Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control System Engineering

Automatic laser marking system

Master Candidate

Contin Igino

Student ID 2054133

Supervisor

Prof. Ruggero Carli

University of Padova

Co-supervisor

Prof. Pedro Fonseca

Prof. Nuno Lao

University of Aveiro

Academic Year
2022/2023
Graduation Date
23/10/2023

To my parents
and friends

Abstract

The AdaptMark project aims to streamline and enhance the engraving process for a wide range of pieces within the company, improving marking quality and ensuring consistent engraving outcomes. The key objectives of the project include the development of an automatic laser marking system capable of identifying and marking various 3D pieces on a conveyor. The system utilizes a sensor for piece identification, followed by data processing and parameter adjustment for laser engraving. A robotic arm equipped with a laser performs the engraving process and check the quality of the mark. This thesis focuses on two main areas of the AdaptMark project: user interface development and system integration.

Sommario

Il progetto AdaptMark mira a semplificare e migliorare il processo di incisione per un'ampia gamma di pezzi all'interno dell'azienda, migliorando la qualità della marcatura e garantendo risultati di incisione coerenti. In particolare l'obiettivo è lo sviluppo di un sistema di marcatura laser automatico in grado di identificare e marcare vari pezzi 3D su un nastro trasportatore. Il sistema utilizza un sensore per l'identificazione del pezzo, seguito dall'elaborazione dei dati e dalla regolazione dei parametri per l'incisione laser. Un braccio robotico dotato di laser esegue il processo di incisione e controlla la qualità del marchio. Questa tesi si concentra su due aree principali del progetto AdaptMark: lo sviluppo dell'interfaccia utente e l'integrazione di sistema.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

UI	User Interface
GUI	Graphical User InterfaceTk
Tk	Tkinter
TCL	Tool Command Language
API	Application Programming Interface
DXF	Drawing Exchange Format
PNG	Portable Network Graphics
GOOD	Global Orthographic Object Descriptor
ROS	Robot Operating System
RANSAC	RANdom SAmple Consensus
ICP	Iterative Closest Point
ha	Height and Area
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
RDBMS	relational database management system
3D	Three dimensional
2D	Two dimensional
RGB	Red Green Blue

RGB-D	Red Green Blue - Depth
I/O	Input / Output
OO	Object-Oriented

1 Introduction

This thesis is based on the AdaptMark project, which is developed in collaboration between the JPM Industry and the University of Aveiro. Aim of this project is the development of a robotic system able to laser engrave pieces of different shapes and materials, and verify the quality of such engraving. In this chapter will be first present the framework of this project, followed by the company presentation, the motivation and the Objective, and finally the description of the structure of this document.

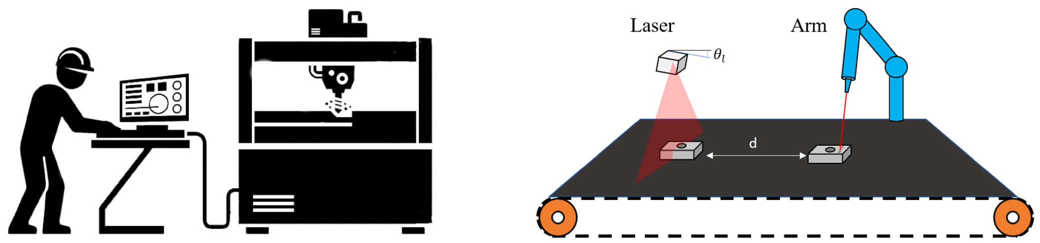
1.1 The company

JPM Industry, founded in 1994 in Vale de Cambra, Portugal, is a technologically advanced company specializing in industrial automation and metalworking. Over the years, the company has established itself as a leader in the production, installation, maintenance, and repair of industrial equipment and units. As part of the larger JPM Group, which includes JPM Renováveis and Jointsteel Process Technologies, JPM Industry operates in diverse business areas such as industrial automation, metalworking, and renewable energy. The company provides innovative engineering solutions, including automated industrial equipment, robotic systems, and energy supply systems using natural resources. JPM Industry serves a global clientele and is particularly focused on optimizing production and logistics processes for companies in the food, chemical, and pharmaceutical industries. With a solid track record and expertise in various fields, JPM Industry continues to deliver cutting-edge solutions and maintain strong partnerships with clients from diverse sectors.

https://jpm.pt/pt/inicio/

1.2 Framework

The JPM Industry operates a laser engraving line that is responsible for marking pieces produced using various materials and manufacturing methods. The engraving process typically involves applying an alphanumeric code and a logo onto the pieces. To achieve consistent engravings, JPM currently employs a laser and different templates. However, this process requires manual intervention from an operator. The operator's responsibilities include selecting the appropriate template, positioning the piece, configuring the laser parameters, and inspecting the final quality of the engraving. The variability of the laser parameters for different materials poses a significant challenge in this process. As a result, there have been instances where the marking does not meet the required standards. Within the engraving line, a wide variety of pieces arrive for marking, as depicted in Picture 3.10. Consequently, the operator must constantly adjust the laser parameters, leading to an increased possibility of errors. Additionally, the operator needs to conduct multiple marking tests to accurately calibrate the laser parameters for each new material. This process not only consumes time and energy but also results in material wastage.



(a) Current marking system (b) Proposed approach

Figure 1.1: The system contains a laser scanner to scan the piece moving on a conveyor, later a manipulator marks the piece using a laser beam.

1.3 Motivation

The motivation behind the development of the automatic laser marking system stems from the need to enhance productivity, efficiency, and accuracy within the laser engraving process. The company, in which this project is undertaken, currently operates a laser engraving line that requires constant manual intervention from operators. However, this manual intervention introduces several challenges and limitations. The dependence on human operators not only increases the likelihood of errors but also hampers the overall productivity of the system. The motivation behind the automatic laser marking system project is to overcome these challenges by developing an intelligent and automated solution. By leveraging advanced sensor technology, data processing algorithms, and robotic arm integration, the system aims to eliminate the need for constant operator intervention and ensure accurate and consistent laser marking on the 3D pieces. Furthermore, the motivation behind this project extends beyond the specific requirements of the company. The development of an automatic laser marking system has broader implications for industries that rely on laser engraving processes, such as manufacturing, automotive, and electronics. By automating the marking process, companies can reduce labor costs, improve production throughput, enhance product traceability, and achieve higher overall operational efficiency.

1.4 Objective

The objective of this thesis is to design, develop, and implement an automatic laser marking system that eliminates the need for manual intervention, reduces errors, and enhances productivity and accuracy in the laser engraving process. The primary goal is to create a fully automated system capable of identifying different 3D pieces running on a conveyor, processing the data, setting appropriate laser parameters, performing laser engraving using a robotic arm, and checking the final quality. Additionally, this system will incorporate machine learning algorithms to dynamically adjust laser parameters for optimal marking quality. The AdaptMartk project can be divided into three subsystems:

- **Identification subsystem** - to correctly identify the pieces running on the conveyor as well as the position and orientation.

- **Manipulation and engraving subsystem** - responsible of the engraving of the piece through the robotic arm and the laser.

- **Quality checking subsystem** - to check if the engraving satisfy a minimum quality.

The described system can be seen in the figure 1.2. At the current state of

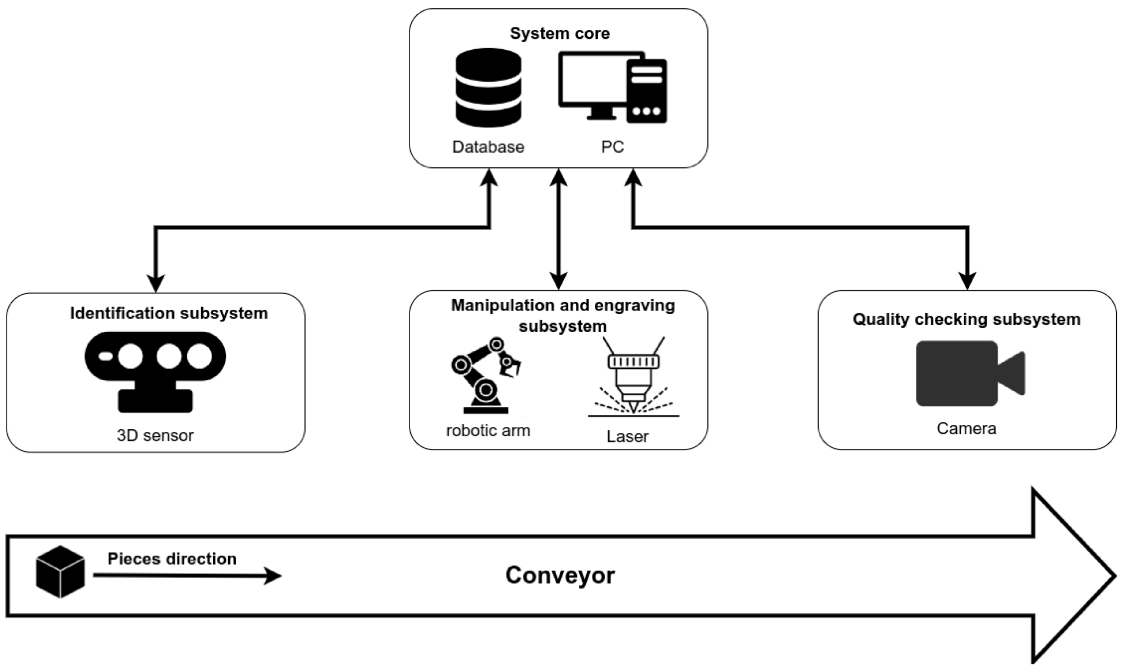


Figure 1.2: AdaptMark general scheme

the project, it's important to note that a significant portion has already been developed by other students at the University of Aveiro in collaboration with JPM Industry. Specifically, substantial progress has been made in the identification subsystem, including a thorough study of the technologies to be applied. Additionally, the quality check subsystem is already fully implemented. Keeping in mind the current state of the project, this thesis will cover the following points:

- Development of an user interface to interact with the system.

- Creation of a database to store the data of the all system.

- Connection of all the system parts.

1.5 Thesis structure

This thesis is organized into several chapters, each addressing a specific aspect of the automatic laser marking system. The structure of the thesis is as follows:

- **Chapter 1 - Introduction**: This chapter provides an overview of the company, presents the framework in which the project is conducted, outlines the motivation behind the development of the automatic laser marking system, and defines the objective of the thesis.
- **Chapter 2 - Technology and tools**: This chapter delves into the core elements that form the foundation of this thesis. It explores the key technologies and tools used throughout the development of the automatic laser marking system. These components enable data processing, 3D data representation, geometric transformations, ROS integration, and other functionalities crucial for the successful implementation of the system.
- **Chapter 3 - User Interface Design and Development**: This chapter explores the design principles and considerations for developing the user interface of the automatic laser marking system. It discusses the graphical elements, control functionalities, and usability aspects incorporated into the user interface. It also outlines the software development approach used for the implementation.
- **Chapter 4 - System Connection**: This chapter focuses on the integration and connection of various components within the automatic laser marking system. It discusses the communication protocols, hardware requirements, and software configurations that enable seamless interaction between different subsystems.
- **Chapter 5: Testing and Evaluation**: This chapter presents the testing methodology and the results obtained during the evaluation of the automatic laser marking system. It assesses the system's performance, accuracy, and reliability, validating its effectiveness in comparison to manual operations.
- **Chapter 6: Conclusion and Future Work**: The final chapter summarizes the achievements of the automatic laser marking system project, highlights the contributions made, and discusses the implications and potential future developments. It also addresses any limitations encountered during the project and suggests areas for further improvement and research.

2 Technology and tools

This chapter provides an overview of the various technologies and tools utilized in the development of the automatic laser marking system. It covers the libraries and algorithms used for data processing, 3D data representation, geometric transformations, ROS integration, and other relevant technologies that contribute to the functionality of the system.

2.1 3D Data Representation

The Three dimensional (3D) datasets are found in different forms that vary in both the structure and the properties. In this section, we reviewed the different categories of 3D data representation (see figure 2.1) which include: Raw data, solids, surfaces, multi views and high-level structures [3].

- **Raw data**: Raw 3D data can be obtained by different divergent scanning devices. Some of the popular 3D data representations that belong to this group are point cloud, Red Green Blue - Depth (RGB-D) data, and 3D projections
  - **Point Cloud**: A point cloud is simply a set of 3D data points, and each point is represented by three coordinates in a Cartesian or other coordinate systems. It is regarded as a set of unstructured 3D points that symbolize the geometry of 3D objects and are utilized in many computer vision tasks.
  - **RGB-D data**: RGB-D data is the union of Red Green Blue (RGB) data and a depth map D of the image. It combines color information (RGB) with the corresponding depth information (D), enabling the representation of both the color and depth aspects of a scene. This type

of representation is particularly valuable for tasks that require both color appearance and depth perception, such as object recognition and scene understanding.

  - **3D Data projections**: 3D projections are a way of mapping 3D points to Two dimensional (2D) planes. It is realized using imaginary projections which give the projected data crucial features of the original 3D object. Many of the projection methods convert the 3D model to a 2D grid with key properties.

- **Solids**: Solids representations of 3D models are virtually space control information for a given object. Usually the information is binary which implies that the space can be occupied by the object or none. The two major solids representations used in this category are Octree and Voxels.
- **Surfaces**: Surfaces polygons are usually used in boundary representation of 3D objects which surround the inner part of the object. The set of this polygons are usually stored for the description of the object which has the benefit of simplicity and speeding of the rendering of the surface and object display because all surfaces can be characterize with linear equations. There are many methods for surface representations of 3D objects such as the polygon mesh, sub division, parametric and implicit but among these representations polygon mesh is the most popular surface representation.
- **High-level structures**: 3D shapes can be represented in the form of high level 3D shape descriptors which is a simplified representation that contains the geometric characteristics of the 3D object.
- **Multi-view data**: Another form of 3D data representation is to render a set of images from verities of views and takes the pile image and use as an input to CNN which can be used for shape analysis tasks.

In this project, we exclusively utilize a raw Point Cloud as our chosen data representation. This decision is primarily driven by efficiency considerations, as other information such as color, while potentially useful in some contexts, could introduce unnecessary complexity and slow down the data processing pipeline.

2.2 3D Geometric Transformations

3D transformation is the process of manipulating the view of a three-dimensional object with respect to its original position by modifying its physical attributes through various methods of transformation, such as Translation, Scaling, Rotation, Shear, etc. [10]. In this section, we will focus on the translation and rotation transformations, which play a crucial role in the development of the automatic laser marking system.

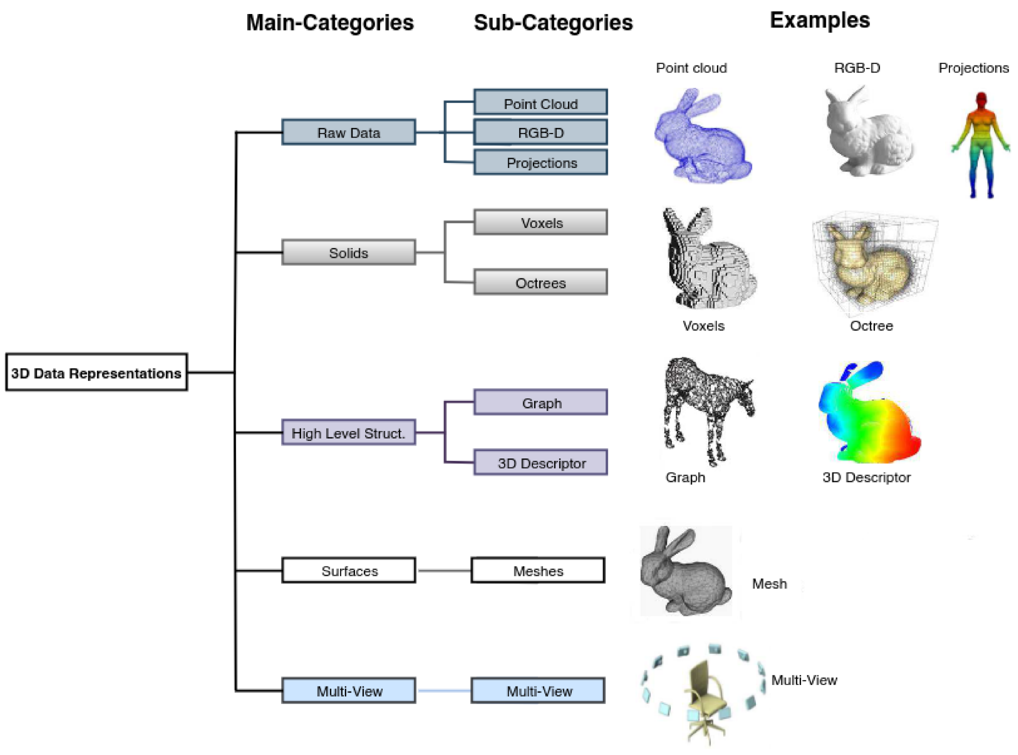


Figure 2.1: Different 3D data representations [3]

- **Translation** - Moving object is called translation, in 3 dimensional homogeneous coordinate representation , a point is transformed from position $P = (x, y, z)$ to $P' = (x', y', z')$. This can be written as:
  Using $P' = T \cdot P$:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1}$$

- **Rotation** - Rotation needs an angle and an axis, is defined according to the right-hand rule. Rotations are orthogonal matrices, preserving distances and angles.

  - Rotation about x-axis:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotation about y-axis:

$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– Rotation about z-axis:

$$R_x = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this project, only the translation and rotation around the Z-axis will be utilized for manipulating the view of the 3D objects.

2.3 Registration Algorithms

A registration algorithm, in the context of computer vision and 3D data processing, refers to a set of methods and techniques used to align or match two or more sets of data points from different coordinate systems. The goal of registration is to find the transformation that best aligns the two sets of data, so they overlap or fit together accurately. In other words, registration algorithms aim to find the optimal translation, rotation, and scaling that aligns one point cloud or 3D model to another. Two popular registration algorithms are:

- **RANdom SAmple Consensus (RANSAC)** - is an iterative algorithm that estimates the parameters of a mathematical model from a set of observed data points containing outliers. It is commonly used for solving the correspondence problem, such as finding the best transformation between two point clouds [2].

Algorithm 1 RANSAC algorithm [1]

```
1. Select randomly the minimum number of points required to determine the
   model parameters.
2. Solve for the parameters of the model.
3. Determine how many points from the set of all points fit with a predefined
   tolerance $\epsilon$.
4. If the fraction of the number of inliers over the total number points in the
   set exceeds a predefined threshold $\tau$ , re-estimate the model parameters
   using all the identified inliers and terminate.
5. Otherwise, repeat steps 1 through 4 (maximum of N times).
```

In the context of this project, RANSAC is applied to identify the points belonging to the surface of the piece accurately. By identifying the inliers (points on the surface), RANSAC enables precise localization of the area to be engraved. The algorithm iteratively selects random point samples, estimates the model parameters, and determines the inliers within a predefined tolerance. It then re-estimates the model using the identified inliers until a specified threshold of inliers is met or a maximum number of iterations is reached.

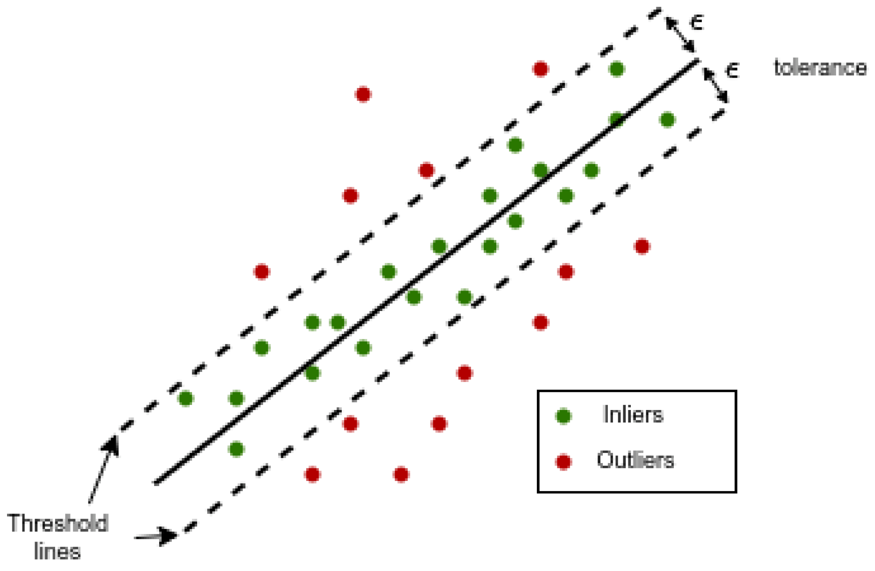


Figure 2.2: RANSAC illustration

- **Iterative Closest Point (ICP)** - is an iterative algorithm commonly employed for point cloud registration, aiming to align two point clouds by minimizing the distance between their corresponding points. It is widely used in scenarios where an initial estimate of the transformation is available. Within the context of this project, the ICP algorithm was considered as a potential means to derive the transformation (translation and rotation) of the pieces as they moved along the conveyor. However, after conducting multiple tests, it became evident that this technique encountered challenges due to the relatively simple geometry of the pieces. This simplicity led to difficulties in accurately computing the transformation matrix, resulting in errors during the registration process. Essentially, the algorithm steps are:

Algorithm 2 ICP algorithm [13]

```
1. For each point (from the whole set of vertices usually referred to as dense or
   a selection of pairs of vertices from each model) in the source point cloud,
   match the closest point in the reference point cloud (or a selected set).
2. Estimate the combination of rotation and translation using a root mean
   square point to point distance metric minimization technique which will
   best align each source point to its match found in the previous step. This
   step may also involve weighting points and rejecting outliers prior to align-
   ment.
3. Transform the source points using the obtained transformation.
4. Iterate (re-associate the points, and so on).
```

2.4 ROS

In this project, a significant portion of the development will take place on Linux to leverage the capabilities of ROS, the Robot Operating System. ROS is an open-source software development kit specifically designed for robotics applications. For this project, ROS version 'noetic ninjemys' will be utilized, which is primarily developed for Ubuntu 20.04 (Focal), making it compatible with the Linux environment. Understanding the key concepts of ROS is crucial for comprehending its role in connecting the various components of the automatic laser marking system. The essential concepts to grasp include [4]:

- **Nodes** - Nodes are processes that perform computation and communicate with other nodes through topics or services. A ROS node is written with the use of a ROS client library, such as roscpp or rospy.
- **Master** - The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
- **Messages** - Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields.
- **Topics** - Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or

subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence

- **Services** - The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply.

- **Bags** - Bags are a format for saving and playing back ROS message data. Bags are an important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.

Moreover some useful ROS tools were used during the development of the project in order to test the results or check the data transfer, these tools are:

- **Rqt** - Rqt is a Qt-based framework for GUI development for ROS. Rqt consists of three following metapackages:

  - **rqt** - Core modules of rqt (ROS GUI) framework.
  - **rqt_common_plugins** - ROS backend tools suite that can be used on/off of robot runtime.
  - **rqt_robot_plugins** - Rqt plugins that are particularly used with robots during their runtime.

  from this the one that was used the most was the rqt_graph from the rqt_common_plugins, which provides a GUI plugin for visualizing the connection of the nodes and their associated topics [9].

- **Rviz** - Rviz, abbreviation for ROS visualization, is a powerful 3D visualization tool for ROS. It allows the user to visualize a lot of information, like robots, the environments they work in, and sensor data. In this project was mainly used to display point clouds. It is a highly configurable tool, with many different types of visualizations and plugins [6].

- **roslaunch** - roslaunch is a tool used to launch multiple ROS nodes both locally and remotely, as well as setting parameters on the ROS parameter server. roslaunch configuration files, which are written using XML can easily automate a complex startup and configuration process into a single command. roslaunch scripts can include other roslaunch scripts, launch nodes on specific machines, and even restart processes which die during execution [5].

2.5 Libraries

This section introduces several libraries that were instrumental in the development of the automatic laser marking system. These libraries served various purposes, ranging from 3D data manipulation to graphical visualization and data processing. Below is a list of key libraries utilized in the project:

- **Open3D** - Open3D is an open-source library designed to facilitate the rapid development of software that involves working with 3D data. This library offers support for three fundamental types of representations: point clouds, meshes, and RGB-D images. For each representation, Open3D provides a comprehensive suite of essential processing algorithms, including functionalities like Input / Output (I/O) operations, sampling, visualization, and data conversion. Moreover, it includes a collection of widely used advanced algorithms, such as normal estimation, ICP registration, and volumetric integration. One notable feature of Open3D is its dual language support, offering implementations in both C++ and Python, making it versatile and accessible for developers. In the context of this thesis, Open3D played a crucial role in tasks involving the reading and saving of point cloud data. [16]
- **Matplotlib** - Matplotlib is a versatile, cross-platform data visualization and graphical plotting library designed for Python and compatible with its numerical extension, NumPy. It serves as a robust open-source alternative to proprietary tools like MATLAB. Matplotlib's flexibility extends to enabling developers to seamlessly integrate plots into graphical user interface (GUI) applications. Matplotlib provides two primary Application Programming Interfaces (APIs) for creating plots:
  - The Pyplot API: This Application Programming Interface (API) builds on a hierarchy of Python code objects, with matplotlib.pyplot as its top-level module. It offers a convenient, stateful interface, resembling the style of MATLAB for those familiar with it.
  - The Object-Oriented (OO) API: This collection of objects allows for assembling and customizing plots with a higher degree of flexibility compared to the pyplot API. It also offers direct access to Matplotlib's backend layers, making it more powerful but potentially more complex to use.

  Notably, Matplotlib's roots trace back to its origins as an open-source alternative to MATLAB, hence the stylistic similarities in the pyplot interface. However, the OO API stands out for its customization capabilities [11] [15].

  For this thesis, Matplotlib played a crucial role in representing point cloud data within the User Interface, making it a fundamental component for visualizing 3D data.
- **PIL (Python Imaging Library)** - PIL is a free and open-source Python library that extends the capabilities of the Python programming language

by providing support for opening, manipulating, and saving various image file formats. It is compatible with Windows, Mac OS X, and Linux.

PIL offers a comprehensive set of standard procedures for image manipulation, including per-pixel operations, masking, transparency handling, image filtering (e.g., blurring, contouring, smoothing, or edge detection), image enhancement (e.g., sharpening, brightness adjustment, contrast adjustment, and color manipulation), text addition to images, and many more.

In this thesis, PIL was employed for image processing tasks within the tkinter environment.

- **OpenCV** - OpenCV (Open Source Computer Vision Library) is an open-source software library that encompasses a vast array of computer vision algorithms. This library follows a modular structure, comprising numerous shared or static libraries. OpenCV offers support for multiple programming languages, including C++, Python, Java, and more, and it is compatible with various operating systems such as Windows, Linux, OS X, Android, and iOS.

  In this thesis, OpenCV was predominantly employed for image processing purposes, as elaborated in Section 3.4.

- **Ezdxf** - Ezdxf is a Python package used for creating new Drawing Exchange Format (DXF) documents and reading, modifying, and writing existing DXF documents. This library is indispensable for this project due to the necessity of using DXF files as templates for laser marking.

2.6 Other Tools and Technologies

In addition to the aforementioned libraries and algorithms, several other tools and technologies were essential for the development of the automatic laser marking system. These tools facilitated the integration of different components, data exchange, and system control. Some of the notable tools and technologies used in this project include:

2.6.1 MySQL Database

MySQL is an open-source relational database management system (RDBMS). A relational database organizes data into one or more data tables in which data may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database [7].

2.6.2 Tkinter

Tkinter is a standard Python library for creating graphical user interfaces. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python. Some definitions [8]:

- **Window** - This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on the user's display screen.
- **Widget** - The generic term for any of the building blocks that make up an application in a graphical user interface. There are four stages to creating a widget:
  - **Create** - create it within a frame
  - **Configure** - change the widgets attributes.
  - **Pack** - pack it into position so it becomes visible. Developers also have the option to use grid or place.
  - **Bind** - bind it to a function or event.

  These are often compressed, and the order can vary.
- **Frame** - In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.
- **Child and parent** - When any widget is created, a parentchild relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

2.6.3 Socket Communication

Socket communication is essentially a mechanism of communication between processes. Its main components are as follows [14]:

1. **Relevant Protocols**:The protocols used for communication mechanism are specified. The general protocols are Transmission Control Protocol / Internet Protocol (TCP/IP) or User Datagram Protocol (UDP).
2. **Local address**: Mainly the address that the Internet assigns to the host under TCP/IP protocol.
3. **Local port number**: Mainly used to distinguish programs related to local operation.
4. **Remote address**: mainly refers to the address assigned to the client host according to TCP/IP protocol.

5. **Remote port number**: It is mainly used to distinguish the program run by the remote host.

The core of Socket communication mechanism is the Socket function. There are nine main functions in the whole Socket communication mechanism. The specific functions are as follows:

1. Socket () function, to create the Socket.
2. Use the Blind () function to specify the local IP address and the corresponding port number
3. Connect () is mainly used to connect clients
4. Accept () is used to connect clients
5. listen (), to listen on the connection.
6. send (), for data transfer.
7. Rec(), for data Receiving.
8. select (), to Query socket status.
9. closesocket () , close the socket.

The whole Socket communication process can be summarized as follows: Open-Read-Write-Close. Under the TCP/IP protocol, the program first opens a network connection, and then the program can close the connection through the continuous read-write operation mentioned above (see also figure 2.3).

2.6.4 Gocator 2490

The Gocator 2490 3D sensor played a pivotal role in acquiring the point cloud data of the pieces on the conveyor. This sensor, manufactured by LMI Technologies, was chosen for its outstanding capabilities and suitability for the project's requirements.
Main features:

- **Laser Line Profiler**: The sensor utilizes a laser line profiler to capture precise 3D data by projecting a laser line onto the target surface and measuring its reflections.
- **High-Speed Scanning**: With its high-speed scanning capabilities, the Gocator 2490 can efficiently capture 3D point cloud data of rapidly moving objects, making it well-suited for conveyor-based applications.

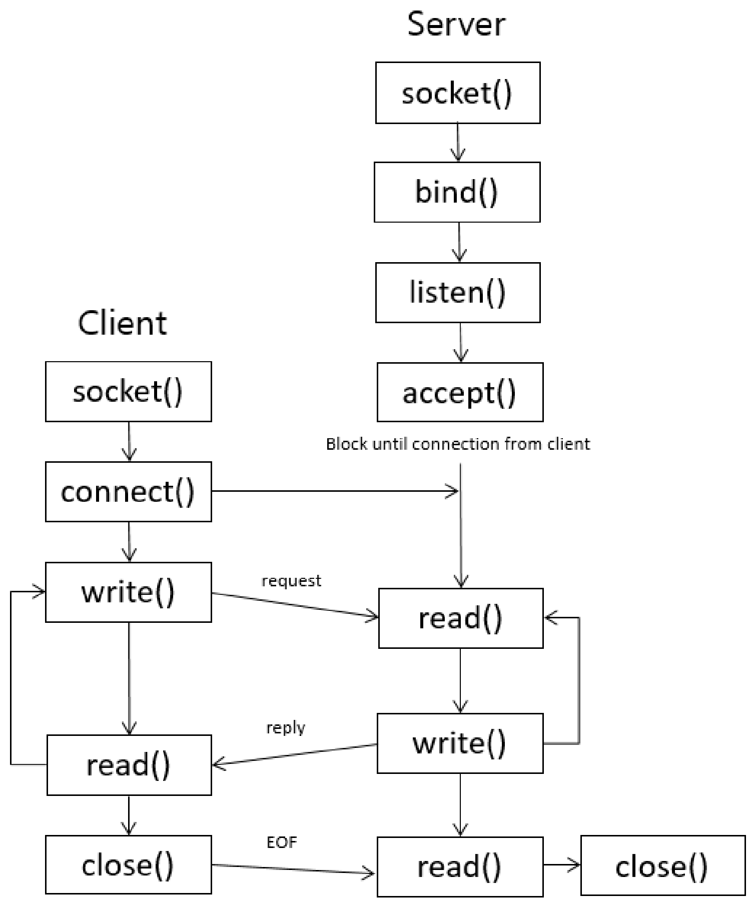


Figure 2.3: Architecture diagram of socket comunication

- **High Resolution**: The sensor offers high-resolution scanning, ensuring accurate and detailed point cloud data of the pieces, essential for reliable object recognition and precise engraving.
- **Robustness and Reliability**: Designed for industrial environments, the Gocator 2490 is built to withstand challenging conditions, ensuring reliable and consistent performance even in harsh manufacturing settings.

To achieve accurate and reliable point cloud data, the Gocator 2490 underwent a precise calibration procedure. The calibration process involved obtaining and setting specific parameters, as described in the sensor's datasheet. The parameters obtained from this procedure are presented in the table 2.1.

Moreover, the sensor requires an alignment procedure to calibrate it with the transport system, allowing for the automatic calculation of transformations (rotations and offsets) applied to the sensor's scan data during target scanning. This calibration process ensures seamless synchronization between the sensor

Figure 2.4: Gocator 2490

Parameter		Value
Scan Mode		Surface
Trigger source		Encoder
Active area	X field of view	473 mm
	Measurement range	max
	X start	-254 mm
	Z start	683 mm
Exposure	Mode	Dynamic
	Min	80
	Max	150
Alignment	Type	Moving
	Target	Disk 100mm
Surface generation		Continuous
Part detection	Frame of reference	Sensor
	Height threshold	1 mm
	Max Part Length	30 cm

Table 2.1: Table of the parameters obtained from the calibration procedure.

and the conveyor's encoder, as specified in the sensor's data sheet. The alignment involves the use of a specific disk moving on the conveyor, a process known as encoder calibration.

3 User Interface Design and Development

The User Interface (UI) plays a crucial role in any automated system, serving as the primary point of interaction between the user and the underlying technology. In the context of the automatic laser marking system project, the user interface serves as a bridge connecting operators or technicians with the various components and functionalities of the system. This chapter focuses on the design, development, and implementation of the user interface within the automatic laser marking system.

3.1 Framework

In the realm of software development, a framework refers to a structured environment or platform that provides a foundation for building applications. It consists of pre-defined libraries, tools, and conventions that facilitate the development process and allow developers to focus on application-specific functionality. The chosen framework for the user interface development of the automatic laser marking system is Tkinter (Tk).

Tkinter is a widely adopted Graphical User Interface (GUI) framework for Python. It provides a set of Python bindings for the Tk GUI toolkit, which originated from the Tool Command Language (TCL) scripting language. Tkinter offers several advantages that align with the requirements of the automatic laser marking system, including:

- **Integration with Python:** Tkinter seamlessly integrates with Python, the chosen programming language for the user interface. This integration allows for efficient development, leveraging the extensive capabilities and libraries available in the Python ecosystem.
- **Cross-platform Compatibility:** Tkinter ensures that the user interface can run on multiple operating systems, including Windows, macOS, and Linux. This cross-platform compatibility is crucial as it enables the automatic laser marking system to be deployed across different environments without extensive modifications.
- **Ease of Use and Customizability:** Tkinter provides a high-level API that simplifies the development of user interfaces. It offers a wide range of customizable widgets, allowing developers to create visually appealing and intuitive interfaces. Tkinter's simplicity and flexibility enable the customization of the user interface to meet the specific needs of the automatic laser marking system.
- **Stability and Community Support:** Tkinter has been widely adopted and has a robust codebase, making it stable and reliable. Additionally, it benefits from a large and supportive community of developers, providing extensive documentation, tutorials, and resources to assist in the development process.

While Tkinter was the chosen framework for the automatic laser marking system's user interface, it is important to note that there are other viable options available. Some notable alternatives to Tkinter include PyQt, wxPython, Kivy, PySide, and PyGTK, each with their own unique features, strengths, and community support. The selection of Tkinter was based on its alignment with the project's requirements, the integration with Python, its cross-platform compatibility, ease of use, customizability, and stability.

In the next sections of this chapter, we will explore the design and development process of the user interface, showcasing how Tkinter was leveraged to create an intuitive and effective interface for the automatic laser marking system.

3.2 Program structure

The user interface (UI) of the automatic laser marking system is designed leveraging the power of Python classes and Tkinter frames. This section provides an overview of the main structure of the program, highlighting the use of classes and the creation of different Tkinter frames for each "page" or section of the interface.

3.2.1 Python classes

To ensure modularity and maintainability, the UI is implemented using Python classes. The main application class acts as the central point of the UI program, managing the overall flow, interaction between frames, and system-level functionalities. It initializes the UI, handles events, and coordinates the exchange of data between different components.

3.2.2 Tkinter frames

The UI program utilizes Tkinter frames to organize the interface into distinct sections or "pages" that correspond to different tasks and functionalities. Each frame represents a specific aspect of the marking process or user interaction. For example, there may be separate frames for piece identification, insert new pieces in the system, look the pieces already present in the system, check the marking history. Each frame contains the necessary widgets, buttons, labels, and input fields relevant to its specific functionality. The use of frames allows for a logical separation of tasks, making the UI more intuitive and user-friendly. It enables operators or technicians to navigate through different sections of the interface seamlessly, focusing on the specific task at hand while keeping the overall system context intact. Additionally, frames can be dynamically loaded or hidden based on the current stage of the marking process or user input, providing a streamlined and guided user experience. A general scheme of the used frames in figure 3.1.

By utilizing Python classes and Tkinter frames, the user interface is designed with a modular and organized structure. This approach enhances maintainability, extensibility, and code reusability while providing a well-organized and intuitive user experience. In the subsequent subsections, we will explore the specific tasks and functionalities of the user interface, examining how they are implemented within their respective frames and contribute to the overall system control and interaction.

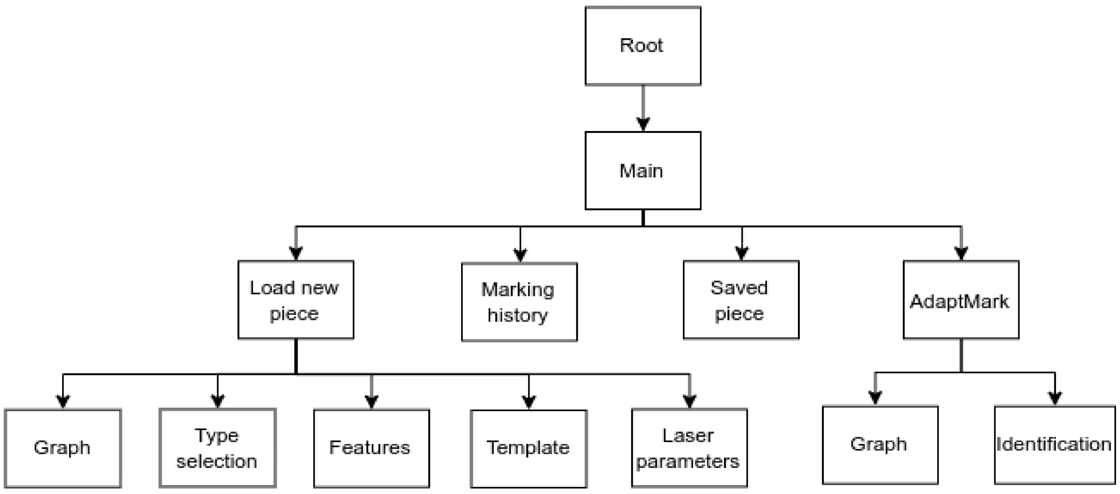


Figure 3.1: User interface: Frame scheme

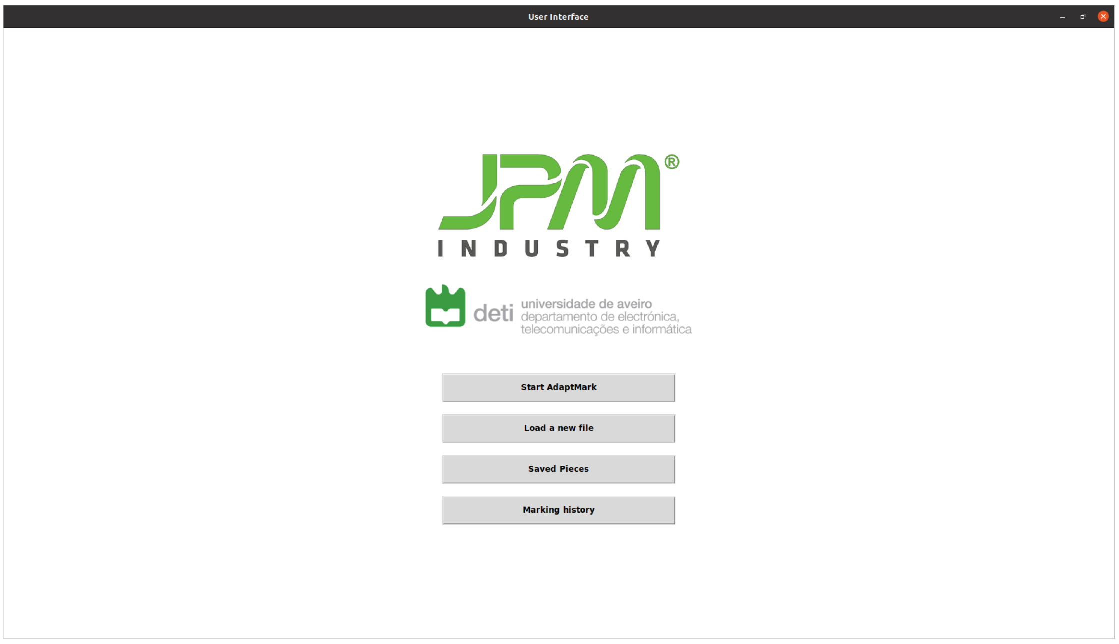

User Interface
INDUSTRY
deti universidade de aveiro departamento de electrónica, telecomunicações e informática
Start AdaptMark
Load a new file
Saved Pieces
Marking history

Figure 3.2: User interface: Main page

3.3 Tasks and Functionalities

In this section, the tasks and functionalities performed by the user interface are discussed. Each subsection focuses on a specific aspect or job of the user interface, ensuring a comprehensive understanding of its capabilities.

3.3.1 Load a new file

A key function of the user interface is to facilitate the initialization of the automatic laser marking system with new pieces, capturing their features and associating them with the appropriate template position and orientation. This process allows operators to integrate new pieces into the system and specify important details such as physical dimensions of the piece (figure 3.4), template positioning on the piece (figure 3.5), and laser parameters for the engraving of the template (figure 3.6).

To streamline the process, the user interface categorizes the pieces into five distinct categories based on their features. These categories include circular or non-circular, with the presence of holes and/or notches (see figure 3.9), moreover as initial test of the identification module only some pieces were used and this were identified by a number from 1 to 18 (figure 3.10). By classifying the pieces into these categories, operators can easily navigate through the initialization process, selecting the appropriate category that aligns with the piece's specific characteristics.

The user interface provides operators with the capability to scan the new piece, capturing its unique identifiers and storing them in the system's database.

During this process, the user interface employs a specific layout. The left part of the monitor displays the graph frame, utilizing the matplotlib library to visualize the acquired point cloud of the piece. Operators can interact with the graph frame to assess the piece visually and select the appropriate category based on the categorized features.

On the right side of the monitor, the user interface presents a series of frames to guide operators through the procedure of loading a new piece. The first frame is the selection frame, where operators can choose the piece type based on the predefined categorization (Figure 3.10). Subsequently, operators move through the feature selection frame (Figure 3.4), the template positioning frame (Figure 3.5), and the laser parameters frame (Figure 3.6). Each frame guides the operator in providing the necessary information, such as piece features, template positioning, and laser parameters, to ensure accurate and effective engraving.

By adopting this structured approach within the user interface, operators are systematically guided through the process of loading a new piece into the automatic laser marking system.

The feature selection frame is dependent on the piece type selected pre-

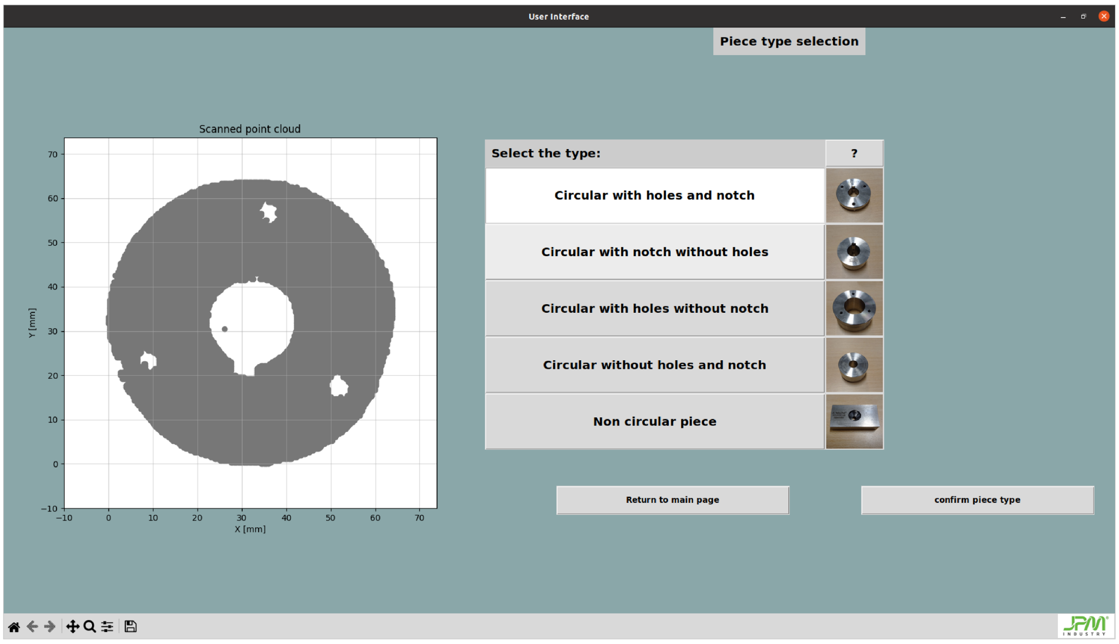


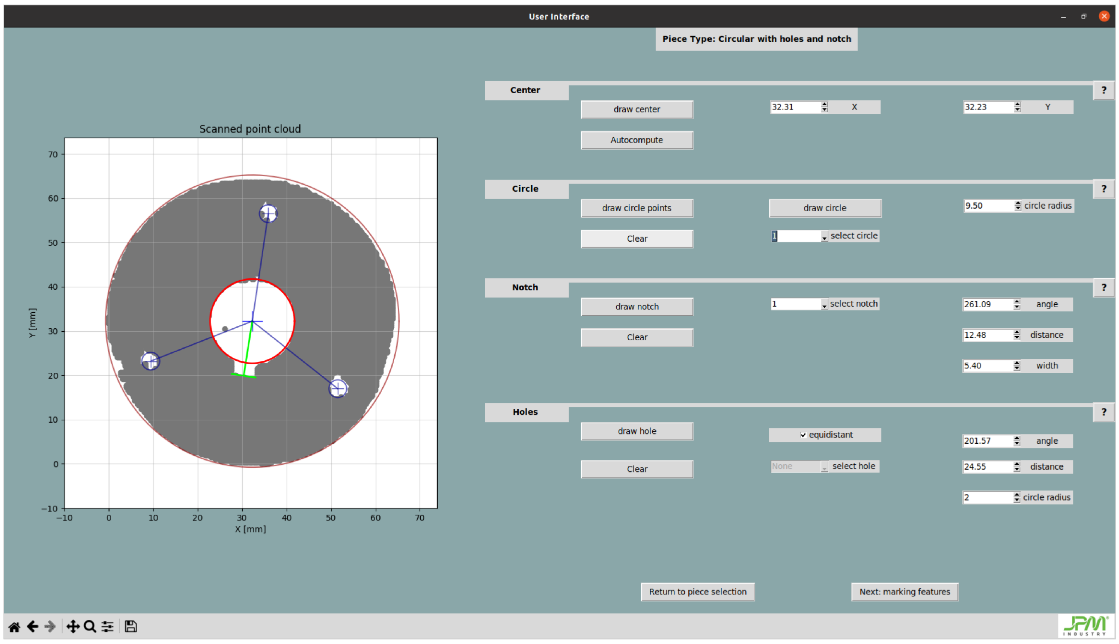
Figure 3.3: User interface: category selection frame

viously. It dynamically displays the necessary features based on the selected category. For example, if a non-circular piece is selected, different features such as vertex selection will be displayed (Figure 3.7).

After selecting the features, the marking position frame is presented. Here, the point cloud is translated to the origin based on the identified center of the piece. Additionally, the piece may be rotated depending on the "main feature" that was defined (Figure 3.8).

Moreover, for a more accurate placement of the template over the point cloud, the selected template is processed and displayed on the designated point. This process involves utilizing image processing operations, as described in Section 3.4, to enhance the template visualization and alignment with the point cloud.

The final frame within the user interface is dedicated to configuring the laser parameters. Operators can specify the desired laser settings for engraving, such as velocity, frequency, intensity, but also the material of the piece and the minimum quality of engraving. Once the parameters are set, the user interface saves the relevant data in the system's database as described in Section 3.5.



Piece Type: Circular with holes and notch

Center: 32.31 X; 32.23 Y

Circle: 9.50 circle radius; 1 select circle

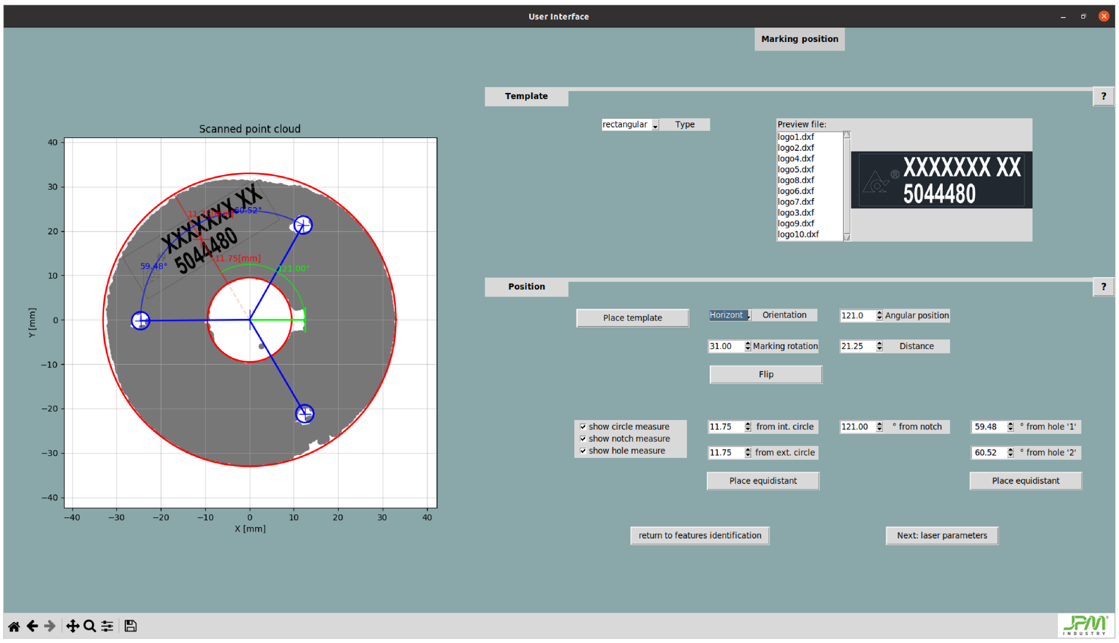
Notch: 1 select notch; 261.09 angle; 12.48 distance; 5.40 width

Holes: ✓ equidistant; None select hole; 201.57 angle; 24.55 distance; 2 circle radius

Figure 3.4: User interface: Features identification

3.3.2 Saved Pieces

The user interface provides operators with the ability to manage and view all the pieces that have been previously saved in the system's database. The "Saved Pieces" functionality allows operators to access a comprehensive list of stored pieces and perform various actions such as viewing piece features, modifying piece details, or deleting pieces if necessary (figure 3.11). Operators have the flexibility to modify saved pieces when necessary. The user interface allows for editing piece attributes, enabling operators to update dimensions, adjust template positions, or modify laser parameters based on new requirements or changes in production specifications. This feature enhances the system's adaptability and allows for efficient management of saved pieces. Furthermore, operators can also remove pieces from the database through the user interface. This option provides a streamlined approach to delete unnecessary or outdated pieces, ensuring the database remains organized and up to date.



Type: rectangular

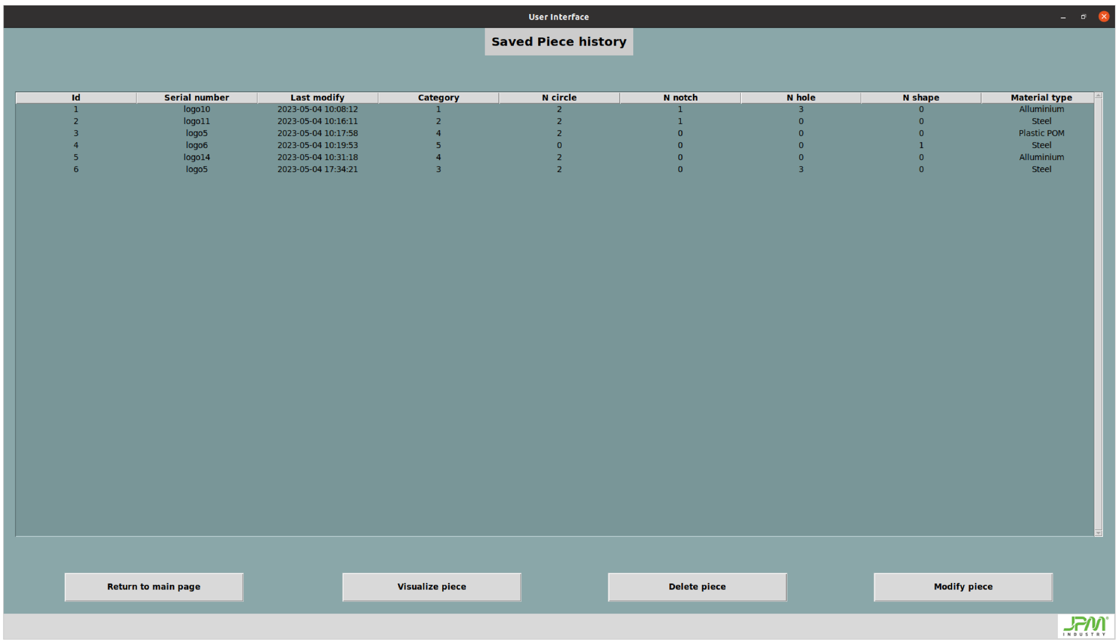
Preview file:
logo1.dxf
logo2.dxf
logo4.dxf
logo5.dxf
logo8.dxf
logo6.dxf
logo7.dxf
logo3.dxf
logo9.dxf
logo10.dxf

Orientation: Horizont
Angular position: 121.0
Marking rotation: 31.00
Distance: 21.25

☑ show circle measure
☑ show notch measure
☑ show hole measure

11.75 from int. circle
11.75 from ext. circle
121.00 ° from notch
59.48 ° from hole '1'
60.52 ° from hole '2'

Figure 3.5: User interface: Template position

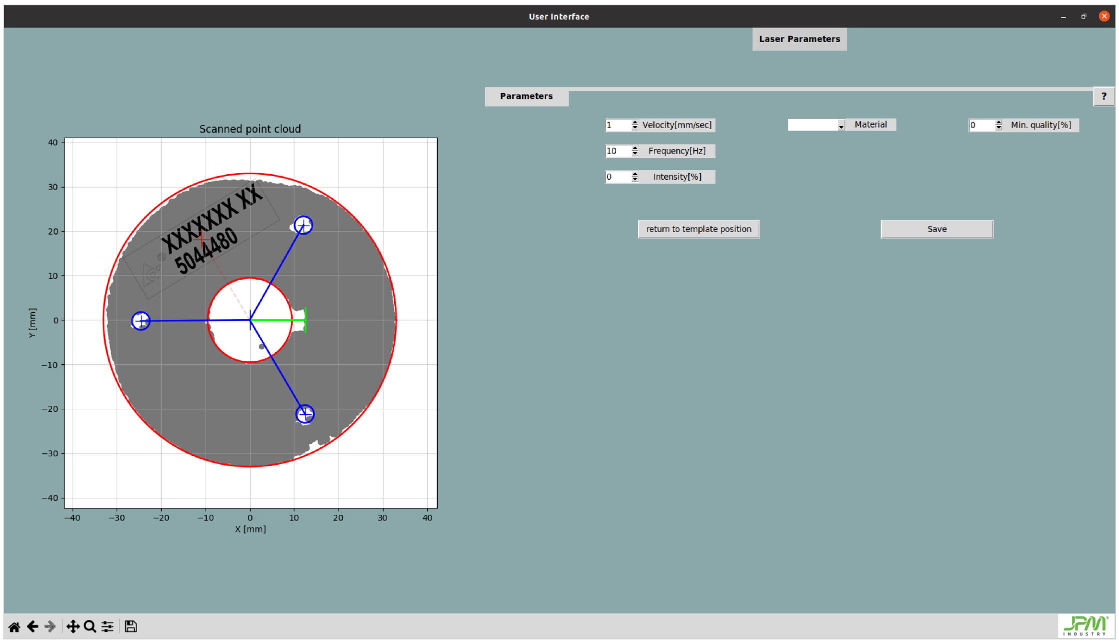


Id	Serial number	Last modify	Category	N circle	N notch	N hole	N shape	Material type
1	logo10	2023-05-04 10:08:12	1	2	1	3	0	Alluminium
2	logo11	2023-05-04 10:16:11	2	2	1	0	0	Steel
3	logo5	2023-05-04 10:17:58	4	2	0	0	0	Plastic POM
4	logo6	2023-05-04 10:19:53	5	0	0	0	1	Steel
5	logo14	2023-05-04 10:31:18	4	2	0	0	0	Alluminium
6	logo5	2023-05-04 17:34:21	3	2	0	3	0	Steel

Figure 3.11: User interface: Saved piece

3.3.3 Marking History

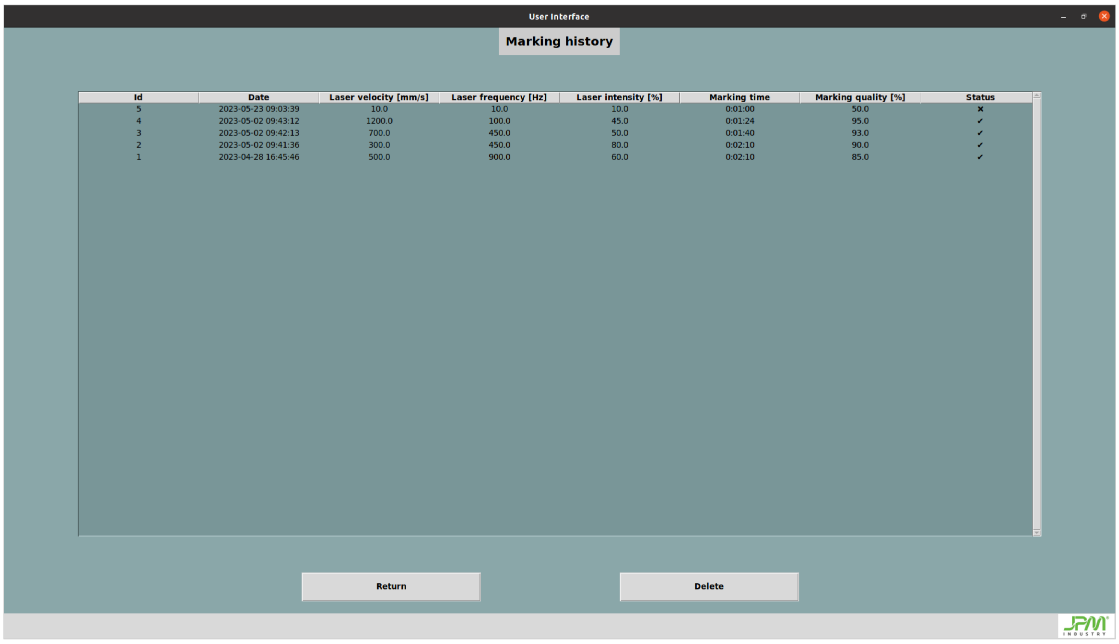
The user interface includes a "Marking History" feature that provides operators with a comprehensive overview of all the markings performed by the



Velocity[mm/sec]	1
Frequency[Hz]	10
Intensity[%]	0
Material	
Min. quality[%]	0

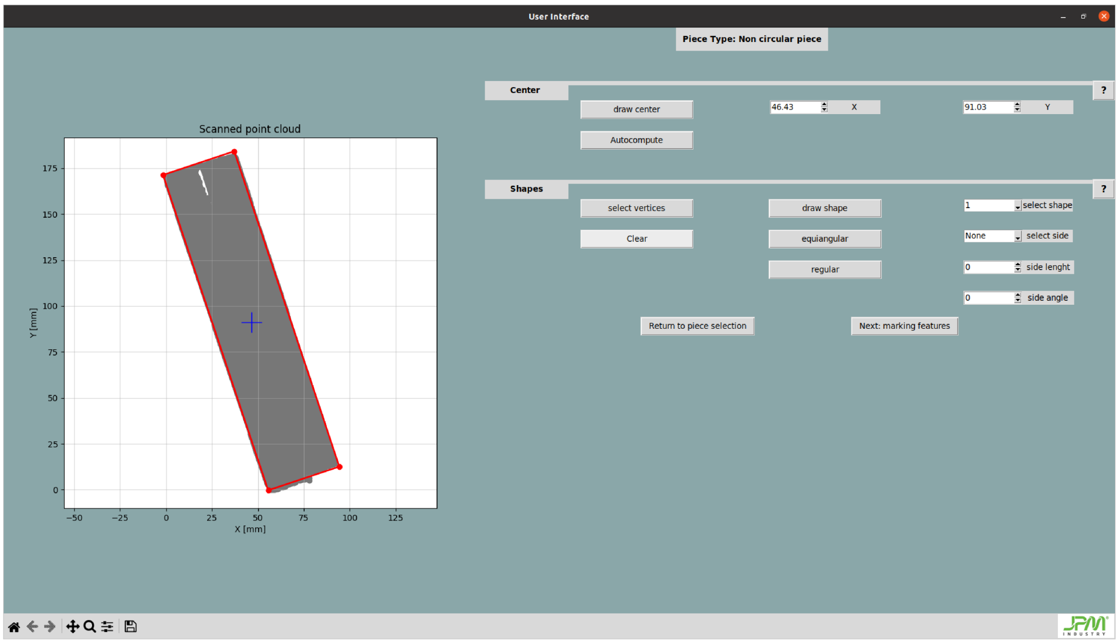
Figure 3.6: User interface: Laser parameters

system. This functionality allows operators to review the history of completed markings, including associated laser parameters and marking quality (figure 3.12). Operators can access the "Marking History" section to examine the details of previous markings, such as date, time, piece identification, and related laser parameters.



Id	Date	Laser velocity [mm/s]	Laser frequency [Hz]	Laser intensity [%]	Marking time	Marking quality [%]	Status
5	2023-05-23 09:03:39	10.0	10.0	10.0	0:01:00	50.0	✗
4	2023-05-02 09:43:12	1200.0	100.0	45.0	0:01:24	95.0	✓
3	2023-05-02 09:42:13	700.0	450.0	50.0	0:01:40	93.0	✓
2	2023-05-02 09:41:36	300.0	450.0	80.0	0:02:10	90.0	✓
1	2023-04-28 16:45:46	500.0	900.0	60.0	0:02:10	85.0	✓

Figure 3.12: User interface: Marking history



Piece Type: Non circular piece

46.43 X

91.03 Y

1 select shape

None select side

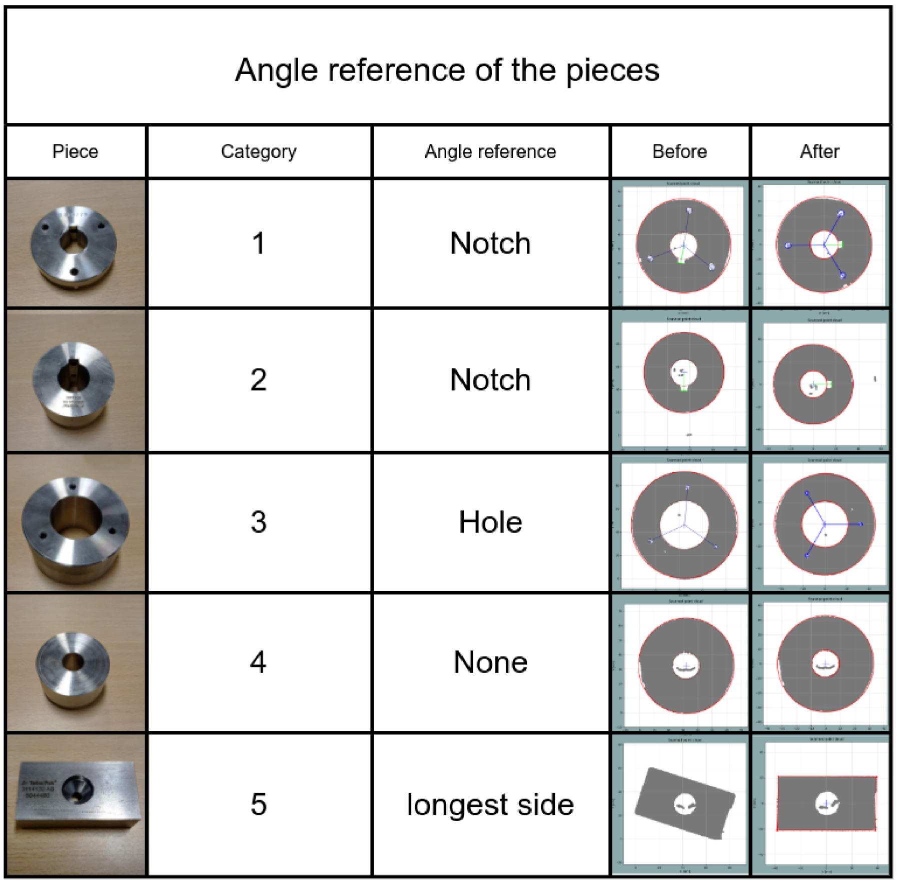
0 side lenght

0 side angle

Figure 3.7: User interface: Not circular piece

3.3.4 Start Adaptmark

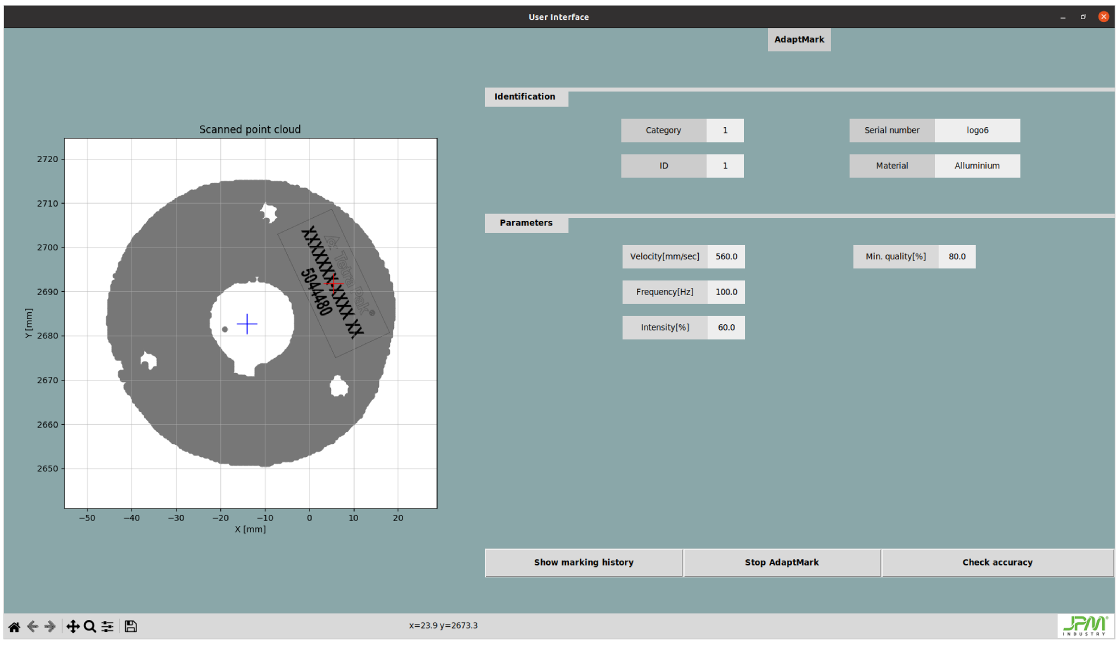
The "Start Adaptmark" frame provides operators with a comprehensive overview of the marking process. By visualizing the acquired point cloud, operators can verify the accurate identification and positioning of the piece on the conveyor. Within this frame, operators can access real-time visualization of the acquired point cloud and the associated parameters for the identified piece. This information includes details such as the material type, the corresponding template, and the laser parameters required for engraving (see Figure 3.13).



Angle reference of the pieces

Piece	Category	Angle reference	Before	After
	1	Notch		
	2	Notch		
	3	Hole		
	4	None		
	5	longest side		

Figure 3.8: Rotation of the pieces based on the category



AdaptMark

Identification

Category	1	Serial number	logo6
ID	1	Material	Alluminium

Parameters

Velocity[mm/sec]	560.0	Min. quality[%]	80.0
Frequency[Hz]	100.0		
Intensity[%]	60.0		

Figure 3.13: User interface: Start AdaptMark

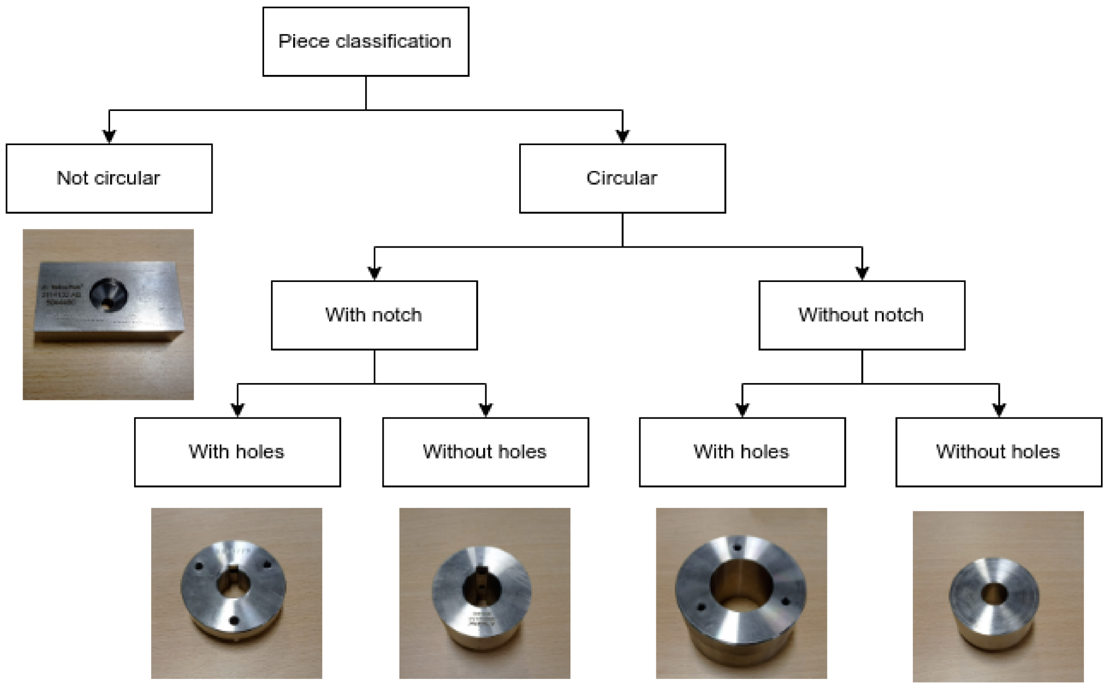


Figure 3.9: Piece classification

3.4 Image processing

Image processing plays an important role in the user interface by enabling template choice and positioning. It involves the manipulation and analysis of images to facilitate operations within the interface. Image processing techniques are employed to convert templates, provided in DXF format, into a suitable format for processing within the user interface. Image processing can be defined as the application of various algorithms and techniques to digital images in order to extract information, enhance visual quality, or perform specific operations. In the context of the automatic laser marking system, image processing is utilized to convert templates from DXF format to a more compatible format, such as Portable Network Graphics (PNG). The conversion process involves translating the vector-based DXF files into raster-based images that can be easily manipulated within the user interface. This conversion allows for efficient processing and placement of the template on the pieces. Additionally, image processing techniques are applied to perform specific operations on the converted image. These operations may include removing the image border and eliminating background elements, such as white space, to ensure accurate template placement and maximize the marking area. In summary, image processing in the user interface involves converting DXF templates into a compatible format and applying operations to optimize their placement on the pieces. In the figure 3.14 is possible to see an example of image processing applied to one template.

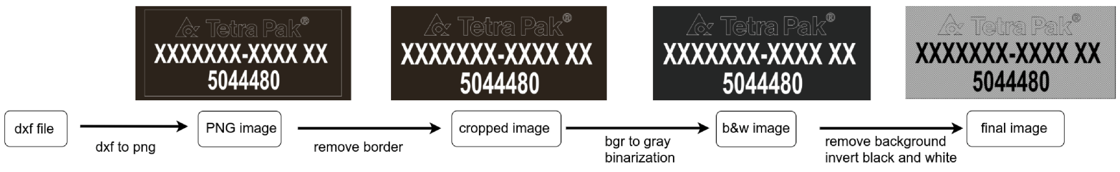


Figure 3.14: Image processing

3.5 Database

The automatic laser marking system utilizes a MySQL database to store and manage all the relevant data related to the pieces in a comprehensive and intelligent manner. The database serves as a central repository for storing piece information, laser parameters, marking history, and other relevant data. Initially, a concept for the table structure was developed (refer to Figure 3.15), providing an initial framework for the database design. However, to meet specific requirements and accommodate system needs, some modifications were made, resulting in the final creation of database tables (refer to Figure 3.16). To ensure efficient data management and retrieval, each table within the database has a designated primary key. The primary key uniquely identifies each record in the table, providing a reliable means of reference. In this case, the primary key field is the ID, which is assigned a unique numerical value for each piece. The ID starts from one, and subsequent pieces are assigned incrementing values. Additionally, the 'serial number' field in each table corresponds to the name of the associated DXF file, which is also the text to be engraved on the piece. To accomplish the database integration, the user interface utilizes the MySQL Connector, a Python library that provides an interface for connecting to and interacting with MySQL databases. This integration allows for the seamless storage and retrieval of piece-related data, laser parameters, and marking history within the database. An example of the code used to retrieve data from the database is shown in Code 3.1. This code demonstrates how the user interface can establish a connection to the MySQL database and execute SQL queries to retrieve specific information from the database tables.

```
# Establish a connection to the MySQL database
```

```python
try:
    mydb = mysql.connector.connect( host="localhost",
                                    user=self.mysql_user,
                                    password=self.mysql_psw,
                                    database="AdaptMark"
    )
    # check if the connection is OK
    if not mydb.is_connected():
        tk.messagebox.showerror(title='Error', message="Database 
    connection failed, returning to the main page")
        self.clickExitButton()

    # create a cursor to execute SQL queries
    cursor = mydb.cursor()

    # operation in sql language
    sql = 'select * from Piece_Category where id = %s;'
    # execute the query
    cursor.execute(sql,(id,))
    records = cursor.fetchall()[0]

except mysql.connector.Error as e:
    msg = "Error reading data from MySQL table:\n"+str(e) + "\
    nreturning to the main page"
    tk.messagebox.showerror(title='Error', message=msg)
    self.clickExitButton()
    return
```

Code 3.1: Example of use of Mysql connector

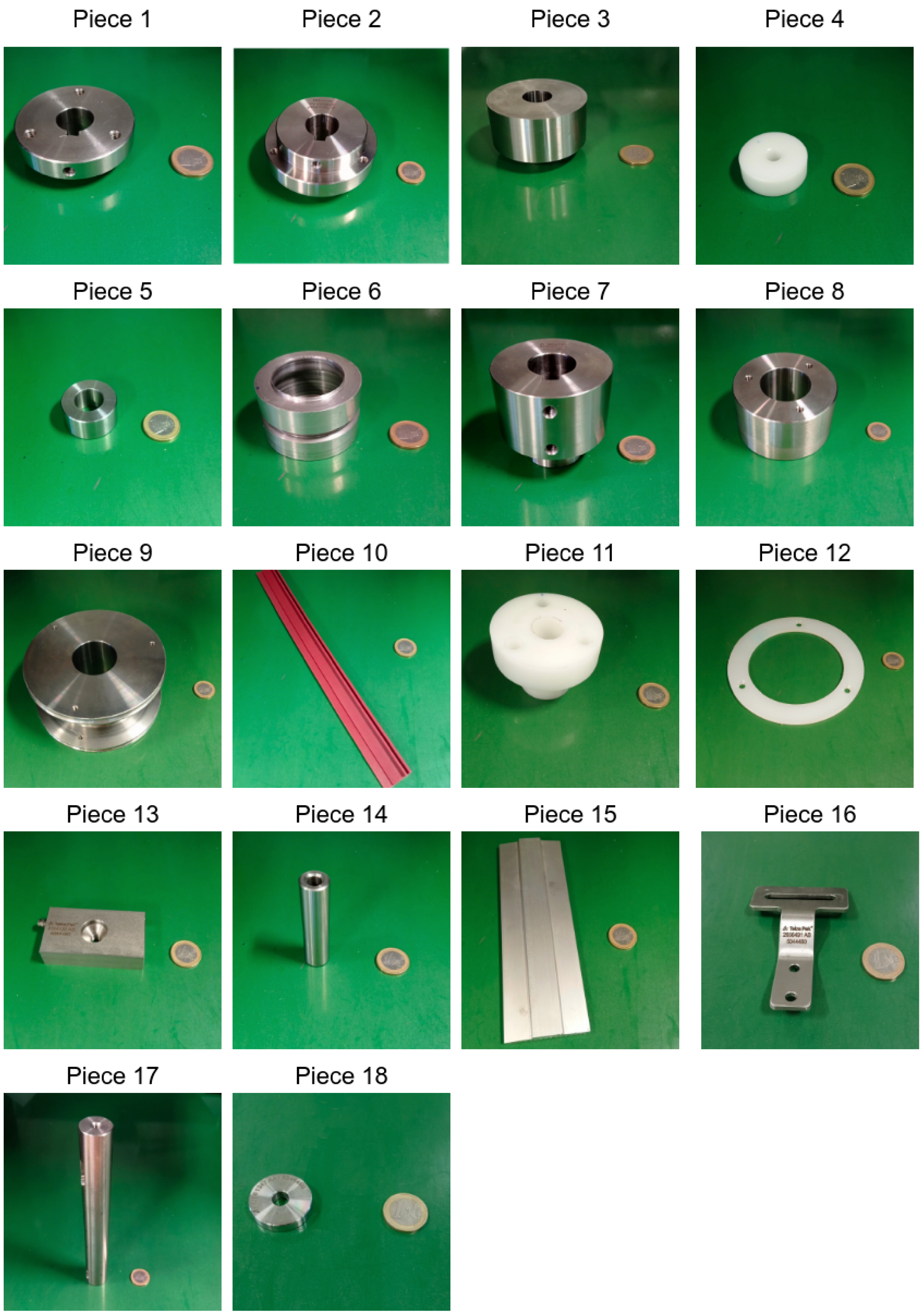


Figure 3.10: Pieces used during the identification test

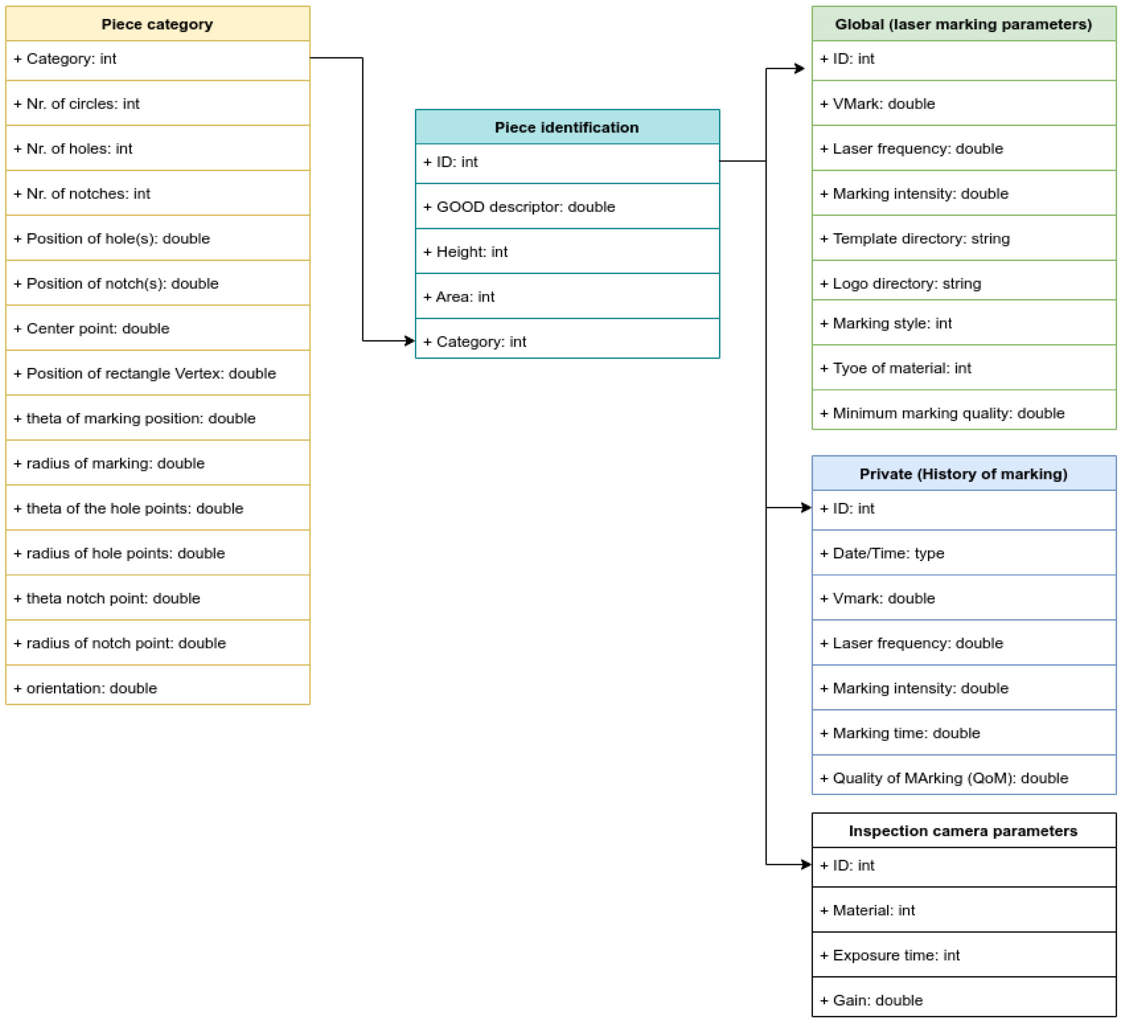
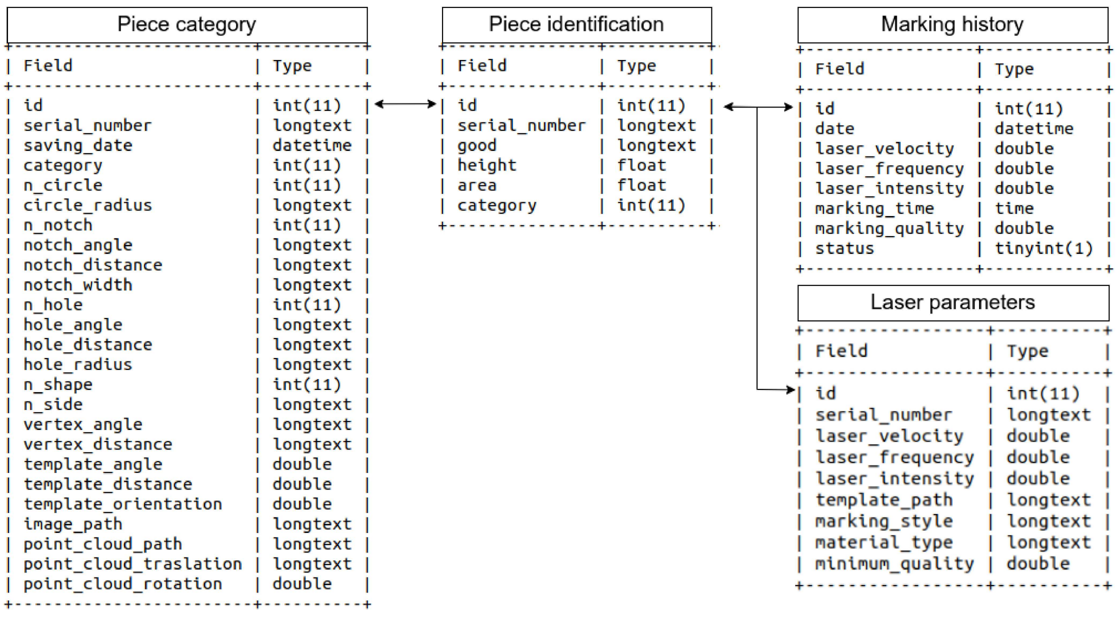

Piece category
+ Category: int
+ Nr. of circles: int
+ Nr. of holes: int
+ Nr. of notches: int
+ Position of hole(s): double
+ Position of notch(s): double
+ Center point: double
+ Position of rectangle Vertex: double
+ theta of marking position: double
+ radius of marking: double
+ theta of the hole points: double
+ radius of hole points: double
+ theta notch point: double
+ radius of notch point: double
+ orientation: double
Piece identification
+ ID: int
+ GOOD descriptor: double
+ Height: int
+ Area: int
+ Category: int
Global (laser marking parameters)
+ ID: int
+ VMark: double
+ Laser frequency: double
+ Marking intensity: double
+ Template directory: string
+ Logo directory: string
+ Marking style: int
+ Tyoe of material: int
+ Minimum marking quality: double
Private (History of marking)
+ ID: int
+ Date/Time: type
+ Vmark: double
+ Laser frequency: double
+ Marking intensity: double
+ Marking time: double
+ Quality of MArking (QoM): double
Inspection camera parameters
+ ID: int
+ Material: int
+ Exposure time: int
+ Gain: double

Figure 3.15: Original database tables



Piece category

Field	Type
id	int(11)
serial_number	longtext
saving_date	datetime
category	int(11)
n_circle	int(11)
circle_radius	longtext
n_notch	int(11)
notch_angle	longtext
notch_distance	longtext
notch_width	longtext
n_hole	int(11)
hole_angle	longtext
hole_distance	longtext
hole_radius	longtext
n_shape	int(11)
n_side	longtext
vertex_angle	longtext
vertex_distance	longtext
template_angle	double
template_distance	double
template_orientation	double
image_path	longtext
point_cloud_path	longtext
point_cloud_traslation	longtext
point_cloud_rotation	double

Piece identification

Field	Type
id	int(11)
serial_number	longtext
good	longtext
height	float
area	float
category	int(11)

Marking history

Field	Type
id	int(11)
date	datetime
laser_velocity	double
laser_frequency	double
laser_intensity	double
marking_time	time
marking_quality	double
status	tinyint(1)

Laser parameters

Field	Type
id	int(11)
serial_number	longtext
laser_velocity	double
laser_frequency	double
laser_intensity	double
template_path	longtext
marking_style	longtext
material_type	longtext
minimum_quality	double

Figure 3.16: Final Mysql tables

4 System Connection

The automatic laser marking system comprises multiple components and programs that were developed by different parties and designed to work on different operating systems. To establish a seamless connection and ensure the system's overall functionality, a strategic approach was adopted. This section provides an overview of the system connection strategy, addressing the challenges and solutions implemented to achieve integration. To address the diverse operating system requirements, the decision was made to utilize a Windows environment with a Linux virtual machine. This setup enabled the integration of various components that were developed to work on different platforms. The Linux virtual machine was chosen to leverage the capabilities of Robot Operating System (ROS), a widely adopted framework for robot control and communication. One of the challenges encountered during the integration process was the sensor connection. Initially, the sensor communication was tested and implemented on Linux. However, within the virtual machine environment, direct communication with the sensor was not possible. As a result, it was necessary to relocate the program responsible for acquiring point cloud data from the sensor to the Windows environment. To enable communication between Windows and Linux, a socket-based communication method was employed. Messages containing the acquired point cloud data were sent from Windows to Linux, facilitating the seamless exchange of information between the two environments (see figure 4.1).

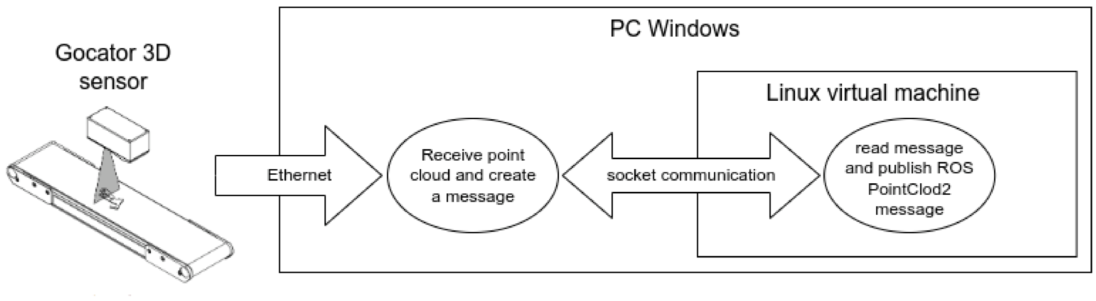

Gocator 3D sensor
PC Windows
Linux virtual machine
Ethernet
Receive point cloud and create a message
socket communication
read message and publish ROS PointClod2 message

Figure 4.1: System scheme

4.1 ROS INTEGRATION

To facilitate the integration of various existing programs within the system, the AdaptMark project utilized Robot Operating System (ROS) inside a Linux virtual machine. ROS is a flexible framework for writing robot software that provides a collection of tools, libraries, and conventions for simplifying the development of complex robotic systems.

Within the ROS environment, several nodes were created to enable communication and data exchange between different components of the system:

- **gocator node socket**: This node communicates with a program in the Windows environment. The program receives the point cloud from the 3D sensor, creates a message, and sends it through socket communication. The ROS node receives the point cloud data from the socket, decodes the message, and publishes it as a PointCloud2 ROS message, which contains the points of the point cloud.

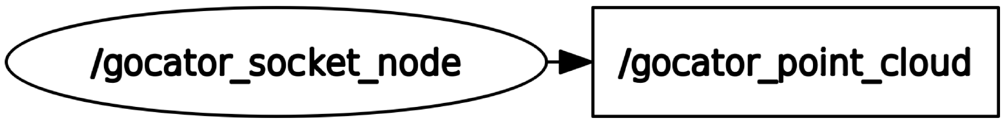


Figure 4.2: gocator node socket: connection scheme

- **ransac node**: The ransac node applies the RANdom SAmple Consensus (RANSAC) method to the point cloud data in order to extract the plane of interest on the piece where the engraving will take place. The resulting point cloud is sent in a PointCloud2 ROS message for further processing.

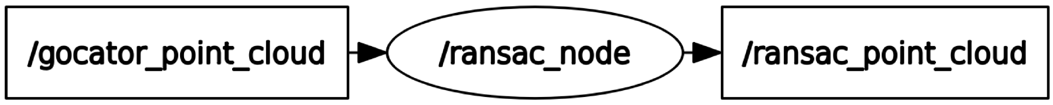


Figure 4.3: ransac node: connection scheme

- **good Node**: This node receives the original point cloud and extracts the Global Orthographic Object Descriptor (GOOD) from it [12]. The resulting descriptors are published as a Float32MultiArray ROS message. These descriptors are utilized by the Identification node for piece identification, as well as by the User Interface for storing the descriptors in the database when adding a new piece.

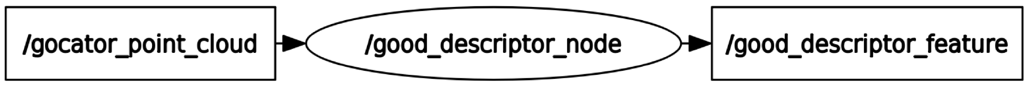


Figure 4.4: good node: connection scheme

- **ha node**: The Height and Area (ha) node computes the height and area of the surface designated for engraving, which was extracted using RANSAC. The computed height and area values are published as a String message, formatted as a single string.

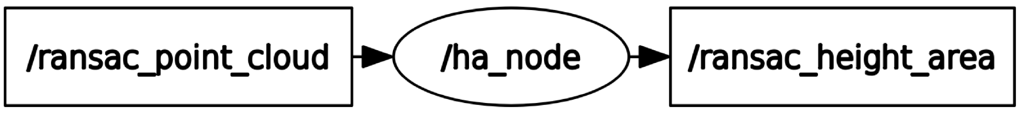


Figure 4.5: ha node: connection scheme

- **identifier node**: The Identifier node is responsible for identifying the pieces on the conveyor. It uses three features for identification: height, area, and good descriptors [12]. After identification, the piece ID is obtained, along with the piece category. Based on the category, the program attempts to identify additional features of the piece (e.g., circles, holes, notches) to determine the orientation on the conveyor and the position of the piece's center. The Identifier node publishes a Float32MultiArray ROS message containing the piece ID, position of the center, and position of the engraving center point:

$$(\mathrm{ID}, x_{\text{center}}, y_{\text{center}}, x_{\text{engraving}}, y_{\text{engraving}}). \tag{4.1}$$

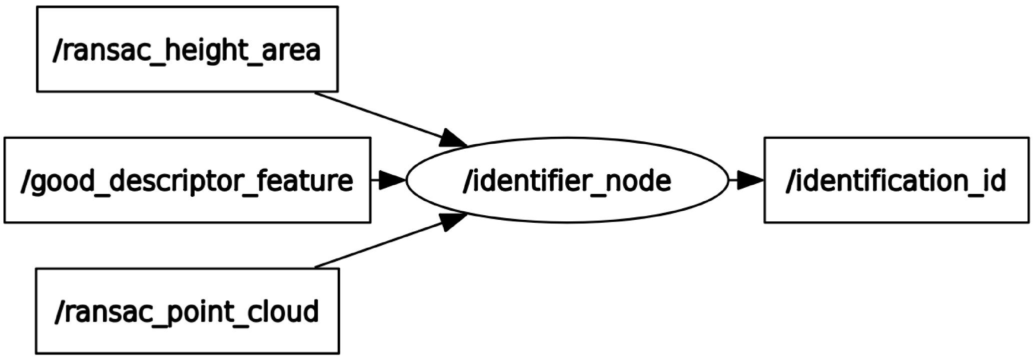


Figure 4.6: identifier node: connection scheme

- **gui node**: The GUI (Graphical User Interface) node is responsible for the user interface of the system. This node has subscribers to four important topics: the RANSAC point cloud, height and area data, good descriptors [12], and the identification ID. If the user interface is being used to insert a new piece into the database, the received data is utilized to store the piece information in the database. In this case, the identification ID is not used. On the other hand, if the system is being used for engraving pieces during normal operation, the identification ID message is used to retrieve the information of the identified piece from the database and determine the position for engraving.

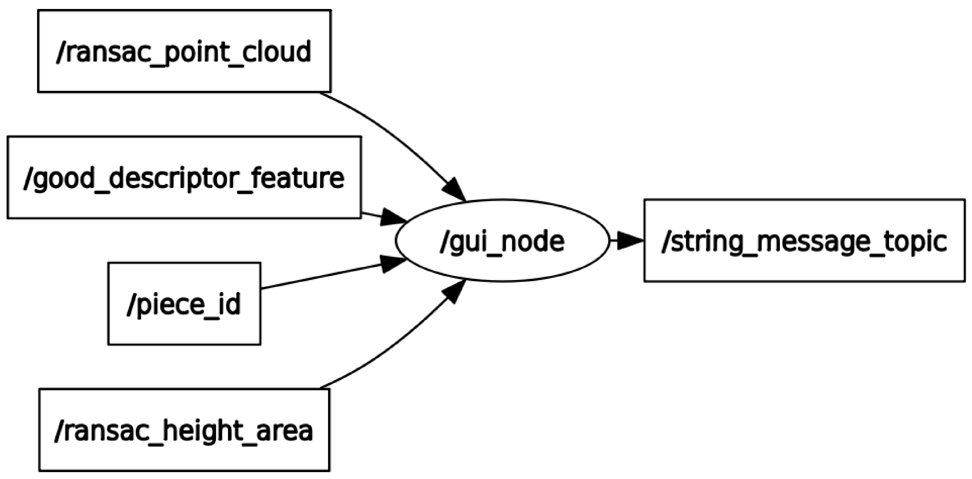


Figure 4.7: gui node: connection scheme

Additionally, a launch file was developed to streamline the initialization of the entire system, allowing for the straightforward startup of all system nodes with a single command. The resultant interconnection between nodes is visually depicted in Figure 4.8 for reference.

By leveraging ROS and implementing these nodes, the AdaptMark project successfully achieved the integration and communication among different components of the system, enabling seamless data exchange and coordination between the user interface, piece identification, point cloud processing, and engraving tasks.

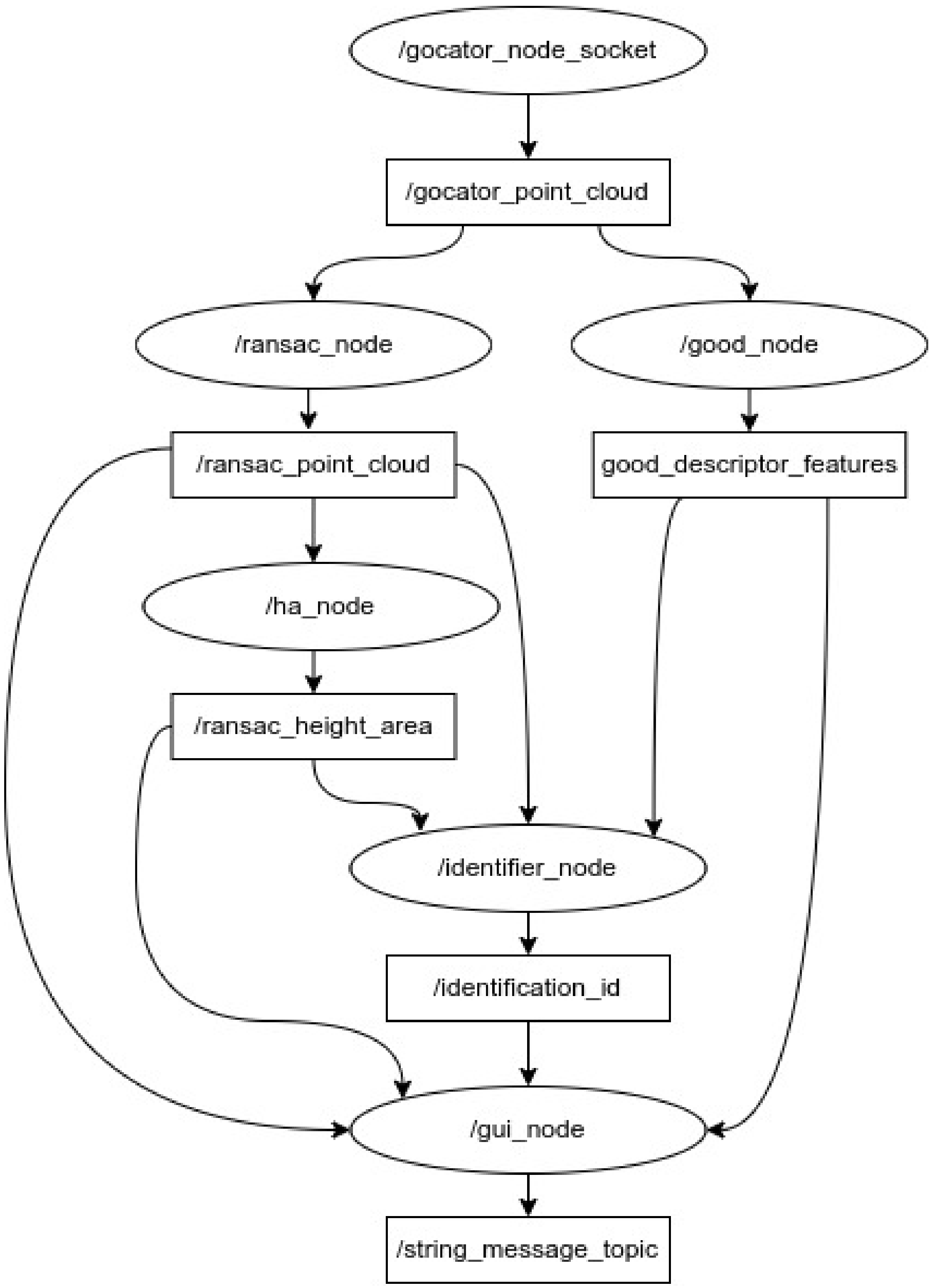


Figure 4.8: ROS connection scheme

4.2 Socket Communication

In the AdaptMark project, socket communication was employed to enable seamless communication between the Windows environment and the Linux virtual machine. Socket communication is a network communication method that allows for data exchange between different processes running on different machines or within the same machine.

The need for socket communication arose due to the diverse environments in which the project was developed. The main environment for the system was Windows, while the Linux virtual machine was utilized for ROS integration and certain components of the system.

Socket communication facilitated the exchange of critical information, such as the acquired point cloud data from the 3D sensor, between the Windows and Linux environments. The communication flow involved a program running in the Windows operating system receiving the point cloud data, packaging it into a message, and transmitting it through socket communication. On the other end, a ROS node within the Linux virtual machine received the message, decoded it, and processed the point cloud data for further tasks, such as piece identification and engraving. The subsequent subsections will delve into the implementation details of the socket server and the socket client, exploring how they collaborate to ensure reliable communication and integration between the two environments.

4.2.1 Socket client

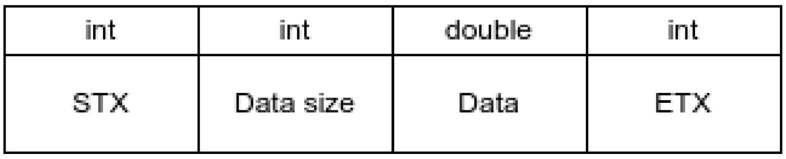
The socket client in our case is the program in the widows environment and is written in C++. The socket message format, as shown in Figure 4.9, was designed to package the point cloud data efficiently. The message format includes a Start of Message (STX) marker to identify the beginning of a new message, followed by the size of the data. It's important to note that the size represents the number of valid points within the point cloud. In accordance with the definition of the point structure (see code 4.1), denominated as 'ProfilePoint' and provided by the sensor, each point consists of three double-precision values (x, y, and z). Consequently, the actual data size can be calculated as follows:

$$\text{data size} = 3 \times \text{valid points} \times \text{double} \tag{4.2}$$

```cpp
typedef struct ProfilePoint
{
  double x;   // position along laser line
  double y;   // position along the direction of travel
  double z;   // height (at the given x position)
  unsigned char intensity;
} ProfilePoint;
```

Code 4.1: Structure of a 3Dpoint

Additionally, an End of Message (ETX) marker is used to indicate the message's completion. The C++ code for sending this message from the socket server is provided below:



int	int	double	int
STX	Data size	Data	ETX

Figure 4.9: socket message format

```cpp
int messagesize = 3*sizeof(int) + (sizeof(double)*3*valid_point); // define message size
char* message = new char[messagesize]; //initialize message
int start_msg=111111111; // define a start message
memcpy(message, &start_msg, sizeof(int)); // copy the start message
memcpy(message+sizeof(int), &valid_point, sizeof(int)); // copy the size of the point cloud
int offset = 2*sizeof(int); // actual offset of the message
for (rowIdx = 0; rowIdx < GoSurfaceMsg_Length(surfaceMsg); rowIdx++)
{
     for (colIdx = 0; colIdx < GoSurfaceMsg_Width(surfaceMsg); colIdx++)
     {
         if (surfaceBuffer[rowIdx][colIdx].z != INVALID_RANGE_DOUBLE)
         {
             memcpy(message+offset, &surfaceBuffer[rowIdx][colIdx].x, sizeof(double));
             offset+=sizeof(double);
             memcpy(message+offset, &surfaceBuffer[rowIdx][colIdx].y, sizeof(double));
             offset+=sizeof(double);
             memcpy(message+offset, &surfaceBuffer[rowIdx][colIdx].z, sizeof(double));
```

```
            offset+=sizeof(double);
        }
    }
}
int end_msg=777777777; // define a end of message
memcpy(message+offset, &end_msg, sizeof(int));
```

Code 4.2: Sending point cloud through socket comunication

After sending the message, the socket client will wait for an acknowledgment from the server to ensure that the message is correctly received. In case of a negative acknowledgment, the client will reattempt to send the message for a predetermined number of attempts.

4.2.2 Socket server

The socket server, running within the Linux virtual machine, is implemented using Python. Its primary purpose is to connect to the socket client in the Windows environment and receive messages. The following Python code demonstrates how the socket server connects to the client:

```
int serverSocket, clientSocket;
struct sockaddr_in serverAddr, clientAddr;
socklen_t clientAddrLen = sizeof(clientAddr);

// Create socket
serverSocket = socket(AF_INET, SOCK_STREAM, 0);
if (serverSocket < 0) {
    std::cerr << "Error creating socket." << std::endl;
    return 1;
}
serverAddr.sin_family = AF_INET;
serverAddr.sin_addr.s_addr = INADDR_ANY;
serverAddr.sin_port = htons(PORT); // # define PORT 12345

// Bind socket to address
if (bind(serverSocket, (struct sockaddr *)&serverAddr, sizeof(
    serverAddr)) < 0) {
    std::cerr << "Error binding socket to address." << std::endl;
    return 1;
}

// Listen for incoming connections
```

```cpp
if (listen(serverSocket, 1) < 0) {
    std::cerr << "Error listening for connections." << std::endl;
    return 1;
}
std::cout << "Server is listening for connections." << std::endl;

// Accept client connection
clientSocket = accept(serverSocket, (struct sockaddr *)&clientAddr, &
    clientAddrLen);
if (clientSocket < 0) {
    std::cerr << "Error accepting client connection." << std::endl;
    return 1;
}
std::cout << "Client connected." << std::endl;
```

Code 4.3: Server connection with client

After establishing a connection with the client, the socket server will wait for the reception of a message. Once the message is received, it will be decoded, and the server will check if the message is complete. The server will then send an acknowledgment to the client. If the acknowledgment is negative, the client will resend the message. Otherwise, the server will proceed with processing the received data for create a ROS PointCloud2 message, in order to publish it.

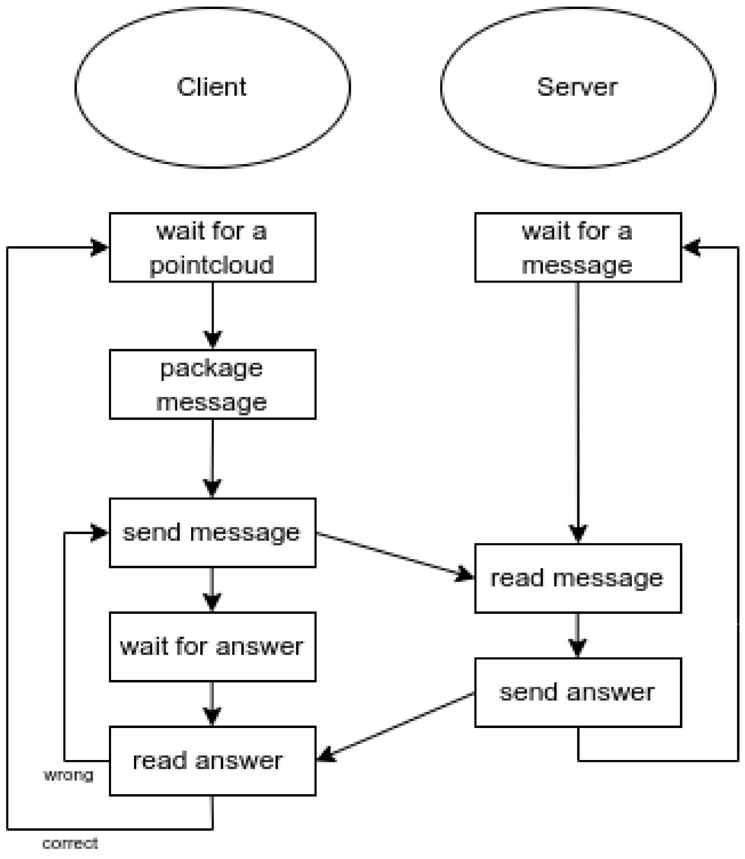


Figure 4.10: socket protocol

This communication method allowed for the seamless integration of the sensor data acquisition program with the rest of the system, despite the constraints imposed by the virtual machine environment. By utilizing socket communication, the AdaptMark project successfully bridged the gap between the Windows and Linux environments, enabling efficient data exchange and system coordination.

5 Testing and Evaluation

In this chapter, we undertake a thorough examination of the testing process that the developed system has undergone to validate its functionality, accuracy, and overall performance. The principal objective of these tests is to ensure the seamless identification of various 3D pieces on a moving conveyor and the precise determination of their positions and orientations within the system.

5.1 Testing Setup

During the testing phase, we employed point clouds as a substitute for real pieces. These point clouds were generated from previous scans of the pieces in different positions on the conveyor: at the center, to the right, and to the left. Additionally, the point cloud of each piece as stored in the database was retained. The strategic use of these point clouds significantly expedited the testing phase and allowed us to focus on two crucial aspects: the identification of pieces and the estimation of their positions. This approach circumvented the uncertainties associated with conveyor movement and sensor performance, providing a controlled environment for rigorous testing.

To simulate the sensor output point clouds accurately, a dedicated program was developed to publish point clouds that emulate the sensor's data output. This program retrieves a point cloud from a specified file path, creates a ROS message, and publishes it on the topic used by the sensor, called '/gocator_point_cloud'

The initial step of our testing process involved populating the system's database with a diverse set of pieces. This task was accomplished by utilizing the real sensor and the developed user interface.

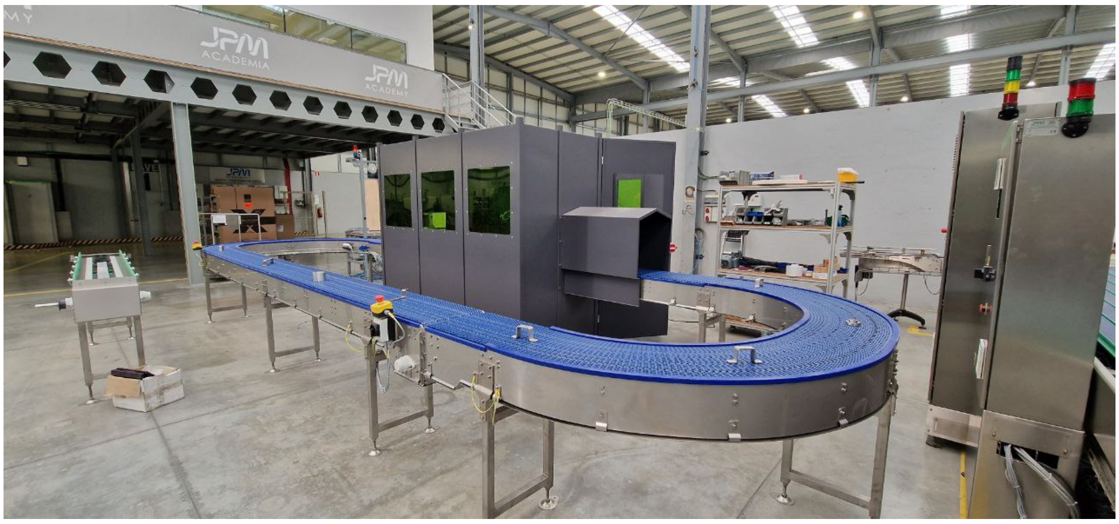
Upon completion of this phase, the system's database contained a total of 11 distinct pieces, each accompanied by four distinct point clouds. These point clouds were generated at different instances: one during the initial insertion of the piece into the database and three others at separate times. Specifically, one point cloud was acquired with the piece positioned at the center of the conveyor, another with the piece on the left side, and the third with the piece on the right side. Using different point clouds of the same piece provided more variability in the point clouds, allowing us to account for a wider range of real-world scenarios.

5.2 Identification Testing

To test the identification module, a subset of 11 pieces was selected from the initial set of 18 (pieces 1 to 9, as well as 13 and 18, see Figure 3.10). This choice was motivated by several factors. For instance, the use of a different conveyor system, which is less stable, made it impractical to study pieces such as IDs 14 and 17 (see Figure 3.10). These pieces are susceptible to small oscillations, which could result in poor scans or even cause them to fall from their positions.

One key distinction in this set of tests compared to previous ones is the utilization of a real-world scenario in which the sensor will operate. This scenario includes the conveyor system and the use of the database. Furthermore, in these tests, the original piece is compared with three point clouds obtained in different scenarios, making recognition more challenging.

The testing process began with the population of the database using the selected pieces. Subsequently, the system was tested using all available point clouds for each piece. This includes the point cloud extracted during this phase, along with three additional point clouds captured when the piece was in different positions on the conveyor: center, left, and right.



(a) Setup in the factory

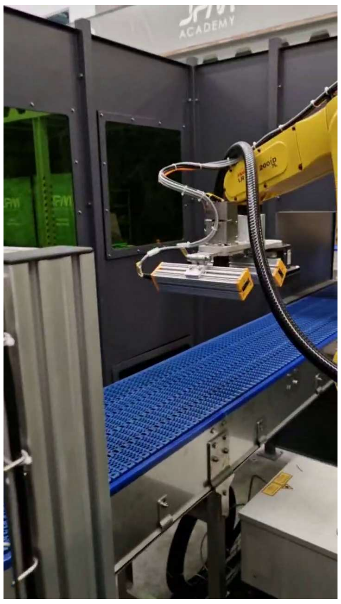


Figure 5.1: Arm with marking laser

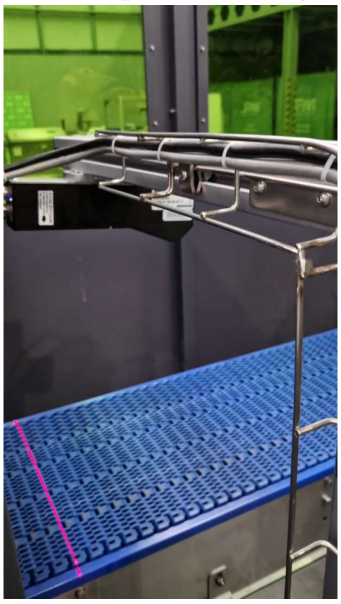


Figure 5.2: laser scanner

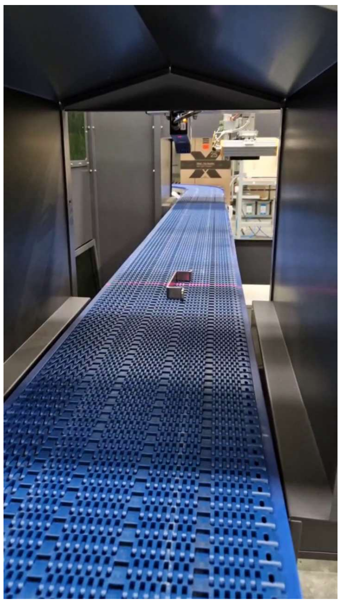


Figure 5.3: Pieces on the conveyor

Figure 5.4: Our setup in the factory

5.3 Pose Estimation Testing

In addition to identifying pieces accurately, the system must correctly determine the position and orientation (pose) of the piece on the conveyor. After identifying a piece, the system utilizes its associated category to estimate the position of the piece's center, as well as any other relevant features such as boundaries (circles or shapes), holes, and notches. By combining this estimated information with the database information for the piece, the system computes the exact position and orientation for marking. This phase of testing reveals two types of errors:

- **Center Error**: This metric measures the difference between the estimated center position and the actual center position of the piece, calculated as the Euclidean distance between the two points.
- **Rotation Error**: This metric evaluates the accuracy of estimating the angular position of features (e.g., notches or holes) on circular pieces. The error in the angular estimation of these features directly translates to an error in the marking position.
- **Mark Error**: This metric represents the discrepancy between the estimated marking position and the actual marking position.

To calculate these errors accurately, it's essential to determine the true center of the piece and its orientation. Given the absence of an exact method for this, we adopted a pragmatic approach. We overlaid the manually drawn features of the piece, as defined during the initialization phase (when the piece was inserted into the system database), onto the point cloud. The goal was to manually align these features with the point cloud until they visually matched. While this method is not exact, it represents the best practical solution we could find for estimating the center and orientation of the piece.

Next, we employ three error calculation metrics:

- **Center Error Calculation**: To determine the Center Error, we compute the Euclidean distance between the system's estimated center position and the manually identified actual center position of the piece. This metric quantifies the accuracy of the system in locating the center of the piece.
- **Rotation Error Calculation**: For assessing the rotation error, we evaluate the system's ability to estimate the angular positions of features, such as notches or holes, on circular pieces. This metric assesses the deviation in the angular estimation of these features from their actual positions, subsequently impacting the calculated marking position.

- **Mark Error Calculation**: The Mark Error is calculated by measuring the difference between the estimated marking position and the actual marking position of the piece, which is derived from the manually identified center. This calculation provides insight into the precision of the system in determining the exact marking location.

5.4 Results and Analysis

In the following section the numerical results of the test presented before will be illustrated and exterminated to better understand their meaning and their influence on the final results.

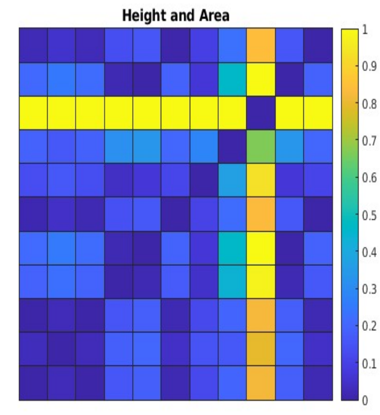
5.4.1 Identification results

The identification process, as it was developed, combines height, area, and the GOOD descriptor to identify each piece. For every piece, these three features are computed and compared with all the information in the database. This comparison yields a distance measurement, which is then normalized to obtain a vector of values ranging from zero to one. Finally, the sums of the height and area, along with the GOOD descriptor, produce a vector in which the value closest to zero indicates the identified piece.

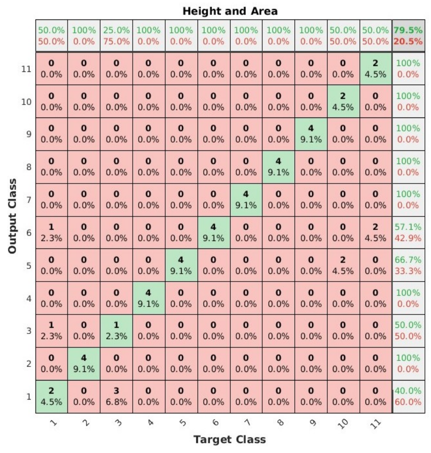
After testing all 11 pieces, each four times, we obtained matrices of 11 by 44 for each descriptor. To simply the results, it was computed the mean for each feature (e.g., height, area, and GOOD) for each piece across the four tests. So at the end we will have matrices 11 by 11.

From these matrices, we generated confusion matrices and graphical representations to assess the classification performance. Confusion matrices are valuable tools for visualizing the accuracy of our identification process. These graphical representations, generated using tools like 'imagesc' in MATLAB, provide a visual depiction of the matrices. In these representations, values closest to zero appear as dark blue, indicating correct identification, while values closer to one appear as yellow. Thus, a dark blue diagonal signifies accurate identification of the pieces

One of the key observations from this test is the necessity to adjust the normalization of height and area. Originally, these two features were summed together and then normalized as a single value. However, as evident from the

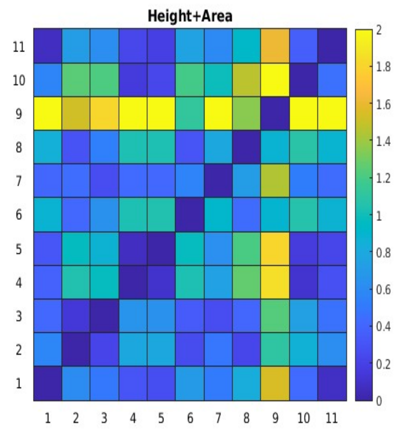


(a) Height and Area - Matrix Image

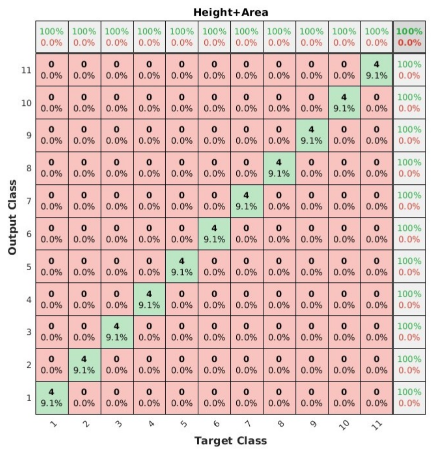


(b) Height and Area - Confusion Matrix

Figure 5.5: Height and Area normalized together



(a) Height and Area - Matrix Image



(b) Height and Area - Confusion Matrix

Figure 5.6: Height and Area normalized separately

resulting matrices, this approach heavily favored area and didn't adequately consider height due to their differing magnitudes. The area values are in the order of thousands, while the height values are in the order of hundreds.

To address this imbalance, a correction was implemented. Now, each feature (height and area) is individually normalized before being combined with the GOOD descriptor. This adjustment ensures that each feature contributes proportionately to the final identification result. The corrected approach, which sums the normalized features with GOOD, provides a more balanced representation of the piece's characteristics in the identification process.

From the confusion matrices in Figure 5.5 and 5.6, you can observe the performance of height and area when they are treated differently in the identification process. In Figure 5.5b, which shows the confusion matrix for height and area when normalized together, there are several instances of misclassified classes, from the graphical representation of the normalized distance matrix in Figure 5.5a, you can observe the high degree of confusion along the diagonal. Conversely, figure 5.6b, which displays the confusion matrix for height and area when normalized separately, demonstrates a more accurate classification of classes, moreover, the diagonal in Figure 5.6a is noticeably more distinct.

The complete matrices for each feature in both graphical representation and confusion matrix form are presented in the appendix. These matrices provide a detailed view of the classification performance and the impact of different feature normalization methods on the identification process.

5.4.2 Pose estimation results

Piece id	Cat.	Center err. [mm]	Mark err. [mm]	Rotation err. [deg]
1	1	1,16	2,34	2,02
2	2	1,14	1,15	2,81
3	4	1	1	0
4	4	0,9	0,9	0
5	4	1,1	1,1	0
6	2	1,12	8,07	25,56
7	4	0,91	0,91	0
8	3	1,03	1,07	1,69
9	3	1,10	1,16	0,56
10	4	1	1	0
Mean value		1,05	1.97	6,53

Table 5.1: Table of the results of the pose estimation testing.

In Table 5.1, it's important to note that the values of center error, mark error, and rotation error for each piece are calculated as the means of four measurements.

An interesting observation arises for pieces categorized as 4, which do not have holes or notches. In such cases, the center error and mark error are consistently the same. This occurs because, in the absence of holes or notches, the marking position is solely determined by the distance from the center. Consequently, the rotation error becomes negligible for these pieces, resulting in a value of zero.

It's worth highlighting a particular instance with piece ID 6. During one of the four tests, there was a significant deviation in the estimation of the rotation, with an error of 95 degrees. This rotation error translated to a substantial mark position error of 28 mm from the actual location. In essence, the template position appeared to be rotated by nearly 90 degrees more than expected. To rectify this error, further investigation into the estimation of the notch position is required.

Another valuable insight that can be gleaned from the data is the system's precision in estimating the mark position on the piece's surface. This precision is represented by the mean value of the mark errors listed in Table 5.1.

6 Conclusions and Future Works

In this concluding chapter, we summarize the key findings and insights obtained during the development and testing of the Automatic Laser Marking System. We also outline potential directions for future research and system enhancements.

6.1 Final Consideration

The AdaptMark project, developed through collaboration between JPM Industry and the University of Aveiro, seeks to streamline the engraving process, traditionally reliant on human operators. It aims to develop an automatic laser engraving system capable of autonomously engraving pieces of various geometries and materials. This automatic system is divided into three core subsystems: the identification subsystem, the manipulation and laser engraving subsystem, and the quality check subsystem

This thesis does not exclusively focus on any one of these subsystems. Instead, its primary objective is to bridge the gap between these subsystems and integrate them seamlessly, while also enabling supervision by human operators. This entails complete system integration to facilitate the supervision and control of the entire process.

The initial task was to establish a connection between human operators and the machine. By understanding the operator's requirements, we embarked on the development of a user interface tailored to meet these needs. The central function of this user interface is to populate the database with information about new pieces. This enables operators to easily scan a piece and select its features. Subsequently, additional functionalities were developed to review the historical performance of the system and its current status. The development of this user interface posed several challenges, spanning from graphic element manipulation and geometric calculations to image processing, database information exchange, and managing the complexities of a sizable codebase.

Following this, efforts were directed toward connecting the subsystems. The challenge here lay in integrating different programs, written in varying languages and operating on different systems. The solution was to leverage the capabilities of ROS, which necessitates Linux to run. However, some components of the system were designed to function on Windows. To bridge this divide, we retained Windows as the primary OS and introduced a virtual machine running Linux for ROS. This architectural choice introduced its own set of challenges, notably in inter-system communication. The solution involved implementing socket communication between these two operating systems. While this system setup was a pragmatic compromise, future enhancements may involve simplifying the architecture to expedite system responsiveness.

Lastly, we conducted a series of simple tests to assess the system's precision in identifying pieces and estimating their poses. The results indicate that the system performs well overall. However, the identification part exhibited some uncertainties not anticipated in prior studies, while the pose estimation yielded satisfactory results.

6.2 Future Works

One of the primary objectives of this thesis was to successfully integrate the various components of the system. While this integration has been achieved, it has introduced some complexities, such as the need for socket communication between different operating systems. As we look towards the future, simplifying the system setup is a key area for improvement. This may involve streamlining the architecture, potentially by utilizing a single operating system, to enhance the overall system workflow.

In the development of the user interface, our focus was primarily on the integration of the identification subsystem and related functionalities. However, there are crucial aspects of the engraving process that have not yet been addressed. Specifically, the control of the conveyor and the robotic arm, which are essential for the finalization of the engraving process, were not within the scope of this thesis. During the development and testing phases, the system operated with a continuously running conveyor and without the involvement of the robotic arm.

In future work, it is important to address these aspects effectively. This involves correctly managing the movement of the conveyor, together with the coordinates for the robotic arm.

Following these immediate developments, the next phase of system enhancement should focus on accommodating pieces with varying geometries. This expansion will allow the system to engrave a broader range of components effectively.

In summary, the path forward for the AdaptMark project includes simplifying system setup, incorporating conveyor and robotic arm control, and expanding capabilities to handle diverse piece geometries. These future works will significantly contribute to the project's evolution and effectiveness in meeting its objectives.

References

[1] Konstantinos G Derpanis. "Overview of the RANSAC Algorithm". In: *Image Rochester NY* 4.1 (2010), pp. 2–3.

[2] M. Fischler and R. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395. URL: /brokenurl#%20http://publication.wilsonwong.me/load.php?id=233282275.

[3] Abubakar Gezawa et al. "A Review on Deep Learning Approaches for 3D Data Representations in Retrieval and Classifications". In: *IEEE Access* PP (Mar. 2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.2982196.

[4] *ROS concepts*. URL: http://wiki.ros.org/ROS/Concepts
Retrieved: 19/7/2023.

[5] *ROS roslaunch*. URL: http://wiki.ros.org/roslaunch
Retrieved: 19/7/2023.

[6] *ROS Rviz*. URL:http://wiki.ros.org/rviz
Retrieved: 19/7/2023.

[7] *Mysql*. URL:https://en.wikipedia.org/wiki/MySQL
Retrieved: 19/7/2023.

[8] *Tkinter*. URL:https://en.wikipedia.org/wiki/Tkinter
Retrieved: 19/7/2023.

[9] *ROS* rqt_graph. URL: https://wiki.ros.org/rqt_graph?distro=noetic
Retrieved: 19/7/2023.

[10] *3D-Transformation*. URL:https://www.geeksforgeeks.org/computer-graphics-3d-composite-transformation/
Retrived: 19/7/2023.

[11] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[12] S. Hamidreza Kasaei et al. "GOOD: A global orthographic object descriptor for 3D object recognition and manipulation". In: *Pattern Recognition Letters* 83 (2016). Efficient Shape Representation, Matching, Ranking, and its Applications, pp. 312–320. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2016.07.006. URL: https://www.sciencedirect.com/science/article/pii/S0167865516301684.

[13] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm". In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423.

[14] Haiping Si et al. "Analysis of Socket Communication Technology Based on Machine Learning Algorithms Under TCP/IP Protocol in Network Virtual Laboratory System". In: *IEEE Access* 7 (2019), pp. 80453–80464. DOI: 10.1109/ACCESS.2019.2923052.

[15] *What Is Matplotlib In Python?* Retrieved on 04/09/2023. URL: https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/.

[16] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

Appendix

Complete results from the identification test:

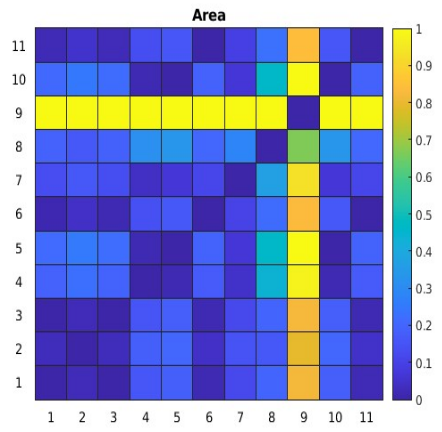


Figure 6.1: Area - Matrix Image

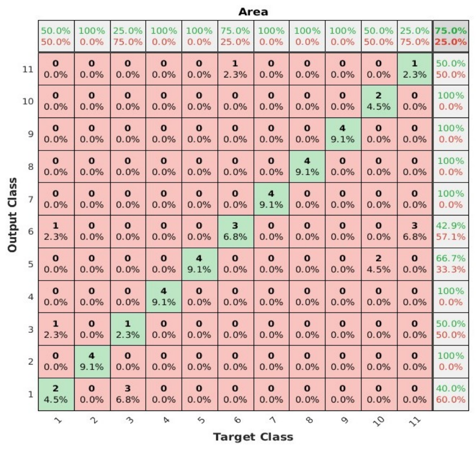


Figure 6.2: Area - Confusion Matrix

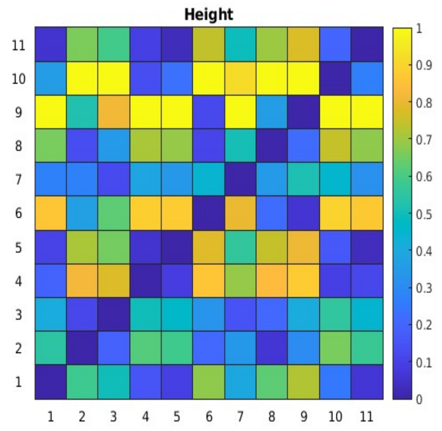


Figure 6.3: Height - Matrix Image

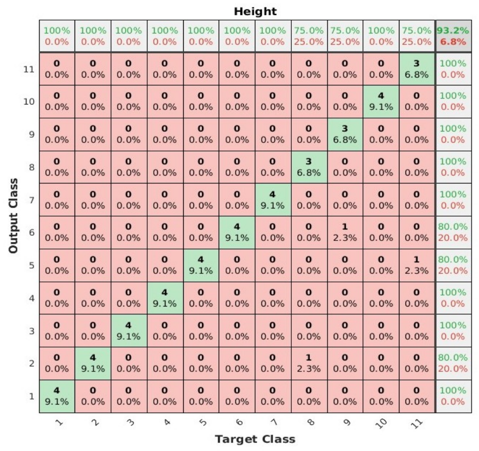


Figure 6.4: Height - Confusion Matrix

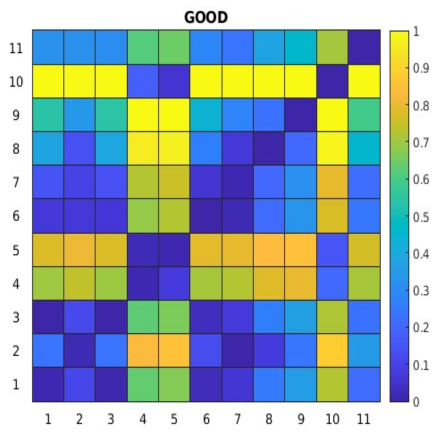


Figure 6.5: GOOD - Matrix Image

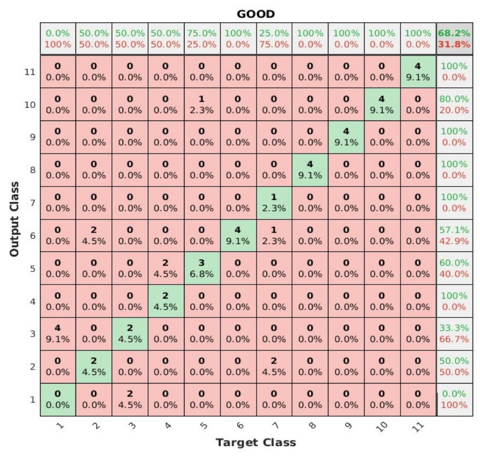


Figure 6.6: GOOD - Confusion Matrix

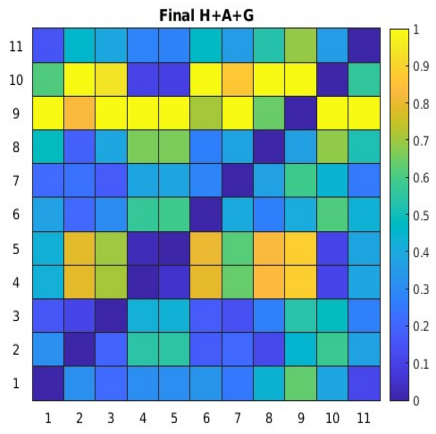


Figure 6.7: Final - Matrix Image

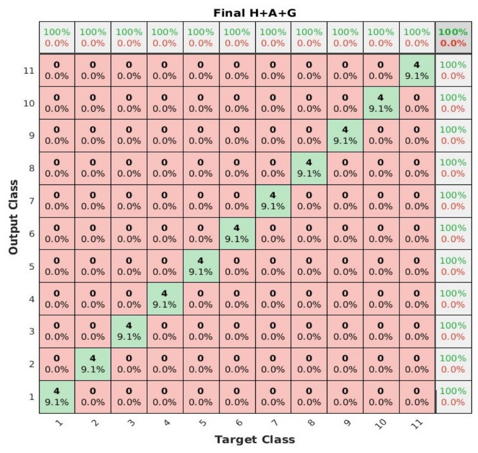


Figure 6.8: Final - Confusion Matrix

Figure 6.9: Graphical Representation and Confusion Matrices for Different Features

Acknowledgments

To my esteemed advisors, Professor Pedro Fonseca and Professor Nuno Lao, i extend my sincere thanks for your help, support, and the wealth of knowledge you imparted during the course of this thesis.

My deepest appreciation also goes to my colleagues, Reza Javanmard and Diogo Barbosa Fernandes, whose collaboration and shared insights not only made this research intellectually stimulating but also enjoyable. Our teamwork has been instrumental in our collective success.

I would like to extend my gratitude to JPM Industry for affording me the opportunity to work on this innovative project. In particular, I want to thank Fillipe Ribeiro for his continuous assistance throughout the project and his warm hospitality during my tenure at the company.

To everyone who has been part of this academic journey, whether through significant contributions or small gestures of support, your involvement has left a lasting impact, and I am truly thankful for your invaluable assistance and encouragement.

To my beloved family and friends, I owe a profound debt of gratitude for your unwavering support, encouragement, and understanding during this demanding phase of my academic pursuit. Your love and unwavering encouragement have been a constant source of strength.