: 132
Sat. Max: 212
Value Min: 94
Value Max: 255
Confirm: 0
Restore: 0

Figure 21. Configuration using trackbars

Once the parameters for the image processing pipeline have been configured, the user can save them in the database at the specified entry (see Figure 22). This makes the proposed approach more adaptable and user-friendly because different sets of parameters that work best for various environmental conditions can be easily restored. For instance, if the warehouse is artificially illuminated only at night and there is enough street lighting during the day, two sets of parameters could be set up and easily switched as needed.

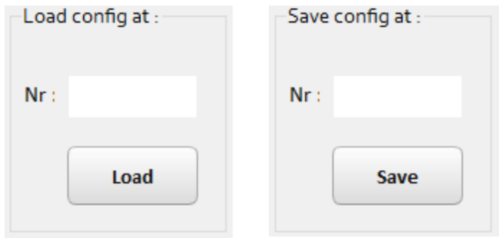


Figure 22. Configured parameters management possibility

The way the configuration procedure is divided might seem illogical, or that some steps should be further divided because they require too many parameters to be set. However, the author's experience has shown that this is the most convenient method for accurately setting these parameters.

### 4.2.5 Load detection

After the corner points have been found, it becomes possible to warp the perspective based on these points. The result can be seen in Figure 23. It is important to keep in mind that the function used for perspective transformation requires the points to be in strict order [69]. Since there is no guarantee of the order of the points found at the previous step (see Section 4.2.2), it is required to reorder the corresponding array. Knowing the imposed order for the points (top-left, top-right, bottom-right, bottom-left) the rule based on the difference in x and y values was applied. The bottom-right corner point is the one with the greatest sum of x and y, while the top-left point has the smallest sum. The bottom-left point has the largest difference in x and y values, whereas the top-right corner point has the smallest. All subsequent image processing is performed on the frame with a transformed perspective (see Figure 23). The reasons for this are that the frame has a lower resolution, which reduces the processing time, and that it is easier to match the extracted load location with the crane’s encoder-based positioning system.

As mentioned in Section 2.2, QR codes have a high potential for widespread use in crane workspace mapping. The reason for this is that barcodes are a low-cost and widely used method of matching an object with machine-readable information. The widespread use of QR codes in our daily life gives a reason to believe that the same trend will be moved to industrial warehousing. Due to modern printing capabilities, attaching or printing the QR code over the goods holding container is quite simple. Furthermore, because of their error correction capabilities, QR codes are robust to dirt and scratches [70]. Considering the merits of the above, it was decided to use QR codes to keep the data associated with the corresponding cargo in the proposed system.

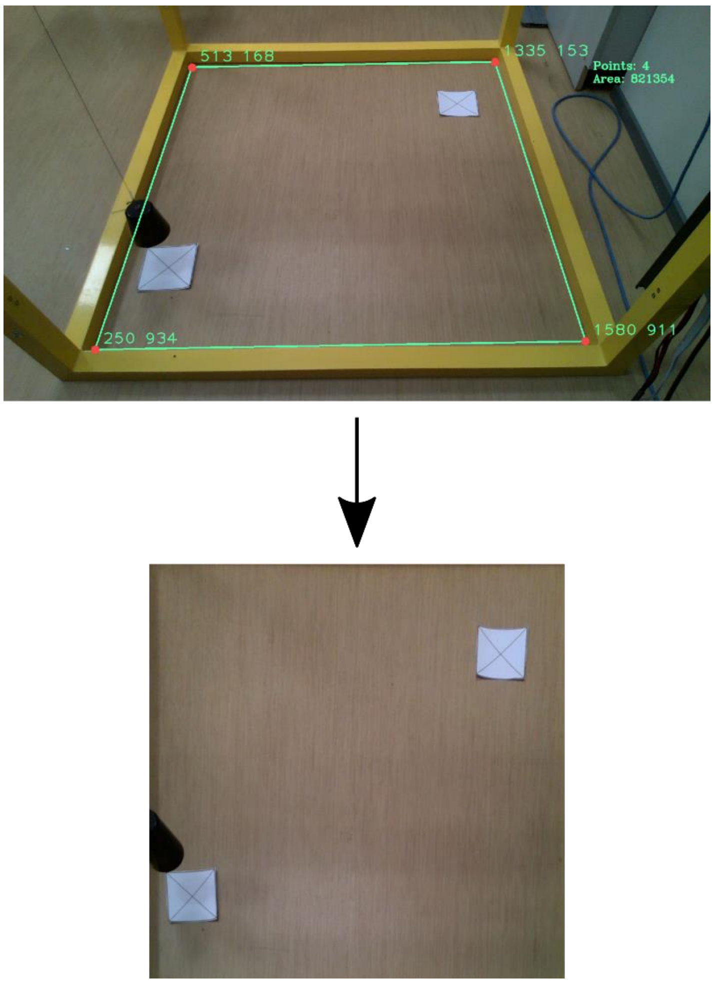


Figure 23. Warped perspective

Assuming that the QR code is centered on the roof of a container, it is necessary to locate the center point of the barcode and send the corresponding coordinates to the crane in order to pick up the load. A dedicated Python library called pyzbar is used to detect and decode a QR code. It has all the necessary Python bindings for accessing the ZBar library. ZBar is an open-source software suite for reading various types of barcodes, including QR codes. All the performance-sensitive image processing is done in C, making it suitable for real-time applications [71].

Despite the fact that ZBar performs some image pre-processing, the performance of the function intended for detecting and decoding QR codes within the frame was quite poor. An adaptive binarization was used to improve the detection rate of barcodes. The implementation from the Kraken library was chosen to binarize the frame as it produces good results [72]. Kraken is an open-source Optical Character Recognition (OCR) system [73]. However, since the image cleaning performed prior to OCR is very similar to the noise filtering required for barcode detection, it also works well with QR codes. The only disadvantage of the referred implementation is the time required to binarize an image. On average the algorithm takes 500 milliseconds to process a frame on CPU (PC configuration can be found in Table 2). Hence, it was decided to scan QR codes once every two seconds – to reduce the CPU load. Because cargos are expected to have low dynamics, the proposed resolving cannot be considered as a significant shortcoming. Figure 24 shows the outcome of an adaptive binarization as well as the found contours of QR codes.

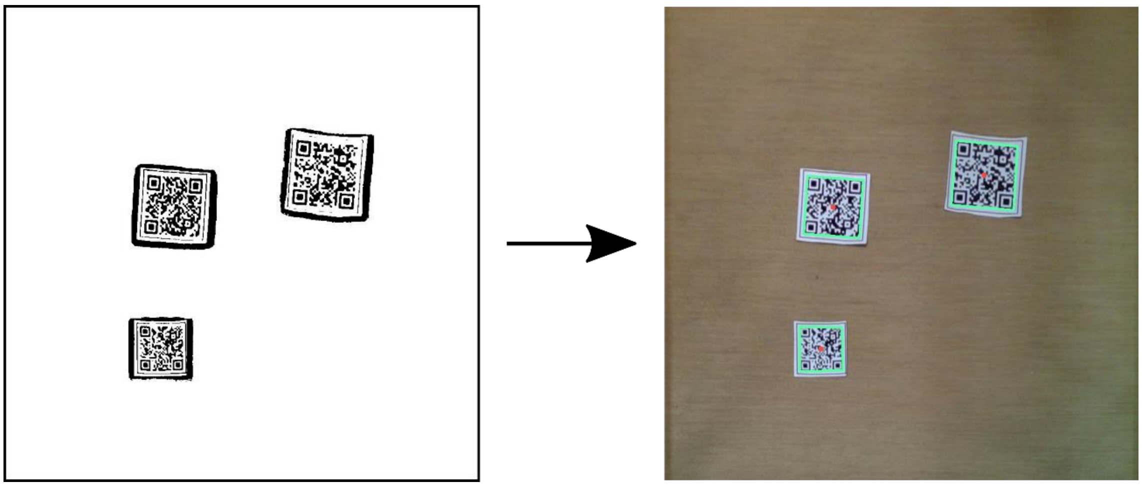


Figure 24. Binarized and QR highlighted frames

Knowing the polygonal curves that form the contour around the QR code allows finding its centroid (i.e., the center of the object). OpenCV includes a function that computes all of the moments of a polygon up to the third order [74]. Knowing the moments, the centroid can be found using the following formula [75]:

$$C_x = \frac{M_{10}}{M_{00}};\ C_y = \frac{M_{01}}{M_{00}} \tag{1}$$

In Equation (1), $C_x$ and $C_y$ are the x, y – coordinates of the centroid and $M$ denotes the moment (that corresponds to the output of the corresponding OpenCV function). The resulting centroids are depicted in Figure 24 as red dots. The found x, y coordinates could then be mapped to the crane's coordinate plane and used to set the desired position.

## 4.3 DL-based CV advantages

As previously stated, more and more researchers try to adopt various DNN architectures for different tasks. Despite the challenges and drawbacks discussed in Section 4.1, the hype and widespread use of DNNs has strong reasons behind it. In tasks such as object detection, image classification, and semantic segmentation, DNN outperforms traditional CV [56]. Moreover, DL has opened many doors, and the scope of its application areas is likely to grow in the future. For instance, it is already possible to colorize, reconstruct, or even increase the resolution of an image. Furthermore, using neural networks, the style of one image could be transferred to another [76]. A lot of debates and concerns are related to the “Deepfake” technology. This technique is also associated with CV and is typically based on Generative Adversarial Networks (GANs) [77]. These are just a few examples of what the DL-based CV is capable of, and because the field is overheated right now, new models, as well as the application possibilities are being invented with a high frequency.

Since the scope of the thesis is limited, the following list highlights the general reasons why CNNs became such a popular CV tool, and why, in the author’s opinion, they might be especially relevant for industrial purposes (the list is based on [56], [78], [58]):

**Flexibility** – the same DNN architecture can be applied to a variety of tasks. DL-based CV models are trained but not programmed. This means that the model can be adapted to any specific problem. The only requirement is the corresponding training dataset. There is no need to look through the constituting layers of a model. For instance, the same architecture can be used for cancer screening or re-trained to search for defects in the road surface.

**Accuracy** – the latest DNN architectures provide substantially higher accuracy, especially in tasks such as object detection, image classification, semantic segmentation. The general rule is to provide the model with as much training data as possible. However, it is important to keep in mind that the data should be diverse and qualitatively labeled. With a sufficient amount of diverse data and appropriate training parameters, the achieved accuracy could be much higher than that of conventional CV. Moreover, the DL-based approach can withstand a larger number of classes to classify, which is the problem in the case of a traditional CV. The reason for this is that with a large number of classes, manual feature extraction becomes tricky.

**Human factor** – the DL-based approach employs an algorithmic model that enables the machine to learn the underlying patterns in a given dataset on its own. DNN understands the pixel-level nuances that make up the parts of a larger image. This eliminates the risk of unmodeled variables that engineers may overlook. People tend to make mistakes due to ignorance or a failure to understand some complex, hidden or non-intuitive phenomenon.

**Data availability** – in general, there is no shortage of data in the world. Camera systems are now ubiquitous, and if necessary, an additional camera can be mounted in the required location. The cost of a camera, as well as the price per data storage capacity, is low. The bottleneck associated with the data labeling process could be resolved if desired (for instance, by involving outsourcing). Moreover, as previously stated, a conventional CV algorithm developed in advance might semi-automate this time-consuming process. Since it is a global concern in the field, researchers are working to find an appropriate solution. Recently, a concept known as Active Learning has emerged. This technique makes use of the deployed system to identify problematic cases. For example, if it was found during system operation (or testing period) that two similar subjects are frequently mixed up under certain conditions, the focus should be shifted exactly to them. The process is iterative. It significantly reduces the time required to improve system performance [79].

**Simplicity** – people tend to make their work easier, especially engineers. Computer systems are very useful in this regard because they allow us to automate our daily routines. The same trend is observed in the field of CV. The DL-based approach allows features to be deduced automatically and optimally tuned for the desired outcome. The algorithm extracts the relevant features without being explicitly told to do so, which is much easier than manual feature extraction.

**High automation** – due to the popularity of DL-based CV and its application simplicity, a large number of libraries, end-to-end frameworks, and platforms have been emerged. They provide modern tools and state-of-the-art models that make it easy for developers to build and deploy DL-powered applications. These platforms are mostly open source and have a large community behind them. This makes the demanding learning curve smoother. The most popular ones are TensorFlow, PyTorch, and Keras [80]. Moreover, there are ecosystems, that allow training a model even without touching a code, for instance, Roboflow [81].

**Tolerability** – CNNs are more tolerant to the lighting conditions, viewpoint of a camera, background noise, visual obstructions, and intra-class variations. Among the many reasons why researchers favor the DL-based approach, the author believes that the factors listed above are the most important ones. As previously stated, the more diverse data DNN consumes, the better performance it will provide. The collected data may include samples taken under various lighting conditions, in different environments, and even with different cameras. Essentially, there is no difference, especially if the data is labeled manually. This is not the case, if the system is based on classic CV since these factors affect the color consistency (on which conventional CV is very dependent). The same rule applies to the viewpoint of a camera and intra-class variations. In the case of classic CV, the extracted features should be generalized or the algorithm should consider that, for instance, the camera's FOV may vary. Moreover, as previously noted, modern tools provide the ability to artificially augment the training dataset (by changing the brightness, contrast, saturation, cropping, scaling, etc.), which provides the foundation for DNN-based models to be more reliable in terms of tolerability.

**Computation power availability** – the cost of computing dropped significantly nowadays. Modern GPUs can run DNN-based algorithms in near real-time. Some companies even started producing special purpose circuits that allow accelerating AI calculations. A well-known example is the Tensor Processing Unit (TPU), developed by Google [82]. Another example is the chip architecture, developed by Ambarella. Their solution – CVflow® is specifically designed for running the CNN algorithms with extremely low power consumption [83]. Moreover, some companies provide their computation resources for training a model at no cost. For instance, Google – with their "Google Colaboratory", or IBM, which recently launched a similar solution, called "IBM Watson Studio" [84].

Considering the points discussed above, the author believes that the future belongs to the DL-based approach in CV. Traditional CV techniques are still relevant today, but progress is moving forward, and the benefits of DNN will outweigh the drawbacks discussed in Section 4.1. The values mentioned under the accuracy and tolerability points motivate researchers, corporations, and governments to investigate this technology further. The fact that CV is one of the most active sectors for investors attests to the growing interest. [85].

The most significant issue, in the author's opinion, is the high power consumption caused by the high computation resource requirement. It is important to keep in mind, however, that AI algorithms are currently mostly processed on GPUs. In relation to AI, a GPU is a general-purpose processor. The problem will almost certainly be solved with the increased development of special-purpose processors and their widespread use. Taking into account all of the above, it was decided to try to implement the CV module using the CNN model as well.

## 4.4 DL-based CV implementation

There are many DL frameworks, and it is logical that each has its own advantages and disadvantages. Undoubtedly, such variability is a big plus on the global scale because it reduces the risk of field monopolization. Furthermore, different approaches to solving the same problem taken by different communities greatly increase the likelihood of new breakthroughs. However, such variability makes the process of selecting the best solution for your specific problem quite complex, especially for beginners in the area. The choice is complicated by the fact that each community supports the framework over which they develop and use. Another reason is that there is no solution that is ideal in every way.

### 4.4.1 Detectron2

After thorough research, Detectron2 was selected as the platform for developing a DL-based CV module. Detectron2 is the Facebook AI Research library that contains a variety of state-of-the-art models, including Mask R-CNN, which is required to achieve the project's objectives. Detectron2 is implemented in PyTorch, which means that developers will benefit from the advantages that PyTorch has over Tensorflow [86]. The most important ones for this project are related to inference and training time, which is less in the case of PyTorch (if GPU is used for computations) [87]. According to official benchmarks, the training speed of Mask R-CNN in Detectron2 is faster than that of other popular open-source model implementations [88]. Detectron2 is designed with the research use-case in mind, which means it is more flexible in system configuration for controlling the experiments (compared to pure PyTorch) [89]. The aforementioned factors were the key motivators for the author to try to implement the DL-based CV module using Detectron2.

The most significant issue is that Detectron2 does not have official support for Windows Operating System (OS) [90]. Nevertheless, there is a way to install the latest build (v0.5) – at the time of this writing. This method has been validated by the author. It is about using specific versions of libraries and drivers. The link to the comprehensive guide can be found here [91]. It is important to note that the sequence of steps, as well as the specified versions of libraries and drivers, should be strictly followed. Otherwise, this method will not work, which was also verified by the author. Detectron2 installation instructions on Windows OS can also be found in Appendix 2.

### 4.4.2 Model training

Google Colaboratory was selected as the platform for training the model. The training process is straightforward and well described in the official guidelines, which can be found in the Detectron2 GitHub repository [92]. The most time-consuming part was the dataset registration because Detectron2 requires the metadata to be in a specific format that is similar to COCO's annotations. The recently emerged official guide describes in detail what keys should be associated with an image for various tasks [93]. For image annotation, VGG Image Annotator (VIA) was used. It is a lightweight, web-based, open-source tool for image, audio, and video annotations. The user interface is quite simple and, in the author's opinion, very convenient. It allows metadata to be exported in .csv, .json, and COCO formats [94].

Initially, the model was trained using the default parameters specified in the official Colab Notebook. Since the crane's working area extraction accuracy was poor, it was decided to tune the hyperparameters, as they have a significant impact on the performance of the model. The reason for this is that hyperparameters directly control the behavior of the training algorithm [95]. As previously stated, Detectron2 is a powerful platform that includes a variety of state-of-the-art models. It was also mentioned that the platform is designed with the research use-case in mind, which means that it provides more flexibility in the model configuration. However, when it came to configuring the hyperparameters, the author discovered that the things he expected to be provided out-of-the-box just were not. This meant that in order to properly configure the model it was required to deeply dive through the code. Analyzing the library, it was found that all of the configurable hyperparameters are kept in the following configuration file [96]. Since Detectron2 includes a variety of models, the list of parameters is quite vast.

Unfortunately, due to time constraints, it was not possible to understand the impact of each parameter on the resulting accuracy that the trained model could provide. Moreover, reviewing the relevant resources, it was noticed that most researchers believe that trial and error is the best way to find appropriate hyperparameters. The majority also associates parameters tuning with adjusting a black-box function. Hence, it was decided to begin experimenting with the parameters included in the official tutorial, as it stands to reason that these parameters should have the greatest impact on the performance that the trained model will provide. The parameters mentioned are learning rate, number of iterations, batch size per image, and images per batch. The last two parameters represent the total number of Regions of Interest (RoI) per training minibatch (which is calculated by multiplying these parameters). The learning rate and the number of iterations had the greatest impact on the resulting accuracy.

Table 3. The hyperparameters used to train the model

| **Hyperparameter** | **Value** |
|---|---|
| Learning rate | 0.0001 |
| Number of iterations | 1200 |
| Images per batch | 4 |
| Batch size per image | 512 |

Since the amount of labeled data was limited, there was little room for experimentation. The resulting accuracy, shown in Figure 25, was obtained using the hyperparameters, listed in Table 3.

### 4.4.3 Results and discussion

Analyzing the results obtained using the models trained with various parameters, it was discovered that the working area borders (i.e., the borders drafted by the corresponding crane's frame inner contour) are wavy in all cases (see Figure 25). Furthermore, if the angle at which the working area is captured deviates from the ideal camera's FOV, the "waviness" increases. The ideal FOV is assumed to be when the lower bound of an image is parallel to the nearest crane frame side (see Figure 3).

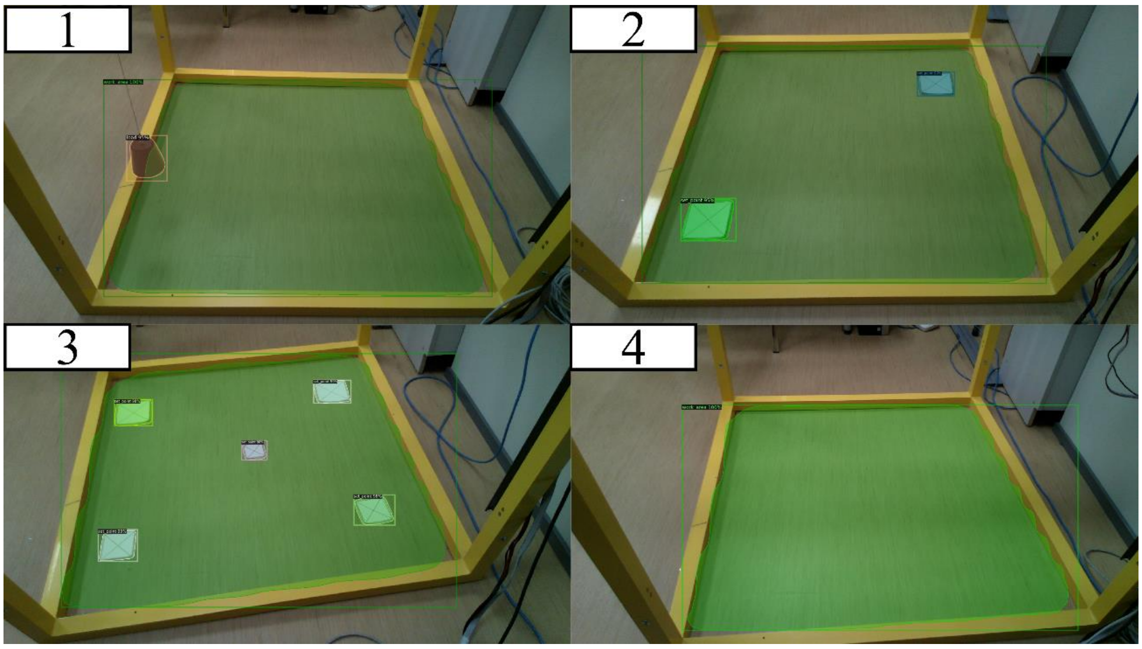


Figure 25. Accuracy obtained using Mask R-CNN

Investigating the root of the problem, it was discovered that the waves discussed above are a side effect of how the Mask R-CNN works. According to several authors, even with large data volumes and an increase in the number of iterations, the problem remains relevant [97], [98]. A promising Mask R-CNN extension was proposed to address this issue [99]. Unfortunately, to the best of the author's knowledge, there are no PointRend implementations available on the internet. The official project's GitHub page only shows the usage and visualizations of the algorithm point sampling stages [100].

Another drawback of the DL-based approach is the time required for single frame processing. On average, it takes 325 milliseconds to process a frame on GPU (PC configuration can be found in Table 2). However, it is important to note that the working area, as well as the objects within it, are detected in a single step.

In terms of object detection, the DL-based approach is expected to perform better. Furthermore, this method should provide better results in object recognition. The reason for this is that, as previously discussed, DNN can handle a greater number of classes to classify than conventional CV. Besides this, rather than searching for type-specific features, it is much easier to train the model to detect and recognize different types of loads.

It is reasonable to expect that the resulting accuracy will be preserved even when the lighting changes. This means that even the high variations in RGB values caused by illuminant flickering, which was a problem with traditional CV implementation, should not pose problems. The only requirement is to collect a sufficient number of frames while the lamp is temporarily turned off. The same requirement applies to the loss of accuracy when the angle at which the working area is captured varies. It is expected that the model will perform better with a more uniform dataset in terms of camera FOV variability.

Because of inaccuracy in estimating the working area borders, it was decided to stick with the classic CV implementation, discussed in Section 4.2. Furthermore, the time required for single frame processing is nearly three times longer, and it is not possible to run a separate part of the algorithm more rarely to increase the overall speed of execution (as was the case with traditional CV implementation). Since the proposed system is intended to be used indoors, where the environment is more predictable, the classic CV implementation seemed to be more sustainable.

# 5 Crane control app

This chapter discusses the developed application for controlling the Inteco 3D Crane. As mentioned previously, the overall control loop consists of inner and outer loops. The inner loop is meant to be running on PLC. In this loop, all the safety- and time-critical functionality is implemented. The outer loop is meant to be running on a PC. In this loop, all the rest functionality is realized, i.e., manipulations with the database, GUI, image processing, and so on. The first part of this chapter describes the overall communication logic between the PC and PLC. The second and third parts are devoted to the PLC and PC implementations, respectively. Finally, at the end of this chapter, the GUI is explained.

## 5.1 Communication logic between PLC and PC

The proposed system involves the constant interaction of two computing devices (see Figure 4), which requires a thorough consideration of their communication. Furthermore, because the inner loop is significantly faster than the outer loop, some functionality, such as actions reliant on the status of position limit switches, should be performed entirely within the inner loop, without the need to wait for a response from the outer loop. Besides that, since the communication possibilities are limited, it is necessary to think through a system of checks and flags so that it could be interpreted and understood on both devices while leaving room for future development.

The first two Modbus registers (each 16-bit long), named control bits and status bits – store the most important data for communication logic between the PC and PLC. As previously stated, it was decided to pack the required single bits into the variables of type WORD and then unpack them on the PC – in order to increase the data transfer rate and provide more general access to the Modbus registers. The control and status bits are implemented in this fashion. Each bit stored in these registers has its own meaning, which is listed in Tables 4 and 5. The only bits in use are described. The absence of a specific bit in the corresponding Table indicates that it is not currently in use and has been reserved for future development.

Table 4 describes the contents of the control bits Modbus register. The bits in this register are used to set a command for the PLC to execute. When a user sets a command into GUI on the PC side, the PLC changes its internal state and performs the corresponding duties. When the command is finished, the PLC resets the corresponding control bit, indicating that the command was successfully completed.

Table 4. Description of control bits

| **Control bit** | **Description** |
|---|---|
| 0 | The bit is used to set the “Start” flag. When the bit is set, the PLC begins executing commands. As a consequence, the bit should be turned on continuously during the crane operation. |
| 1 | The bit is used to set the “Go home” command. The command is intended to move the load to the location specified by position limit switches (in the x, y, and z – directions). After reaching the point, the position measuring encoders are reset. |
| 2 | The bit is used to set the “Go center” command. The command is intended to move the cart to the center of the crane’s working area (in x, y – directions). |
| 3 | The bit is used to set the “Go to” command. The command is intended to move the cart to the desired position. The set point values (in x, y, and z – directions) are retrieved from the corresponding Modbus registers. |
| 4 | The bit is used to set the “Go back” flag. After performing the "Go to" command, the flag indicates the need to return to the initial position. |
| 5 | The bit is used to initiate the manual “Reset angle encoders” procedure. After setting the bit, the PLC waits for the user to confirm that the load is stable (i.e., perpendicular to the floor and not hanging). |
| 6 | The bit is used to set the “Pick QR” command. In this semi-automatic mode, the crane picks up the selected load (noted by a QR code) and places it in the desired position. |
| 7 | The bit is used to set the manual “Lower load” command. When the bit is set, the crane starts lowering the load until the bit is released or the maximum lift-line length is reached. |
| 8 | The bit is used to set the manual “Lift load” command. When the bit is set, the crane starts lifting the load until the bit is released or the corresponding position limit switch is activated. |
| 9 | The bit is used to set the “Move in z-direction” flag. The bit requires that the load be set to the desired height after performing the "Go to" command. |
| 15 | The bit is used to set the “Emergency” command. Motors are immediately turned off once the bit has been set. Movement in any axis direction is prohibited until the bit is reset. |

Table 5 contains the description of status bits Modbus register. The bits in this register are used to indicate the current crane status and to provide additional information to the PC side about what the crane is doing right now within a command. These bits are set (and mostly reset) by the PLC. The exception is bit ten, which is set by the user to confirm that the load is not hanging. The values of status bits are read-out and interpreted by the PC to provide all relevant information to the operator.

Table 5. Description of status bits

| **Status bit** | **Description** |
|---|---|
| 0 | The bit indicates that the crane hardware has been successfully initialized. |
| 1 | The bit indicates the need to reset position encoders. |
| 2 | The bit indicates the need to reset angle encoders. |
| 3 | The bit indicates the cart's movement status. The bit rises when the cart moves (in x, y – directions). |
| 4 | The bit indicates that the load has been picked. |
| 5 | The bit indicates that the command was successfully completed. When a new command arrives, the PLC automatically resets the bit, indicating that the crane has become busy. |
| 6 | The bit indicates that the load lift-line length is set (if the command required to set the load in the z-direction). |
| 9 | The bit indicates that the cart is returning to its initial position (if the command required to return back after reaching set point). |
| 10 | The bit is used as a user acknowledgement that the load is stable (i.e., perpendicular to the floor and not hanging). |
| 11 | The bit indicates that the values of the angle encoders are not changing (i.e., the load is not hanging). |
| 15 | The bit indicates an error. If this bit is set, the system must be reset. |

The following is a general description of how communication between the PC and PLC occurs. When an operator sets a command into the GUI (on the PC side), all of the associated parameters required to execute the command are sent to the PLC. The PLC validates the parameters and executes the command based on its internal state. When the command is set, it is fully processed on the PLC, ensuring safety while performing time-critical functions. The parameters check, which occurs prior to command execution, ensures that the command would be executed correctly. Since the command is processed entirely within the inner loop, any communication problems with the outer loop will not pose a risk (i.e., equipment damage or danger to surroundings). During command execution, the PLC sends continuous feedback, which is decoded on the PC and interpreted in a user-friendly manner on the GUI for the operator.

## 5.2 PLC side development

The programming of ABB controllers takes place in Codesys environment. Codesys itself is hardware-independent and free to download PLC programming environment that is integrated into Automation Builder. Since Codesys is not dependent on any particular device, all the necessary hardware configurations are gathered from the Automation Builder and sent to Codesys as a target file for finalization in the program.

### 5.2.1 PLC responsibilities

As discussed in Section 3.2.1, PLCs are designed to operate in real-time. This means that PLC can provide fast, reliable, and deterministic results. Hence, all safety- and time-critical tasks should be performed on a PLC. In the context of the current setup, the PLC is responsible for the following tasks:

- Maintain a list of Modbus registers
- Respond to the client’s commands
- Send the encoder values to the client
- Provide client with the information about self-internal state
- Check the associated parameters prior to command execution
- Reduce load oscillations by using PID controllers
- Check the status of the position limit switches
- Generate an appropriate PWM signal for controlling the crane motors

### 5.2.2 PLC side implementation

The PLC is programmed as a state machine. Table 6 provides a brief description of what happens inside each state and what the state is used for. A transition from one state to another is triggered by events represented as commands (i.e., the contents of command bits Modbus register). Certain status bits might also require additional state transitions. The interconnection between the states can be seen in Figure 26.

Table 6. The purpose and a brief description of the states

| **State** | **Description** |
|---|---|
| 0 | The initial state in which the hardware is initialized. The validity and plausibility of the inputs connected to the encoders and DC motors are checked in this state. Angle encoders are automatically reset if the load was not hanging during initialization. |
| 10 | The state intended to perform the “Go home” command. After initialization, the system enters this state to reset position encoders. Later, the corresponding command can be used to initiate the transition to this state. |
| 20 | The idle state. In this state, the system is waiting for the "Start" flag to be set. If angle encoders were not reset automatically during initialization, the system asks the operator to reset them manually. Resetting the "Start" flag initiates the transition to this state. |
| 30 | The main state, intended to connect the other states. When a command is executed successfully, the PLC returns to this state, allowing the new command to be set. |
| 40 | The state intended to perform the “Emergency” command. Movement in any direction is prohibited in this state, and motors are kept turned off. This state can be entered at any time by requesting the corresponding command. |
| 50 | The state intended to perform the “Go to” command. In this state, the PLC sets the desired cart position. Additional actions, such as load picking, might also be performed based on the flags set prior to command execution. All required parameters are validated before the command is executed. |
| 60 | The state intended to perform manual lifting or lowering of the load. In this state, the PLC continuously lifts or lowers the load (based on the requested command). After the corresponding control bit is set (or reset), the transition to (and from) this state occurs automatically. |
| 61 | The state intended to lift the load to its highest position. This operation is always performed prior to the execution of the "Go to" command to reduce load oscillations and avoid the risk of load collision (due to height). |

| **State** | **Description** |
|---|---|
| 65 | The state intended to set the load position in the z-direction. The PLC sets this state to adjust the load lift-line length if the corresponding flag was raised prior to the "Go to" command execution. When the desired height is reached, the PLC returns to state 50 to complete the command. |
| 70 | The state intended to perform the manual “Reset angle encoders” procedure. Transfer to this state occurs automatically if angle encoders were not reset during initialization and can be initiated by setting the corresponding command. In this state, the PLC reads the angle encoder values and waits for the user to confirm that the load is perpendicular to the floor and not hanging. |
| 75 | The state intended to automatically reset angle encoders (without the user acknowledgement). Due to the imperfect work of angle encoders at the hardware level, sometimes their values are stuck near zero. If the system detects a lack of load oscillations while the angle encoder values are close to zero, it enters this state. Since no acknowledgement from the operator is required, the system returns to its previous state automatically. |
| 900 | The state intended to indicate a hardware-level problem with position or angle encoders, or motors. |

As shown in Figure 26, the program begins with hardware initialization. At this step, the validity and plausibility of all the used inputs and outputs (i.e., the crane's hardware) connected to the PLC are checked. If there is a hardware-level problem, the controller enters the "Error" state, indicating that wirings should be re-checked. To run the program again, the PLC must be reset. If the crane’s hardware works properly, the system determines whether or not the load is hanging. If the load is not hanging, the PLC automatically resets the encoders that measure the load deviation angle. Otherwise, the angle encoders must be manually reset later. The next step is to execute the "Go home" command to reset the encoders that measure the load position (in x, y, and z – directions). Since the Inteco 3D crane uses incremental encoders that require a reference point, both types of encoders must be reset at the beginning. The PLC then enters the "Idle" state, waiting for the operator to set the "Start" flag. If the angle encoders were not reset automatically during initialization, the system asks the operator to reset them. The preceding describes the system start-up procedure. The crane is ready to work once the procedure is finished.

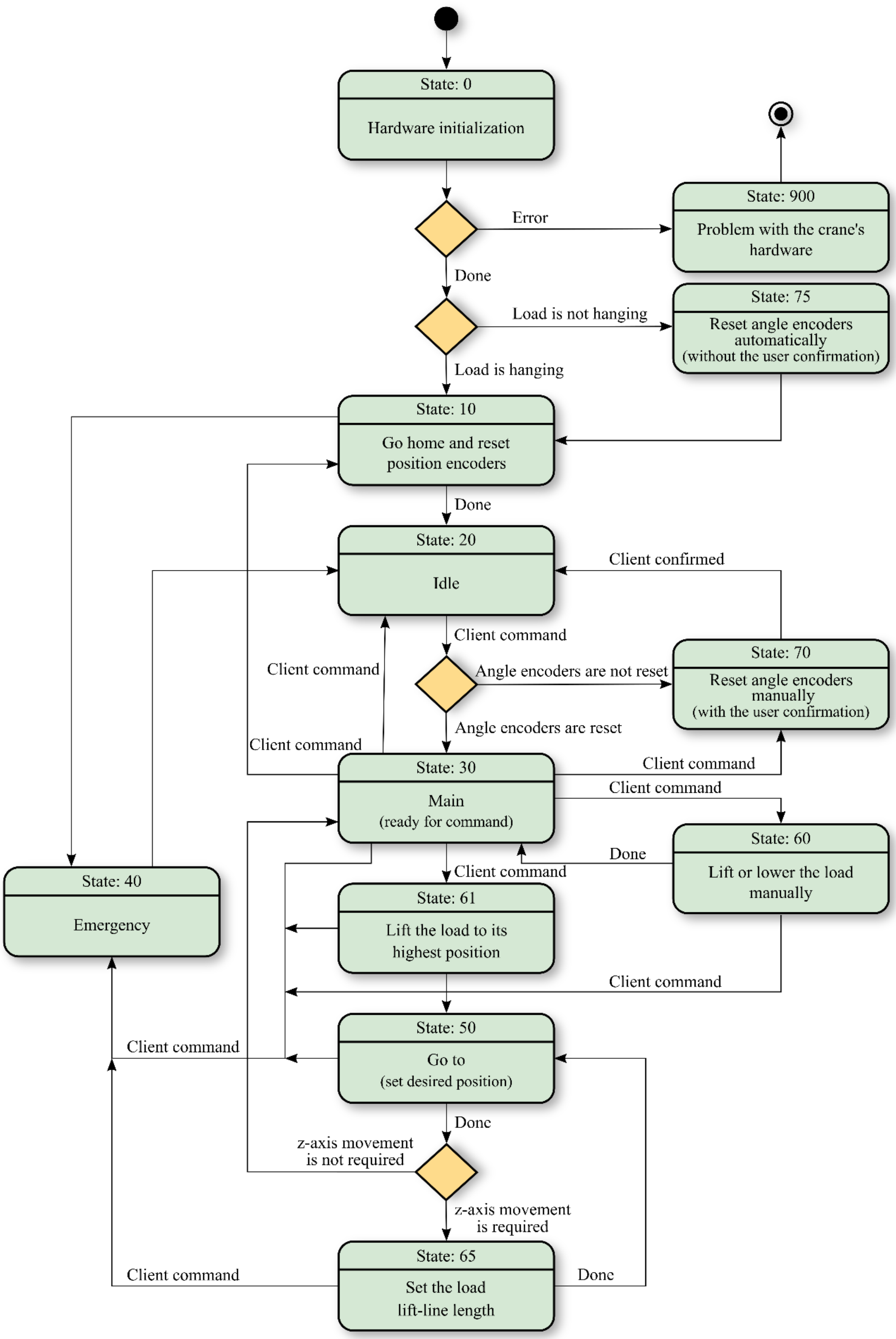

State: 0
Hardware initialization
Error
State: 900
Problem with the crane's hardware
Done
Load is not hanging
State: 75
Reset angle encoders automatically (without the user confirmation)
Load is hanging
State: 10
Go home and reset position encoders
Done
State: 20
Idle
Client confirmed
Client command
Angle encoders are not reset
State: 70
Reset angle encoders manually (with the user confirmation)
Client command
Angle encoders are reset
Client command
State: 30
Main (ready for command)
Client command
Client command
State: 60
Lift or lower the load manually
Done
Client command
State: 40
Emergency
State: 61
Lift the load to its highest position
Client command
State: 50
Go to (set desired position)
Client command
Done
z-axis movement is not required
z-axis movement is required
State: 65
Set the load lift-line length
Client command
Done

Figure 26. PLC side, state machine

The crane control is established through the use of a GUI that is implemented on the PC side. A detailed description of the GUI can be found in Section 5.4. On the PLC side, all commands devoted to load movement are carried out using the same logic. If the PLC is in the "Main" state (i.e., the "Start" flag is raised), the operator can request that the load be moved to any point within the crane's working area. When the PLC receives the "Go to" command, it compares the x, y, and z – coordinates of the desired set point (stored in the corresponding Modbus registers) to the current load position. If the values differ, the PLC starts executing the command. First, it checks whether the load is in its highest position. If this is not the case, the crane lifts the load to reduce load oscillations and avoid the risk of load collision (due to height) while moving. The cart is then driven in the x and y – directions to the set point. When the cart reaches the desired position, the system checks whether or not the load has to be lowered. If the corresponding control bit is set, the system lowers the load to the required length of lift-line. When the system is in the "Pick QR" operating mode, the crane picks or releases the load automatically at the end of the command.

When the system is in the "Main" state, the operator can manually lift, lower, pick, or release the load by pressing the corresponding buttons on the GUI. Because of the static error that accumulates after using the application for a while, it is necessary to reset both types of encoders on a regular basis. The corresponding buttons on the GUI are used to initiate these procedures. To handle the abnormal system behavior, there is the "Emergency" state. The transition to this state can be initiated from any other state associated with load movement (i.e., when the motors are active) by pressing the corresponding button.

The description of mentioned flags and commands can be found in Table 4.

## 5.3 PC side development

As mentioned in Section 3.2.2, it was decided to use Python for programming the outside loop. The selected development environment is the PyCharm [101]. It has a free community edition that includes all of the tools needed to implement the logic according to the proposed architecture.

### 5.3.1 PC responsibilities

Python is not a real-time programming language. However, due to its popularity, it has a great number of open-source frameworks, libraries, and development tools that significantly reduce the time required for project development. Python is widely used for prototyping, data analysis, machine learning, and etc. Furthermore, it provides effective methods and tools for connecting to external services. Considering the proposed architecture, the tasks that PC is responsible for are the following:

- Process the images taken by the camera and extract all the relevant data
- Receive and interpret the feedback from PLC
- Provide the operator with all the useful information by means of GUI
- Allow the operator to control the crane by means of GUI
- Wrap and send the requested commands for controlling the crane

### 5.3.2 PC side implementation

The part of the system that is meant to run on a PC is programmed using a modular approach. The basic idea behind this method is to divide the program into separate sub-programs known as modules. Each module is a self-contained piece of code that can run on its own and, if necessary, be called by the main script. Modular programming simplifies project maintenance, makes code more readable and allows modules to be reused in other applications [102]. Another benefit is that if a hardware component used in the architecture needs to be replaced, only the associated module has to be modified, rather than the entire program. Considering the current setup, the program is divided into six modules, as illustrated in Figure 27.

The main module is at the heart of the program. It is designed to combine all of the sub-modules into a single program. When the main module starts, the GUI with all of the underlying functionality appears. The GUI is described in detail in Section 5.4. The corresponding module is implemented in PyQt5, which is a comprehensive set of Python bindings for Qt v5 [103]. Qt is one of the most powerful and widely used cross-platform GUI libraries. It is written in C++. Qt is compatible with all major operating systems, including Windows, Linux, and Mac OS [104].

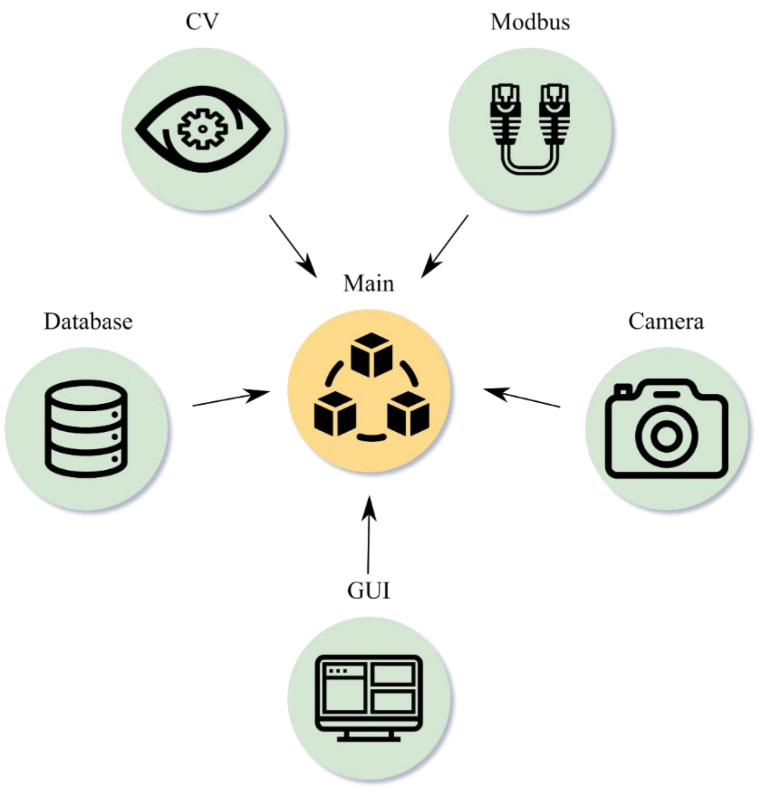


Figure 27. PC side, modular approach

The Camera module is intended to communicate with the Intel® RealSense™ LiDAR Camera L515 (see Table 2). The main purpose of this module is to provide visual input to the system by capturing the crane's working area. The module can also be used to configure the RGB sensor parameters. The parameters in use can then be stored in a database.

The Modbus module is intended to realize the part of communication logic for which the PC is responsible. This module makes it possible to connect to the PLC, retrieve the contents of its Modbus registers, unwrap and interpret the data that is received. Conversely, using this module, set commands and parameters are wrapped and sent to the PLC’s Modbus registers. Section 3.3.5 contains a detailed description of the Python library selected for this module implementation, the PC configuration process, and the communication possibilities with the PLC.

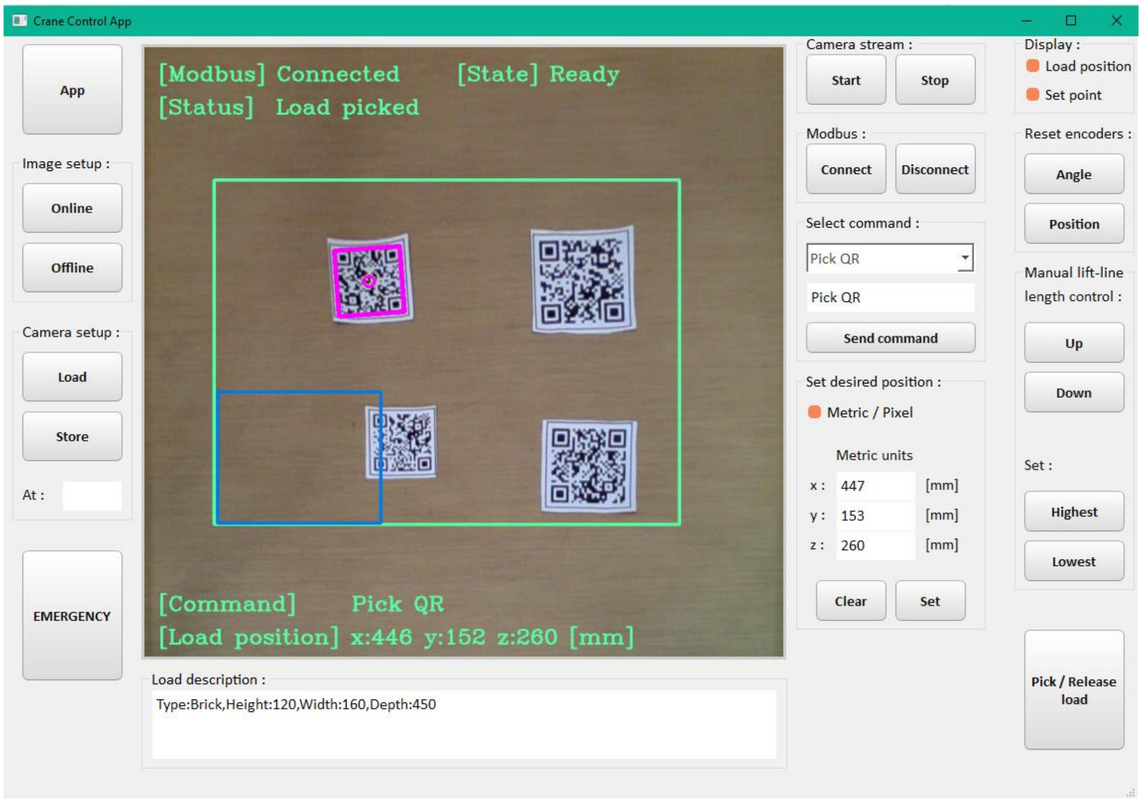
The CV module is responsible for processing the frames received from the Camera module and extracting all relevant data from them. Section 4.2 describes in detail the image processing pipeline used for the required data extraction. Currently, the main purpose of this module is to estimate the x, y – coordinates of the load and convert them to the crane’s coordinate plane. The content of QR codes, which are used to match the load with information about itself, is also decoded within this module. Furthermore, the CV module allows setting the desired cart position with a mouse click.

The Database module is intended to store the parameters used in the image processing pipeline (performed within the CV module) and the RGB sensor parameters (applied within the Camera module). This module is written in sqlite3, which is part of the standard library (since Python 2.5) [105]. SQLite is a file-based SQL database that is self-contained, lightweight, and open-source. SQLite is implemented in C, making the manipulations with the database fast and reliable [106].

## 5.4 Crane control GUI

An application with a GUI is used to control the crane. The implemented GUI is divided into three pages. The main page is used to send commands to the PLC and observe the feedback. It contains all of the information and controls required to operate the crane (see Figure 28). The "App" button on the left GUI panel is used to set the main page. The other two pages are used to configure the hard-coded parameters used in the image processing pipeline. The first page is intended for “online” configuration (i.e., using frames from the live video feed), whereas the second is designed for “offline” configuration (i.e., using the stationary images captured beforehand). The "online" and "offline" configuration pages are set using the corresponding buttons on the left GUI panel. Both pages are very similar. The only difference is in the controls on the bottom panel that are used to set the source for the image to be processed and displayed in the application. The “online” configuration page can be seen in Figure 29.

There is a need for two configuration methods since initial parameter selection (from the ground up) is often performed much more efficiently and conveniently using stationary images, whereas quick tuning is more preferable using the live video feed. The main reason behind this is that switching between images captured from different camera positions takes far less time than moving the camera back and forth. The proposed system assumes variations in camera position, hence the parameters used in the image processing pipeline should be general. Furthermore, each frame contains noise that is distributed unevenly. In other words, the RGB values provided by the camera are not stable, primarily due to the changing environment (i.e., illuminant flickering). It complicates the configuration process, as the results provided by the algorithm at certain frames may confuse the operator. The procedure for configuring hard-coded parameters is described in Section 4.2.4.



[Modbus] Connected [State] Ready
[Status] Load picked

[Command] Pick QR
[Load position] x:446 y:152 z:260 [mm]

Select command : Pick QR

Pick QR

Metric units
x : 447 [mm]
y : 153 [mm]
z : 260 [mm]

Load description :
Type:Brick,Height:120,Width:160,Depth:450

Figure 28. GUI, main page

To start operating the crane, the user should first initialize the camera and connect to the PLC. This can be done using the corresponding buttons on the right GUI panel of the main page. However, it is recommended to verify that the corner points (see Figure 14) are found correctly and confirm them beforehand. The operator has to go to the "online" configuration page to accomplish this. To initialize the camera, the “Start” button has to be pressed. The received frames are processed using the algorithm described in Section 4.2.2, and the result is displayed over the original frames, as shown in Figure 29. Once the operator sees that the corner points have been extracted correctly, they can be confirmed by clicking the corresponding button. It stops the corner points extraction algorithm, and the last found points are then used while the application is running. The problem caused by illuminant flickering is thus resolved. Furthermore, it reduces the load on the CPU. The issue at hand is discussed in Section 4.2.3. If the camera needs to be replaced while the application is running, the corner points extraction algorithm can be restarted by pressing the "Release" button. After the corner points have been confirmed, the operator may return to the main page.

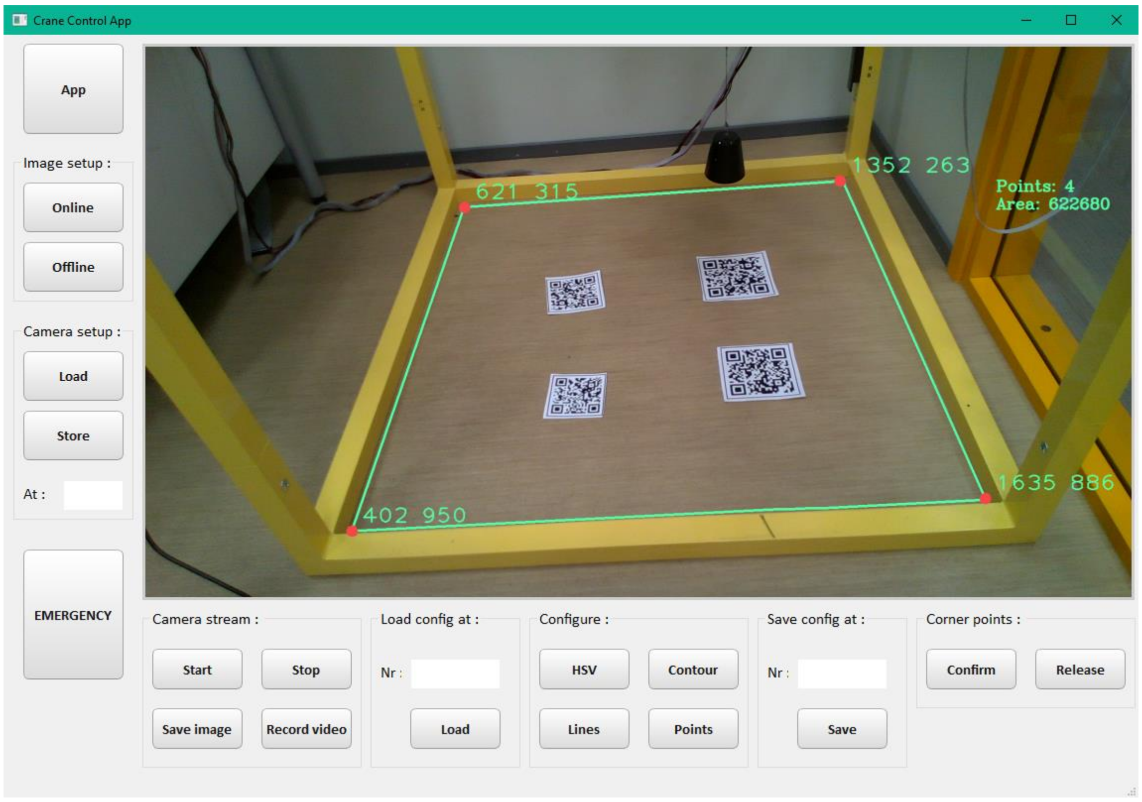


Figure 29. GUI, online configuration page

The main page is organized as follows. The major part is taken up by an interactive image that depicts the crane's working area in real-time. The displayed image shows the working area after warping the perspective (see Figure 23). The image is clickable. It enables the operator to set the desired position or select the load to pick using a mouse. Within the corresponding entries, the selected location is shown in numeric form. The operator could use either metric or "pixel" units. The "Metric / Pixel" checkbox switches between them. Because of the possibility of a misclick, the desired position should be confirmed with the "Set" button. This button is used to send these values to the PLC. To clear the desired position on the PLC side, the "Clear" button has to be pressed. The operator can draw the entered set position over the displayed image by selecting the "Set point" checkbox. The "Load position" checkbox is used to display the current load position as measured by the position encoders. The desired set point is represented by a rose circle, and the current load position is represented by a yellow circle.

The feedback from the PLC, which contains all of the information needed to operate the crane, is displayed over the main image. The information is divided into five entries, as shown in Figure 28. The following information is displayed:

- Modbus – the status of the connection between the PC and PLC
- State – the internal state of the PLC
- Status – the decoded contents of the status bits Modbus register (see Table 5)
- Command – the command that is currently set in the Modbus register
- Load position – the current load position in metric units

To force the PLC to execute a command, the operator must either select it from a dropdown list or type it manually into the corresponding entry. The command should then be sent by pressing the "Send command" button. In the case of the "Go to" and "Go to and back" commands, the desired set position must be specified in advance (i.e., the corresponding entries should be filled). If there is no need to set the lift-line length, the corresponding entry can be left blank. It will automatically be filled with zero. It may be preferable to manually adjust the lift-line length in some cases. For this purpose, there are corresponding "Up" and "Down" buttons on the right GUI panel. It is also possible to lift or lower the load to its maximum by using the "Highest" and "Lowest" buttons, respectively. The "Pick / Release load" button is self-explanatory, and it is used to either pick or release the load, depending on the current status. In case of abnormal system behavior, the operator can immediately turn off all motors by pressing the “Emergency” button on the right GUI panel.

As mentioned in Section 5.2.2, both types of encoders must be reset on a regular basis. To initiate the procedure for resetting the encoders that measure the load deviation angle, the “Angle” button has to be pressed. Following that, the operator must confirm in the pop-up window that the load is perpendicular to the floor and not hanging. To reset the position measuring encoders, the “Position” button has to be pressed.

The developed application allows for the storage and retrieval of RGB sensor parameters. These parameters are stored in a database. The RGB sensor parameters can be adjusted by running the Camera module – from the command line (see Section 5.3.2). However, it is more convenient to configure them using the vendor's "Intel RealSense Viewer" application. The set parameters are preserved until the camera is reloaded.

The following is a description of how the semi-automated “Pick QR” command works. After selecting the load, its center point coordinates are placed into the entries intended for setting the desired position. The required lift-line length is automatically set based on the load height that is encoded in the QR code. The operator should confirm the set point by pressing the "Set" button. To initiate the cart movement to the desired position, the operator should send the "Pick QR" command to the PLC. It can be done by pressing the “Send command” button. When the crane reaches the desired position, it automatically picks up the selected load. Once the load is picked, the system suggests an area to move it based on its type. The suggested area is depicted as a blue rectangle overlaid on top of the main image. The operator must then enter the desired position for the load to be released, confirm it, and send the "Pick QR" command again. When the crane reaches the desired position, the load is automatically released. The "Load description" entry displays the load information encoded in the QR code.

# 6 Summary

The aim of this thesis was to integrate CV technology into the existing 3D crane control system. To accomplish this, two nodes (the camera and the PC) were added to the initial system architecture. The role of the main controller has been moved from the PLC to the PC. It opened up a wide range of possibilities for system improvement. For example, it gave more freedom in designing the GUI, making the crane operation process more interactive and user-friendly.

Since the resulting system includes two computing devices (the PC and PLC) with significantly different program execution speeds, an appropriate control loop was implemented. The responsibilities of the PC and the PLC are divided so that all time- or safety-critical functionality is implemented on the PLC. The communication logic, which is primarily based on a system of checks and flags, was designed preserving the room for future development.

For implementing the CV part of the system, both the DL-based and classic CV approaches were tried out. Since the proposed system is meant to be used indoors, and the traditional CV performed better in these conditions, it was decided to stick with this method.

The proposed system was evaluated and tested in a laboratory environment. The results were satisfactory. The implemented control loop worked well, and there were no problems with communication between the controllers. There were no issues found with the corner points extraction algorithm as well. The corner points were extracted accurately enough, and there was no need to precisely find the right camera's FOV. The accuracy in estimating the coordinates of the load’s center point was high. However, the accuracy dropped slightly when it came to locating the center points of loads that were closer to the camera's nearest crane frame edge. It's very likely because of the perspective transformation performed prior to QR code detection. Considering the aforementioned, it is reasonable to conclude that the proposed system is viable.

## 6.1 Future work

Despite the good results obtained so far, the system still has a lot of room for future development. Currently, the system may only recognize cargos marked with a QR code. Since the practice of marking cargo with QR codes is not yet widespread in the industry, the system should also be able to detect and estimate the dimensions of an unmarked cargo (in order to find its center point). Another aspect to work on is detecting people in the working area and implementing the safe-route planner for estimating cart movement trajectory. The route planner should avoid situations in which the crane moves above a person. The author believes that the DL-based approach is preferable for resolving CV-related issues associated with object detection (including people). Hence, it may be wise to consider developing an algorithm that would automatically capture the crane's working area under certain conditions during crane operation. This will eliminate the need to gather training data in the future.

Another thing to consider is that the corner point extraction algorithm is currently highly dependent on hard-coded parameters that are tuned based on environmental conditions. To address this issue, image preprocessing should employ a color normalization algorithm. Hence, the color mapping function should be determined, which can be done using the DNN. Once the mapping function is found, even if a frame is captured under unknown illumination, the constituent colors could be transformed to ones as if they were captured under predetermined illumination.

# References

[1] J. Lee, H. Davari and V. Pandhare, "Industrial Artificial Intelligence for industry 4.0-based manufacturing systems," *Elsevier,* vol. 18, no. 5, pp. 20-23, 2018.

[2] S. Friscia, "A new patent for reaching new heights in crane automation," 27 10 2020. [Online]. Available: https://blog.se.com/mining-metals-minerals/2020/10/27/a-new-patent-for-reaching-new-heights-in-crane-automation/. [Accessed 25 09 2021].

[3] VOCA, "Optilift Autonomous Crane Control System," 2021. [Online]. Available: https://voca.no/optilift-autonomous-crane-control-system/. [Accessed 26 09 2021].

[4] INTSITE Ltd, "OUR TECHNOLOGY," 2021. [Online]. Available: https://www.intsite.ai/technology/. [Accessed 26 09 2021].

[5] Konecranes, "Five benefits of remote monitoring for cranes," 2021. [Online]. Available: https://www.konecranes.com/. [Accessed 26 09 2021].

[6] M. Moissejev, "3D Crane Control Using ABB AC500 Controller," [Online]. Available: https://digikogu.taltech.ee/en/Item/b7e62ef0-f925-4286-9668-039261b9ecaa.

[7] Inteco, "3DCrane User’s Manual," 2012.

[8] E. Oztemel and S. Gursev, "Literature review of Industry 4.0 and related technologies," *Journal of Intelligent Manufacturing,* vol. 31, pp. 127-182, 2020.

[9] B. Mrugalska and M. WyrwickaPoznan , "Towards Lean Production in Industry 4.0," in *7th International Conference on Engineering, Project, and Production Management*, 2017.

[10] S. Wang, J. Wan, D. Li and C. Zhang, "Implementing Smart Factory of Industrie 4.0: An Outlook," *International Journal of Distributed Sensor Networks,* vol. 12, no. 1, 2016.

[11] A. Schumachera, S. Erol and W. Sihn, "A maturity model for assessing Industry 4.0 readiness and maturity of manufacturing enterprises.," *Procedia CIRP,* vol. 52, p. 256–261, 2018.

[12] J. Qin, Y. Liu and R. Grosvenor, "4.0, A Categorical Framework of Manufacturing for Industry," *Procedia CIRP,* vol. 52, pp. 173-178, 2016.

[13] R. G. Lins, P. R. Araujoa and M. Corazzim, "In-process machine vision monitoring of tool wear for Cyber-Physical Production Systems," *Robotics and Computer Integrated Manufacturing,* vol. 61, 2020.

[14] K. Hyungjung, J. Woo-Kyun, C. In-Gyu and A. Sung-Hoon, "A Low-Cost Vision-Based Monitoring of Computer Numerical Control (CNC) Machine Tools for Small and Medium-Sized Enterprises (SMEs)," *Sensors,* no. 19, 2019.

[15] Hill Crane, "The Use of Crane Offers Numerous Advantages!," 2014. [Online]. Available: https://hillcrane.com/use-crane-offers-numerous-advantages/. [Accessed 01 10 2021].

[16] Acculift, "Top Running And Under-Hung Bridge Crane Configurations Explained," [Online]. Available: https://acculift.com/over-under-hung-bridge-crane-configurations/. [Accessed 01 10 2021].

[17] P. Hyla, "The crane control systems: A survey," in *17th International Conference on Methods and Models in Automation and Robotics*, 2012.

[18] J. Vaughan, E. Maleki and W. Singhose, "Advantages of using command shaping over feedback for crane control," in *Proceedings of the 2010 American Control Conference*, 2010.

[19] C. Mi, Z. Zhang, Y. Huang and Y. Shen, "A fast automated vision system for container corner casting recognition," *Journal of Marine Science and Technology,* vol. 24, no. 1, 2016.

[20] M. S. Rahman, "Machine Vision Techniques for Crane Workspace Mappin," ProQuest, 2015.

[21] Z. Yang, Y. Yuan, M. Zhang, X. Zhao, Y. Zhang and B. Tian, "Safety Distance Identification for Crane Drivers Based on Mask R-CNN," *Sensors*, vol. 19, no. 12, 2019.

[22] W. Singhose and K. C. C. Peng, "Crane Control Using Machine Vision and Wand Following," in *IEEE International Conference on Mechatronics*, 2009.

[23] P. Hyla and J. Szpytko, "The Vision Technique Concept Support Crane Safety Exploitation Process," *Journal of Konbin,* vol. 49, 2019.

[24] Intel Corporation, "Intel RealSense LiDAR Camera L515," [Online]. Available: https://www.intelrealsense.com/lidar-camera-l515/. [Accessed 06 03 2021].

[25] ABB, "AC500 PLC," [Online]. Available: https://new.abb.com/plc/programmable-logic-controllers-plcs/ac500. [Accessed 09 03 2021].

[26] ABB, "PM590-ETH," [Online]. Available: https://new.abb.com/products/1SAP150000R0271/pm590-ethac500-prog-log-controller-2mb. [Accessed 06 03 2021].

[27] ABB, "TA524," [Online]. Available: https://new.abb.com/products/1SAP180600R0001/ta524ac500-dummy-coupler-module. [Accessed 06 03 2021].

[28] ABB, "DA501," [Online]. Available: https://new.abb.com/products/1SAP250700R0001/da501s500-digital-analog-i-o-module. [Accessed 06 03 2021].

[29] ABB, "CD522," [Online]. Available: https://new.abb.com/products/1SAP260300R0001/cd522s500-2xencoder-module-2xpwm-output. [Accessed 06 03 2021].

[30] Inteco, "3D Crane," [Online]. Available: http://www.inteco.com.pl/products/3d-crane/. [Accessed 06 03 2021].

[31] Inteco, "PLC to 3D Crane interface," 2012.

[32] PLCopen, "IEC 61131-3," [Online]. Available: https://plcopen.org/iec-61131-3. [Accessed 20 03 2021].

[33] J. Payne, "IEC 61131-3: What’s the acceptance rate of this control programming standard?," 12 2015. [Online]. Available: https://www.controleng.com/articles/iec-

61131-3-whats-the-acceptance-rate-of-this-control-programming-standard/. [Accessed 20 03 2021].

[34] PLC Academy, "Scan time of the PLC program," 2015. [Online]. Available: https://www.plcacademy.com/scan-time-of-the-plc-program/. [Accessed 23 03 2021].

[35] Mindfire Solutions, "Advantages and Disadvantages of Python Programming Language," 04 2017. [Online]. Available: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121. [Accessed 07 04 2021].

[36] A. Foong, "NASA, Google, FB, Netflix – What Do They Have In Common?," 05 2020. [Online]. Available: https://thelead.io/data-science/companies-that-uses-python. [Accessed 07 04 2021].

[37] TechVidvan Team, "Python Advantages and Disadvantages – Step in the right direction," 03 2021. [Online]. Available: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/. [Accessed 07 04 2021].

[38] S. Charrington, "Artificial Intelligence for Industrial Applications," *CloudPulse Strategies,* 2017.

[39] W. Gastreich, "What is Modbus?," 12 2018. [Online]. Available: https://realpars.com/modbus/. [Accessed 03 02 2021].

[40] Modbus Organization, "Modbus protocol," 2005. [Online]. Available: https://www.modbus.org/specs.php. [Accessed 04 02 2021].

[41] National Instruments corp., "The Modbus Protocol In-Depth," 02 2021. [Online]. Available: https://www.ni.com/en-us/innovations/white-papers/14/the-modbus-protocol-in-depth.html. [Accessed 06 02 2021].

[42] Agilent Technologies, "Choosing between serial and Ethernet," 09 2005. [Online]. Available: http://www.velocity11.com/techdocs/helpsystem/vworks_ug/choosingbetweenserialandethernet.html. [Accessed 10 02 2021].

[43] D. Bohn, "Modbus RTU verses Modbus TCP/IP: What's the Difference?," 06 2017. [Online]. Available: https://www.linkedin.com/pulse/modbus-rtu-verses-tcpip-whats-difference-david-bohn. [Accessed 12 02 2021].

[44] J. S. Rinaldi, "Modbus TCP vs. Modbus RTU," 10 2013. [Online]. Available: https://www.rtautomation.com/rtas-blog/modbus-tcp-vs-modbus-rtu/. [Accessed 12 02 2021].

[45] Real Time Automation, "Modbus TCP/IP," 07 2014. [Online]. Available: https://www.rtautomation.com/technologies/modbus-tcpip/. [Accessed 14 02 2021].

[46] IPC2U Group, "Detailed description of the Modbus TCP protocol with command examples," 04 2017. [Online]. Available: https://ipc2u.com/articles/knowledge-base/detailed-description-of-the-modbus-tcp-protocol-with-command-examples/. [Accessed 17 02 2021].

[47] ABB Group, "Automation Builder online help system".

[48] Modbus Organization, "Modbus application protocol specification," 04 2012. [Online]. Available: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. [Accessed 08 02 2021].

[49] L. Benbenishti, "SCADA Modbus Protocol Vulnerabilities," 04 2017. [Online]. Available: https://www.cyberbit.com/blog/ot-security/scada-modbus-protocol-vulnerabilities/. [Accessed 20 02 2021].

[50] ABB, "ABB Automation Builder," [Online]. Available: https://new.abb.com/plc/automationbuilder. [Accessed 07 03 2021].

[51] InSAT Company, "Modbus Universal MasterOPC сервер," 2014. [Online]. Available: https://insat.ru/products/Universal_MasterOPC/ABB_modbus_universal.pdf. [Accessed 20 04 2021].

[52] Riptide Team, "PyModbus - A Python Modbus Stack," 02 2021. [Online]. Available: https://github.com/riptideio/pymodbus. [Accessed 25 02 2021].

[53] Z. Zasieczny, "Python modbus library," 06 2020. [Online]. Available: https://stackoverflow.com/questions/17081442/python-modbus-library. [Accessed 25 02 2021].

[54] l.lefebvre, "pyModbusTCP," 03 2021. [Online]. Available: https://github.com/sourceperl/pyModbusTCP. [Accessed 15 03 2021].

[55] luc.jean, "modbus-tk: Create Modbus app easily with Python," 05 2020. [Online]. Available: https://github.com/ljean/modbus-tk. [Accessed 01 03 2021].

[56] N. O'Mahony, S. Campbell, A. Carvalho, L. Krpalkova, G. H. Velasco, S. Harapanahalli, D. Riordan and J. Walsh, "Deep Learning vs. Traditional Computer Vision," *Advances in Intelligent Systems and Computing*, vol. 943, pp. 128-144, 2020.

[57] Z. Zbigniew, "Why Deep Learning Has Not Superseded Traditional Computer Vision," 09 03 2018. [Online]. Available: https://zbigatron.com/has-deep-learning-superseded-traditional-computer-vision-techniques/. [Accessed 12 10 2021].

[58] Z. Zbigniew, "The Reasons Behind the Recent Growth of Computer Vision," 26 01 2018. [Online]. Available: https://zbigatron.com/the-reasons-behind-the-recent-growth-of-computer-vision/. [Accessed 12 10 2021].

[59] R. Talluri, "Conventional computer vision coupled with deep learning makes AI better," Network World, 29 11 2017. [Online]. Available: https://www.networkworld.com/article/3239146/conventional-computer-vision-coupled-with-deep-learning-makes-ai-better.html. [Accessed 12 10 2021].

[60] OpenCV team, "About," 04 11 2020. [Online]. Available: https://opencv.org/about/. [Accessed 27 10 2021].

[61] Danny, "Why is color segmentation easier on HSV?," 9 05 2009. [Online]. Available: https://stackoverflow.com/questions/16472716/why-is-color-segmentation-easier-on-hsv. [Accessed 29 10 2021].

[62] A. Rosebrock, "OpenCV Smoothing and Blurring," 28 04 2021. [Online]. Available: https://www.pyimagesearch.com/2021/04/28/opencv-smoothing-and-blurring/. [Accessed 29 10 2021].

[63] Saisha, "Finding the Edge: Canny and Sobel Edge Detectors (Part 1)," 06 09 2020. [Online]. Available: https://medium.com/srm-mic/finding-the-edge-canny-and-sobel-detectors-part-1-65a59b7ef62a. [Accessed 30 10 2021].

[64] Scikit-image development team, "Straight line Hough transform," 2019. [Online]. Available: https://scikit-

image.org/docs/dev/auto_examples/edges/plot_line_hough_transform.html. [Accessed 31 10 2021].

[65] OpenCV team, "Hough Line Transform," 05 07 2021. [Online]. Available: https://docs.opencv.org/4.5.3/d3/de6/tutorial_js_houghlines.html. [Accessed 31 10 2021].

[66] C. Luengo, "What algorithm is implemented in OpenCV contour detection? [closed]," 11 12 2018. [Online]. Available: https://stackoverflow.com/questions/9334166/what-algorithm-is-implemented-in-opencv-contour-detection. [Accessed 01 11 2021].

[67] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing,* vol. 30, no. 1, pp. 36-46, 04 1985.

[68] OpenCV team, "Structural Analysis and Shape Descriptors - approxPolyDP()," 05 07 2021. [Online]. Available: https://docs.opencv.org/4.5.3/d3/dc0/group__imgproc__shape.html#ga0012a5fdaea70b8a9970165d98722b4c. [Accessed 02 11 2021].

[69] OpenCV team, "Geometric Image Transformations," 05 07 2021. [Online]. Available: https://docs.opencv.org/4.5.3/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87. [Accessed 02 12 2021].

[70] itemit, "QR Codes Vs. Barcodes," 05 10 2021. [Online]. Available: https://itemit.com/qr-codes-vs-barcodes-which-is-best/. [Accessed 03 12 2021].

[71] J. Brown, "ZBar bar code reader," 15 07 2021. [Online]. Available: http://zbar.sourceforge.net/. [Accessed 03 12 2021].

[72] J. Hietbrink, "Kraken, the unknown Python OCR system," 01 10 2016. [Online]. Available: https://www.webuildinternet.com/2016/10/01/kraken-the-unknown-python-ocr-system/. [Accessed 03 12 2021].

[73] B. Kiessling, "kraken," 26 02 2019. [Online]. Available: http://kraken.re/master/index.html. [Accessed 03 12 2021].

[74] OpenCV team, "cv::Moments Class Reference," 05 07 2021. [Online]. Available: https://docs.opencv.org/4.5.3/d8/d23/classcv_1_1Moments.html#a0382b98fdb23acdcb05c91a2a44e5a1f. [Accessed 04 12 2021].

[75] K. Bapat, "Find the Center of a Blob (Centroid) using OpenCV (C++/Python)," 19 07 2018. [Online]. Available: https://learnopencv.com/find-center-of-blob-centroid-using-opencv-cpp-python/. [Accessed 04 12 2021].

[76] J. Brownlee, "9 Applications of Deep Learning for Computer Vision," 13 03 2019. [Online]. Available: https://machinelearningmastery.com/applications-of-deep-learning-for-computer-vision/. [Accessed 16 10 2021].

[77] I. Papastratis, "Deepfakes: Face synthesis with GANs and Autoencoders," 02 06 2020. [Online]. Available: https://theaisummer.com/deepfakes/. [Accessed 16 10 2021].

[78] C. Long, "What Is Computer Vision? Why Deep Learning Changed It All," dynam.ai™, 15 01 2021. [Online]. Available: https://www.dynam.ai/what-is-computer-vision-technology/. [Accessed 17 10 2021].

[79] J. Nelson, "What is Active Learning?," 22 11 2020. [Online]. Available: https://blog.roboflow.com/what-is-active-learning/. [Accessed 18 10 2021].

[80] K. Goyal, "Top 10 Deep Learning Frameworks in 2021 You Can’t Ignore," 10 01 2021. [Online]. Available: https://www.upgrad.com/blog/top-deep-learning-frameworks/. [Accessed 19 10 2021].

[81] Roboflow, Inc, "Our Company," 14 10 2021. [Online]. Available: https://roboflow.com/about. [Accessed 19 10 2021].

[82] F. Fahim, "CPU vs GPU vs TPU: Understanding the Difference Between Them," 15 06 2021. [Online]. Available: https://serverguy.com/comparison/cpu-vs-gpu-vs-tpu/. [Accessed 19 10 2021].

[83] D. Takahashi, "Ambarella launches computer vision chips for edge AI," Ambarella, 02 11 2020. [Online]. Available: https://venturebeat.com/2020/11/02/ambarella-launches-computer-vision-chips-for-edge-ai/. [Accessed 17 10 2021].

[84] IBM Corporation, "IBM Watson Studio," 13 10 2021. [Online]. Available: https://www.ibm.com/cloud/watson-studio. [Accessed 20 10 2021].

[85] Tracxn Technologies, "Emerging Startups 2021: Top Computer Vision Startups," 15 09 2021. [Online]. Available: https://tracxn.com/d/emerging-startups/top-computer-vision-startups-2021. [Accessed 21 10 2021].

[86] Y. Wu, F. Massa, R. Girshick, A. Kirillov and W.-Y. Lo, "Detectron2: A PyTorch-based modular object detection library," 10 10 2019. [Online]. Available: https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/. [Accessed 22 10 2021].

[87] D. Goncharov, "Performance benchmarking of TensorFlow and PyTorch," 30 06 2021. [Online]. Available: https://github.com/Chifffa/tf_vs_torch_benchmarking. [Accessed 22 10 2021].

[88] detectron2 contributors, "Benchmarks," 07 10 2021. [Online]. Available: https://detectron2.readthedocs.io/en/latest/notes/benchmarks.html. [Accessed 22 10 2021].

[89] R. Girshick, "torchvision vs detectron2," 19 10 2019. [Online]. Available: https://github.com/facebookresearch/detectron2/issues/117. [Accessed 22 10 2021].

[90] detectron2 contributors, "Installation requirements," 10 10 2019. [Online]. Available: https://detectron2.readthedocs.io/en/latest/tutorials/install.html. [Accessed 22 10 2021].

[91] Yogeshkumarpilli, "How to Install Detectron2 on Windows 10 or 11 –2021(AUG) with the latest build(v0.5).," 02 08 2021. [Online]. Available: https://medium.com/@yogeshkumarpilli/how-to-install-detectron2-on-windows-10-or-11-2021-aug-with-the-latest-build-v0-5-c7333909676f. [Accessed 23 10 2021].

[92] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, "Detectron2," 2019. [Online]. Available: https://github.com/facebookresearch/detectron2. [Accessed 23 10 2021].

[93] detectron2 contributors, "Use Custom Datasets," 29 07 2021. [Online]. Available: https://detectron2.readthedocs.io/en/latest/tutorials/datasets.html. [Accessed 23 10 2021].

[94] A. Dutta, A. Gupta and A. Zisserman, "VGG Image Annotator (VIA)," 2019. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/software/via/. [Accessed 24 10 2021].

[95] Prabhu, "Understanding Hyperparameters and its Optimisation techniques," 03 07 2018. [Online]. Available: https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568. [Accessed 10 11 2021].

[96] detectron2 contributors, "defaults.py," 12 10 2021. [Online]. Available: https://github.com/facebookresearch/detectron2/blob/main/detectron2/config/defaults.py . [Accessed 10 11 2021].

[97] A. Credoz, "Inaccurate masks with Mask-RCNN: Stairs effect and sudden stops," 09 07 2020. [Online]. Available: https://stackoverflow.com/questions/62810854/inaccurate-masks-with-mask-rcnn-stairs-effect-and-sudden-stops. [Accessed 12 11 2021].

[98] J. Doe, "Mask RCNN detecting object but mask is inaccurate," 09 03 2019. [Online]. Available: https://datascience.stackexchange.com/questions/46972/mask-rcnn-detecting-object-but-mask-is-inaccurate. [Accessed 12 11 2021].

[99] A. Kirillov, Y. Wu, K. He and R. Girshick, "PointRend: Image Segmentation as Rendering," 2020.

[100] A. Kirillov, O. Parkhi and B. Cheng, "Pointly-Supervised Instance Segmentation," 29 09 2021. [Online]. Available: https://github.com/facebookresearch/detectron2/tree/main/projects/PointRend. [Accessed 12 11 2021].

[101] JetBrains, "PyCharm. The Python IDE for Professional Developers," 08 2020. [Online]. Available: https://www.jetbrains.com/pycharm/. [Accessed 01 05 2021].

[102] N. Agarwal, "Modular Approach in Programming," 09 2018. [Online]. Available: https://www.geeksforgeeks.org/modular-approach-in-programming/. [Accessed 20 04 2021].

[103] Riverbank Computing Limited, "PyQt5," 29 10 2021. [Online]. Available: https://pypi.org/project/PyQt5/. [Accessed 12 12 2021].

[104] J. Bodnar, "Introduction to PyQt5," 14 08 2020. [Online]. Available: https://zetcode.com/gui/pyqt5/introduction/. [Accessed 12 12 2021].

[105] A. Tiwari, "Introduction to SQLite in Python," 03 07 2020. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-sqlite-in-python/. [Accessed 13 12 2021].

[106] SQLite Development team, "What Is SQLite?," 27 11 2021. [Online]. Available: https://www.sqlite.org/index.html. [Accessed 13 12 2021].

# Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Maksim Moissejev

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "3D crane control using ABB PLC and Computer Vision", supervised by Kristina Vassiljeva
   - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

02.01.2022

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

## Appendix 2 – Software dependencies

The project was implemented using the following development tools:

- PyCharm Community Edition (version 2021.1.3)
- ABB Automation Builder (version 2.3.0847)

Table 7 lists the Python packages that should be installed in order to build the developed application (that runs on the PC side). All the necessary libraries are included in these packages. The column "Version" shows the package release version used during project development. Python 3.8.12 was used as an interpreter, and the virtual environment was created with conda 4.10.1.

Table 7. Python packages that are required to build the application

| **Package** | **Version** |
|---|---|
| PyQt5 | 5.15.6 |
| opencv-python | 4.5.4.60 |
| pyrealsense2 | 2.50.0.3812 |
| pymodbus | 2.5.3 |
| pyzbar | 0.1.8 |
| kraken | 3.0.6 |

Even though the Detectron2-based CV part implementation is not required to build the application, the instructions below could be useful for future development or testing. Since Detectron2 does not provide official support for Windows OS, the following software and drivers with the specified versions should be installed. It is important to note that the given order must be strictly followed. It allows running the latest (at the time of this writing) Detectron2 build (v0.5) on Windows OS. The author verified this method and found it to be valid. It is recommended to create a new virtual environment, even if the libraries installed through this method are not related to those previously installed.

The first step is to install Anaconda. For some reason, the use of a standard Python venv module fails to properly install the packages (in relation to this method).

The second step is to install CUDA toolkit 10.2. It is required to use GPU for calculations.

The third step is to install PyTorch. The version should be compatible with CUDA 10.2.

The fourth step is to reinstall Microsoft Visual Studio. The only "Desktop development with C++" component, which can be found under the *Workloads* tab, is required. It is necessary to restart the system after the installation.

The fifth step is to install cython and pycocotools. Since errors may occur during the basic pycocotools installation, the version that works only on Windows OS should be used instead (pycocotools-windows).

The sixth step is to install Detectron2 from the official GitHub repository. If a new build will be released, version 0.5 should be specified. Otherwise, there is no guarantee that this method will work.

The final step is to install OpenCV and pywin32.