TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical Power Engineering and Mechatronics

MULTI-CAMERA VISION-BASED INVENTORY MANAGEMENT SYSTEM

MITME KAAMERAGA NÄGEMISPÕHINE VARUDE HALDAMISE SÜSTEEM

MASTER THESIS

Student:	Celia Ramos López-Contreras
Student code:	214184MAHM
Supervisor:	Dhanushka Liyanage, PhD
Co-supervisor:	Daniil Valme, Early Stage Researcher

Tallinn, 2023

AUTHOR’S DECLARATION

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

“.......” .... 20…...

Author: ....
/signature /

Thesis is in accordance with terms and requirements

“.......” .... 20.….

Supervisor: …..........................
/signature/

Accepted for defence

“.......”....................20.… .

Chairman of theses defence commission: ....
/name and signature/

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

ABSTRACT

Author: Celia Ramos López-Contreras	*Type of the work:* Master Thesis
Title: Multi-camera vision-based inventory management system	
Date: 18.05.2023	90 *pages (the number of thesis pages including appendices)*
University: Tallinn University of Technology *School*: School of Engineering *Department:* Department of Electrical Power Engineering and Mechatronics	
Supervisor(s) of the thesis: Dhanushka Liyanage, PhD; Daniil Valme, Early Stage Researcher *Consultant(s):*	

Abstract:

The purpose of this thesis is to develop a multi-camera vision-based inventory management system, that is simple and suitable for smaller companies with a low number of resources. The approach to the development of this project appeared to meet the department's need for a system to control and monitor the available materials and the resulting system creates a registry keeping a record of the users' use of such equipment. For that purpose, four goals were initially set, corresponding to the main sections implemented in the project: development of a face recognition application to detect the user, development of two object recognition systems for the lab equipment and the camera kits, and finally, implementation of a program that combines all the previous solutions and provides the information collected by the inventory system.

The final implementation consists of: tool detection by YOLOv8 small algorithm trained with a batch size of 16 and for 150 epochs; boxes (camera equipment) detection using ArUco markers of 5x5 bits and image size of 150 pixels; and face detection and recognition using a Haar feature-based cascade classifier and LPBH respectively. Finally, the functioning of the whole system was tested by the simulation of different scenarios after which, it is shown that the different objectives have been met and the results are satisfactory, with the system being able to keep the inventory managed and updated.

Keywords: Inventory management, object detection, YOLOv8, ArUco markers, facial recognition.

LÕPUTÖÖ LÜHIKOKKUVÕTE

Autor: Celia Ramos López-Contreras	*Lõputöö liik:* Magistritöö
Töö pealkiri: Mitme kaameraga nägemispõhine varude haldamise süsteem	
Kuupäev: 18.05.2023	90 *lk (lõputöö lehekülgede arv koos lisadega)*

Ülikool: Tallinna Tehnikaülikool

Teaduskond: Inseneriteaduskond

Instituut: Elektroenergeetika ja mehhatroonika instituut

Töö juhendaja(d): Dhanushka Liyanage, PhD; Daniil Valme, Early Stage Researcher

Töö konsultant (konsultandid):

Sisu kirjeldus:

Lõputöö eesmärgiks on välja töötada mitme kaameraga visioonil põhinev laohaldussüsteem, mis on lihtne ja sobilik väikese ressursiga väiksematele ettevõtetele. Selle projekti arendamise lähenemisviis näis vastavat osakonna vajadusele olemasolevate materjalide kontrollimise ja jälgimise süsteemi järele ning sellest tulenev süsteem loob registri, mis hoiab arvestust selliste seadmete kasutajate kasutamise kohta. Selleks seati algselt neli eesmärki, mis vastavad projekti põhiosadele: näotuvastusrakenduse väljatöötamine kasutaja tuvastamiseks, kahe objektituvastussüsteemi väljatöötamine laboriseadmete ja kaamerakomplektide jaoks ning lõpuks juurutamine. programmi, mis ühendab kõik varasemad lahendused ja annab inventuurisüsteemi kogutud teabe.

Lõplik teostus koosneb: tööriista tuvastamisest YOLOv8 väikese algoritmi abil, mis on koolitatud partii suurusega 16 ja 150 epohhi jaoks; kastide (kaameraseadmete) tuvastamine, kasutades ArUco markereid suurusega 5x5 bitti ja pildi suurust 150 pikslit; ning näotuvastus ja tuvastamine, kasutades vastavalt Haari funktsioonipõhist kaskaadiklassifikaatorit ja LPBH-d. Lõpuks testiti kogu süsteemi toimimist erinevate stsenaariumide simuleerimisega, mille järel on näidatud, et erinevad eesmärgid on täidetud ja tulemused on rahuldavad ning süsteem suudab hoida laoseisu hallatuna ja ajakohasena.

Märksõnad: Varude haldamine, objektide tuvastamine, YOLOv8, ArUco markerid, näotuvastus.

THESIS TASK

Thesis title in English:	Multi-camera vision-based inventory management system.
Thesis title in Estonian:	Mitme kaameraga nägemispõhine varude haldamise süsteem.
Student:	Celia Ramos López-Contreras, 214181MAHM
Programme:	MAHM, Mechatronics
Type of the work:	Master Thesis
Supervisor of the thesis:	Dhanushka Liyanage, PhD
Co-supervisor of the thesis: (company, position and contact)	Daniil Valme, Early Stage Researcher
Validity period of the thesis task:	**Validity period is given by supervisor** Choose an item. Choose an item.
Submission deadline of the thesis:	**18.05.23**

Supervisor (signature) Student (signature) Head of programme (signature)

Co-supervisor (signature)

1. Reasons for choosing the topic

An inventory system is the process by which you track and keep record at all times of the materials or goods owned any entity, be it an individual or a group or company. Being able to access this information is crucial to implement an organized system and is nowadays used in numerous industrial manufacturers applications, commercial solutions, and warehouses of many companies.

The operation or methodology used in these systems varies from the basic concept of keeping a list manually updated by a user, to the use of modern technologies that are still under development to keep track of the owned objects. Among some renown examples, Amazon Go is a checkout-free shopping system that uses a complex system combining computer vision, sensor fusion and deep learning to be able to detect when a product is taken from or returned to the shelves [3]. Walmart's inventory system uses a Radio-Frequency Identification (RFID) technology to identify the items taken using radio waves [5], [6]. R-Kiosk Go, the first unmanned stored in Estonia, uses a combination of artificial intelligence, several sensors (specially weight sensors) and cameras to control the items taken by each customer [7].

Using a combination of different technologies results in complex and expensive systems. The goal to reach with the development of the system presented in this document is to achieve a performance similar to that developed by large companies, using only machine vision methods, and therefore avoiding the use of additional hardware that increases the price and complexity of the system, making the system accessible for smaller businesses or companies.

2. Thesis objective

The goal is to create a multi-camera vision-based inventory management system, that will create a registry using as data the users with access to the available material and the material itself and keeps a record of its use.

3. List of sub-questions:

1. Development of face recognition technology, that will detect the specific person borrowing an item, from a limited database of personnel allowed to use the equipment in the department.
2. Development of an object recognition system for the lab equipment available.
3. Development of an object recognition system for the camera kits available, which are placed in boxes of similar appearance.
4. Implementation of a user-friendly interface that allows easy access to the information collected by the inventory system.

4. Basic data:

For the implementation of the thesis work several datasets will be collected. First, the user's database would be needed, for the implementation of the face recognition system.

Secondly, an image dataset of the selection of lab equipment available in the department as well as the camera equipment, which in this particular case, will be recognised from a personalized tag placed on each box, therefore needing the tag's information recollection. Both of the last elements will be obtained from the machine vision's department, according to the material available.

5. Research methods

In order to achieve the best possible implementation of the system proposed first, a comprehensive literature review will be carried out to study existing technologies and products and perform their comparison, with the objective of choosing the best possible methodology. Such literature review will be divided into different sections, each of them focusing in one specific related topic: inventory management systems, face recognition technologies and object detection methods, concluding with a summary of the results obtained and a consequent selection of the project's methodology.

When analysing the results from the different sections implemented for the system, data from the image database (used for both training and testing of the model) as well as real time images from the camera system will be used for each section.

6. Graphical material

A number of graphical elements will be included throughout the document, although it is not yet possible to determine whether they will be mostly in the body of the document or in the appendixes.
Among other things, it'll be found different tables with technical requirements or characteristics, workflow diagrams, schemes of the hardware set up implemented as well as pictures of the prototype and set up environment.

7. Thesis structure

The document will be structured in the following way:

1. Introduction
2. Literature review
    - 2.1. Inventory management
    - 2.2. Machine vision-based methods
    - 2.3. Fiducial marker detection
    - 2.4. Face recognition technologies
    - 2.5. Literature review conclusion
    - 2.6. Aim of the work
3. Creating the solution
    - 3.1. Collection of data
    - 3.2. Implementation of methodology
    - 3.3. Validation
    - 3.4. Testing the results
4. Conclusion

8. References

The type of sources used are mostly research articles, reports and books. Some examples of such are:

T. Diwan, G. Anirudh, and J. v. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimed Tools Appl*, 2022, doi: 10.1007/S11042-022-13644-Y.

Z. Zhang, Y. Hu, G. Yu, and J. Dai, "DeepTag: A General Framework for Fiducial Marker Design and Detection," *IEEE Trans Pattern Anal Mach Intell*, no. 01, pp. 1–1, May 2022, doi: 10.1109/TPAMI.2022.3174603.

B. Li, J. Wu, X. Tan, and B. Wang, "ArUco Marker Detection under Occlusion Using Convolutional Neural Network," *Proceedings - 5th International Conference on Automation, Control and Robotics Engineering, CACRE 2020*, pp. 706–711, Sep. 2020, doi: 10.1109/CACRE50138.2020.9230250.

P. A. Harsha Vardhini, S. P. R. D. Reddy, and V. P. Parapatla, "Facial Recognition using OpenCV and Python on Raspberry Pi," *2022 International Mobile and Embedded Technology Conference, MECON 2022*, pp. 480–485, 2022, doi: 10.1109/MECON53876.2022.9751867.

9. Thesis consultants

10. Work stages and schedule

No.	Task description	Deadline
1.	Introduction and literature review chapters	02.12.22
2.	Thesis extended proposal submission	21.02.23
3.	Selection of object and face recognition methods	15.01.23
4.	Implementation of lab equipment recognition system	20.02.23
5.	Implementation of camera's equipment recognition system	05.03.23
6.	Implementation of face recognition system	26.03.23

No.	Task description	Deadline
7.	Combination of technologies for prototype and design of user-friendly interface	09.04.23
8.	Testing the prototype, optimization	20.04.23
9.	Summarizing results and writing report	18.05.23
10.	Submission thesis document	26.05.23
11.	Thesis defences	01.-05.06.23

CONTENTS

List of figures

List of tables

PREFACE

The present master thesis describes the development of a multi-camera vision-based inventory management system, that allows to control the available materials borrowed by the users in the department.

I would like to express my gratitude to Dhanushka Liyanage, Engineer, and Daniil Valme, Early Stage Researcher, for the mentoring during the whole process, and to my family and friends for the constant support and encouragement.

List of abbreviations

2D	2 dimensional
AI	Artificial Intelligence
CLI	Command-Line Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DFL	Detection Loss Function
FN	False Negative
FOV	Field of View
FP	False Positive
FPS	Frame Per Second
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HOG	Histograms of Oriented Gradient
IoU	Intersection over Union
JWO	Just Walk Out
LBP	Local Binary Pattern
LBPH	Local Binary Patterns Histograms
LFW	Labelled Faces in the Wild
mAP	mean Average Precision
MS COCO	Microsoft Common Object in Context
OpenCV	Open-Source Computer Vision
PASCAL VOC	PASCAL Visual Object Classes
R-CNN	Region-based Convolutional Neural Network
RFID	Radio-Frequency Identification
SIFT	Scale-Invariant Feature Transform
SSD	Single Shot Detector
SVM	Support Machine Vector
TN	True Negative
TP	True Positive
YOLO	You Only Look Once

1 INTRODUCTION

Inventory management systems are fundamental for industry manufacturing facilities, warehouses, or companies of any size. By controlling or managing a company’s inventory, including all kinds of elements from raw materials to finished products or assets, reliability, security, and efficiency are added to the system. Some of the solutions already developed use different combinations of technologies, like artificial intelligence, sensory systems, computer vision, deep learning, or Radio-Frequency Identification (RFID), among others. The resulting systems are usually complex and expensive, only reachable for large companies with extensive funding and many means or access to huge databases, usually necessary for this type of developments.

The approach to the development of this project appeared as a response to the demand from the department to have a system to control the available materials. Therefore, the goal is to implement a multi-camera vision-based inventory management system that will create a registry using as data the users with access to the available material and the material itself and keeps a record of its use. Although there exist numerous technologies with which inventory management systems can be implemented, through the use of only vision-based technologies, the system is greatly simplified, and the same functionalities can be obtained at a lower price.

For the purpose of reaching such goal, several sections are implemented within the system:

- Development of face recognition technology, that will detect the specific person borrowing an item from a limited database of personnel allowed to use the equipment in the department.
- Development of an object recognition system for the lab equipment available.
- Development of an object recognition system for the camera kits available, which are placed in boxes of similar appearance.
- Implementation of a program that combines all the previous solutions creating the inventory management system’s logic, and providing the information collected by such system.

The implementation of the system described requires the collection of several datasets. A user’s dataset is needed for the face recognition system, as well as image datasets of both objects and fiducial markers, for the detection of the material available, and that must be included in the inventory management system.

Since the system implemented is oriented towards small and medium size companies that don’t have access to large resources, and although the collection of several datasets is essential, these would be limited. The implementation of the system with these limitations can be considered as the second main objective of the project. Usual systems that require facial recognition or object detection algorithms regularly require at least hundreds of images of each object or user to be detected, while the described system is aimed to perform the same actions with a considerably smaller number of images.

However, consequently, this has repercussions on the functioning of the system, limiting its flexibility. A minimum number of conditions have to be met in order for the system to work correctly and variations in the environment may have an influence on the system's performance.

In chapter 2, the literature review is introduced, with a comparative analysis of the already existing similar solutions and the different methodologies available to implement the described system as well as the final choice. In chapter 3, the design concept is explained and is divided into two sections corresponding to the hardware set up and software architecture. In chapter 4, the different experiments and system's results are analysed and finally, in chapter 5, the conclusions and possible future works are stated.

2 LITERATURE REVIEW

2.1 Inventory management

Inventory management or control is the process of managing a company's inventory, which includes management of elements from raw materials to finished products, as well as tools or materials used by such company. This control or management is essential to run a business efficiently since both a shortage and an abundance of a company's inventory can be detrimental. Nowadays, inventory management systems are used in numerous industrial manufacturing facilities and warehouses. Therefore, inventory management is fundamental for companies of any size in order to keep track of all the assets belonging to such company, and to base complex decisions regarding when to restock inventory, what product amounts to purchase or produce, or what price to pay for such products [1].

The process or methodology followed in these systems varies from the basic concept of keeping a list manually updated by a user (manual tracking) to the use of modern technologies that are still under development to keep track of the owned objects. The number of methods that can be implemented for tracking inventory is very numerous, and the goal of such implementation is to avoid the tedious, repetitive, and lengthy process that simpler and more basic methods imply. Another reason for the implementation of more complex technologies for inventory tracking is to provide a security factor for these systems and avoid misplacement of both the inventory register and the inventories themselves. With further developed systems, we are adding reliability, security, and efficiency to the system [2].

Within the technologies used to implement inventory management systems, or other services that include in their structure similar working processes, we can find artificial intelligence, sensory systems, and machine vision techniques, among others. In 2018, Amazon opened the first store in Seattle with Just Walk Out (JWO) technology, Amazon Go, a checkout-free shopping system in which the customers can directly exit the store and the items taken are automatically charged to the user's account. With a complex system combining computer vision, sensor fusion and deep learning, JWO technology is capable of detecting when a product is taken from or returned to the shelves, therefore creating a virtual shopping cart [3]. This JWO technology is not only available at Amazon's own stores but also provided to third-party environments. As for the customer identification system used, there are two options available: one can be identified with a personal card that must be scanned at the entrance of the store, or an individual is recognised by the scan of their palm, with the new developed technology Amazon One [4].

Using computer vision algorithms, Amazon has developed a way to capture each's individual's palm, that can differentiate it from others by recognising the distinct features on and below the surface and it uses the information linked to each palm to provide a contactless service for the customers. Walmart Inc.'s inventory system uses RFID technology to identify the items taken using radio waves. A tag is attached to each product for its tracking, which is later scanned at the checkout registers, automatically updating the inventory left at the store [5][6]. Finally, in 2022, the first unmanned store is Estonia, R-Kiosk Go, was opened at Tallinn University of Technology. This store, with a similar operation to that used in Amazon Go, uses a combination of artificial intelligence, several sensors, and cameras, focussing on the use of weight sensors to control the items taken and without storing biometrics of the customers, therefore avoiding non-compliance with data protection regulations [7]. In the following table 2.1, the main characteristics of the three systems described above are listed:

Table 2.1 Comparison of related existing products [3]–[7]

	AMAZON GO	**WALMART INVENTORY MANAGEMENT**	**R-KIOSK GO**
Tracking of items	Just Walk Out (JWO) technology - computer vision, sensor fusion and deep learning	Radio-Frequency Identification (RFID)	Artificial intelligence, cameras, and sensors (focussing on weight sensors)
Personal identification	Palm scanner (Amazon One) Scan of credit or debit card App-based entry	No	App-based entry Scan of credit or debit card
Technology's purpose	Multiple product unmanned physical store	Inventory system on physical stores	Multiple product unmanned physical store

However, the use of combined technologies results in complex and expensive systems, unattainable for small businesses or individual users. Amazon, Walmart and R-kiosk are all medium to large companies, which have extensive funding for the development of these systems, and which have many means to obtain the huge databases necessary in many cases for these systems.

The goal to reach with the development of the system presented in this document is to achieve a performance similar to that developed by large companies, using only machine vision methods which can provide the necessary information for the implementation of an inventory management system, and therefore avoiding the use of additional hardware that increases the price and complexity of the system.

2.2 Machine vision-based methods for object recognition

Object recognition is a computer technology whose concept is based on the detection of objects of a specific class in digital images or videos. The main goal of the development of object recognition technologies is to imitate the human ability or intelligence to recognise and locate objects in a matter of moments, using a computer [8].

It is important to differentiate between detection and recognition. Detection is the process of finding relevant objects, of any class, within an image or video, while recognition has the additional task of classifying such objects within a determined class, previously defined [9]. However, the use of these concepts is not so differentiated nowadays, constantly referring to object detection as the recognition of certain objects within a class. Therefore, we can define object detection or recognition as the process that involves both locating the object within the image, usually rounding it with a bounding box, and classifies such object, predicting its class [10].

These technologies are in constant development and it's a highly researched area in recent times; there is a wide amount of uses, among which we can find face recognition and detection, a topic that will be further studied in the following sections, driving assistance, image retrieval, video surveillance and many others.

The methodology used in the implementation of object recognition systems can be divided into two main groups: neural network-based approaches, also referred to as deep learning, and non-neural approaches. Machine learning is an Artificial Intelligence (AI) encompassing parts of both groups, which is defined as the capability of a machine to imitate intelligent human behaviour, with minimal human interference. Deep learning is a subset of machine learning, which implies it is also an AI, that uses artificial neural networks to mimic the human learning process [11]. In spite of being similar technologies, the have a series of key differences that are summarized in the table below:

Table 2.2 Differences between machine learning and deep learning [11]

Conventional machine learning	**Deep learning**
Can train on smaller data sets	Needs large amounts of data
Correction and learning require human intervention	Self-learning from the environment and past data or mistakes
Shorter training and lower accuracy	Longer training and higher accuracy
Simple and linear correlations	Non-linear and complex correlations
Training on a CPU (Central Processing Unit)	Training requires a specialized GPU (Graphics Processing Unit)

2.2.1 Machine learning methodology

Machine learning or non-neural approaches for object recognition require first the use of a method for feature definition and extraction, called feature detection algorithms, followed by a classification method.

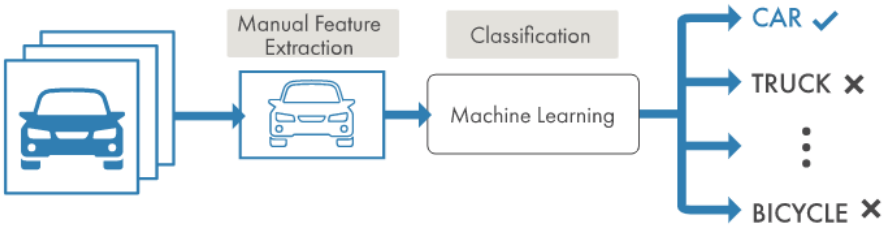


Figure 2.1 Machine learning general procedure [12]

The aim of feature extraction is to extract and represent features in a form adequate for the classification stage [13] and there are several methods for feature definition purposes, being Histograms of Oriented Gradient (HOG), Viola-Jones object detection framework (oriented towards face recognition technologies) and Scale-Invariant Feature Transform (SIFT) some of the most popular options.

HOG uses a feature extractor to identify objects from an image. The procedure is based on the extraction of the most necessary information, disregarding any non-important information, by the use of gradient orientation. Then, it converts the overall size of the image into the form of an array or feature vector. Among its advantages, we can consider its simplicity, easy to understand the information and the fact that it can be used to detect small-scaled images with less computational power. However, its accuracy might be ineffective in certain object detection scenarios with tighter space and is very time-consuming for complex pixel computation in large images [14].

SIFT is also a feature detection algorithm that locates local features in an image known as 'keypoints' with a similar approach as the HOG algorithm, with the main difference and advantage that the features extracted are not affected by the size or orientation of the image, in other words, they are scale and rotation invariant, which is its main advantage. However, the mathematical computations are complex, which makes this algorithm computationally heavy and ineffective in low-powered devices [15].

As previously mentioned, these feature extraction methods require a secondary step known as a classification method and the most popular is Support Machine Vector (SVM), a supervised learning algorithm used for automated object detection and characterization.

It classifies data into different classes by creating a decision boundary (hyperplane) between any two classes to separate them and classify the object [16].

Although these technologies still provide accurate results and have several advantages, like requiring less data and less computing power, with the ongoing development of deep learning methods, machine learning has become an obsolete technology unable to solve complex AI problems and which require ongoing human intervention.

2.2.2 Deep learning methodology

As previously stated, deep learning is a subset of machine learning that uses artificial neural networks to mimic the human learning process and it can perform classification processes form images, text, or sound. It is the key technology behind many current products in constant development, like autonomous driving or voice control in-home devices [12].

Although deep learning is the most accurate in recognition tasks and was first theorized in the 1980s, it has two main limitations, which are the main reasons why this technology is now at its peak development and has recently become useful:

1. Requires a large database of labelled data.
2. Requires a lot of computing power since it performs non-linear and complex correlations; therefore, it needs high-performance GPUs.

The procedure followed to train deep learning models first uses large databases of labelled data and neural network architectures that allow the learning of features from the images without the need to manually extract such features, as was required in machine algorithms.

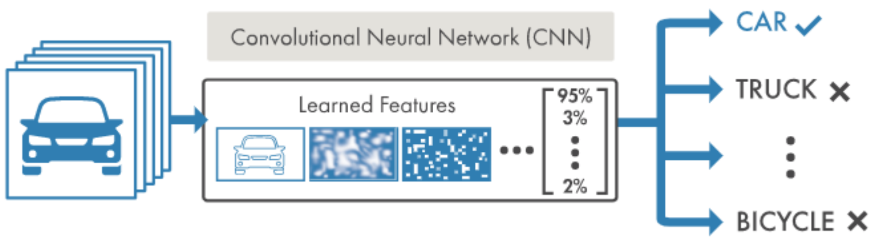


Figure 2.2 Deep learning general procedure [12]

When referring to the use of deep learning techniques for object detection, we can divide the possible methodology into two groups, according to the procedure followed in each of them: two-stage algorithms, in which we can find Region-based Convolutional Neural

Network (R-CNN) and its variations, and one-stage algorithms, which include You Only Look Once (YOLO) family and Single Shot Detector (SSD) algorithms.

In two-stage detectors, first, it's needed an object region proposal method followed by the object classification based on the features extracted from such regions of interest. The results usually have a high accuracy; however, having a two-step process means they are typically slower than other methods [17].

R-CNN is an object detection algorithm whose key concept is region proposals, used to locate objects within an image. Therefore, we first need to extract a region of interest or region proposal using a chosen algorithm and resize the extracted regions, before passing the image through the neural network to classify the object in each region[18][19].

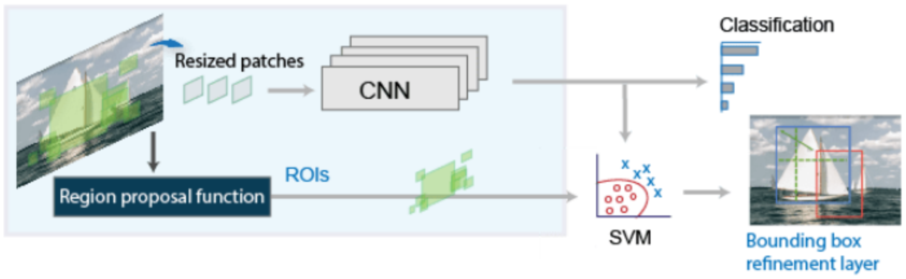


Figure 2.3 R-CNN general procedure [19]

There are different variants of R-CNN, each of them attempting to improve, optimize or speed up the result of the processes within the algorithms.

Fast R-CNN also uses an algorithm to extract region proposals, but instead of processing such regions by cropping or resizing them, it processes the whole image. In R-CNN, each region must be classified individually, whereas Fast R-CNN gathers the features extracted from each region, creating a convolutional feature map. Since the algorithm isn't fed with all the region proposals but only the feature map, it results in a faster and more efficient detection [19][20].

Faster R-CNN replaces the use of an external algorithm with a region proposal network (RPN) to generate region proposals directly in the network. Eliminating the additional algorithm results in a faster and better performance process [19][20].

On the other hand, we have the YOLO family and SSD, one-stage object detection algorithms. The main difference with R-CNN algorithms is that there are no region proposals extractions: it predicts and classifies objects in the image directly. This is the reason why they are often faster, which makes them suitable for real-time applications. However, they also have a main disadvantage, usually unable or having difficulties to recognise irregular shaped objects or a group of small objects [17].

YOLO is based on a convolutional neural network that predicts not only the bounding boxes of the objects, but also the class probabilities for all the objects found in the image [21]. The image is divided into grid cells and each grid predicts bounding boxes and the parameters shown in figure 2.4 for each of the bounding boxes created are calculated, where p_c represents the probability of an object being within the grid, (b_x, b_y, b_w, b_h) stand for the center of the bounding box as well as its width and height and $p(c_i)$ represents the probability of the object belonging to the i^{th} class for the given p_c, where n is the number of classes.

p_c	b_x	b_y	b_w	b_h	$p(c_1)$	$p(c_2)$	...	...	...	...	...	...	...	$p(c_n)$

Figure 2.4 YOLO's bounding box parameters [21]

Furthermore, the confidence score value (c_s) is computed for each bounding box, which reflects how likely the box contains an object and how accurate is the bounding box. Next, class-specific scores (c_{ss}) are computed for each bounding box which reflects the probability of the object belonging to a specific class and how accurately the box encloses the object. Finally, after disregarding some boxes with the help of a threshold set on the confidence score values, non-max suppression is applied to discard less relevant bounding boxes from all those overlapping, selecting the one with higher correlation or similarity.

From the initial YOLO algorithm, several variants were developed, introducing different changes or innovations in each of them:

1. YOLOv2 introduces multi-scale training, which allows the algorithm to detect objects in images with varying input sizes, which results in better accuracy and higher speed [22].
2. YOLOv3 introduces a new network architecture (Darknet-53), bigger, more accurate and faster than previous versions [23].
3. YOLOv4 introduces several novel techniques that improve the CNN accuracy and speed, focusing on the introduction of universal features [24].
4. YOLOv5, developed by Ultralytics, eliminates Darknet's limitations (based on C language) by being implemented in PyTorch, which made it easier for developers to implement different architectures.

The latest releases in the YOLO algorithms family are YOLOR, YOLOv7 and YOLOv8 released in 2021, 2022 and 2023 respectively. YOLOR stands for You Only Learn One Representation and it is proposed as a combination of implicit (based on past experience) and explicit knowledge (based on given data) in a unified network. This new architecture is implemented in three steps: kernel space alignment, prediction refinement and a CNN with multi-tasking learning [25]. YOLOv7 surpasses all previous YOLO algorithms as well as most other object detection methods, improving in both speed and accuracy. It also implies the use of cheaper equipment, faster training and with smaller databases, which is a usual limitation or challenge when using deep learning methodology. It provides a stronger network architecture with a more effective method for feature integration and introduces what is called a trainable Bag of Freebies which increases the accuracy without producing losses in speed in real-time object detection [26].

Lastly, YOLOv8 is the newest state-of-the-art YOLO algorithm, developed by Ultralytics (who also developed YOLOv5), and is considered the highest-performing model and one of the easiest YOLO models to train and deploy in different platforms, from CPUs to GPUs [27]. Furthermore, it supports all previous YOLO versions, making it a flexible solution; it has a high accuracy; provides different developer-convenience features, from a command-line interface (CLI) to a Python package; and there already exists an extensive community around this newly released model, providing access to different resources in case of needed guidance [28], [29].

Finally, also a one-stage object detection algorithm, SSD is a network that merges detections predicted from multiscale features. First, it runs a deep learning convolutional neural network (CNN) on the image used as input to produce network predictions from several feature maps, and then, the algorithm uses such predictions to generate the resulting bounding boxes [30].

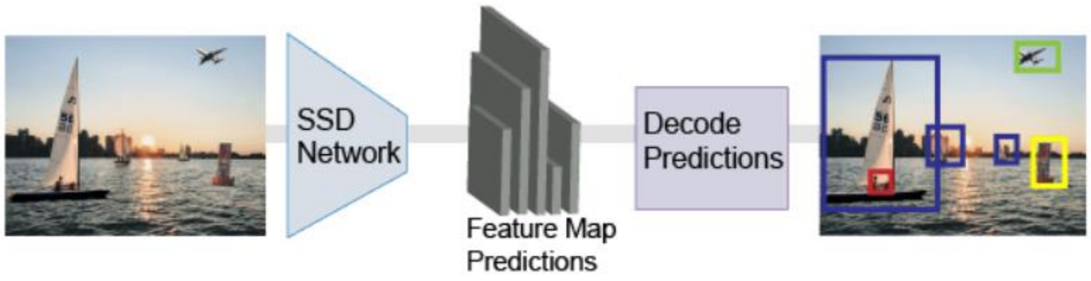


Figure 2.5 SSD general procedure [31]

2.2.3 Evaluation metrics and benchmarking

There are several metrics that help to evaluate the performance of the developed object detection algorithm models, with the goal of selecting the one with the most appropriate features for the particular case required.

Among the most common metrics for evaluation, with which the speed and accuracy can be assessed, we can find Frame Per Second (FPS) and mean Average Precision (mAP). FPS expresses how fast an algorithm or model is in processing an input video and generating the desired output. On the other hand, the most common metric used is mAP and in order to understand how it assesses the chosen approach, a few terms or helper metrics have to be previously defined [32], [33]:

- True Positive (TP) is a correct detection of the annotated bounding box.
- False Positive (FP) is an incorrect detection of the annotated bounding box.
- False Negative (FN) is an annotated bounding box missed or not detected.
- True Negative (TN) is a detection of an incorrect bounding box or negative class, though is not used as an object detection metric.
- Intersection over Union (IoU) represents the degree of overlap between the annotated bounding box (ground truth) and the prediction made, showing the accuracy of the predicted bounding boxes.

From these terms, firstly, we can obtain the precision, defined as the capability of the model to identify only significant objects (percentage of predictions that are correct over all detections made), and is calculated using the following equation [33]:

$$P = TP/(TP + FP) \tag{2.1}$$

where *P* – precision,
TP – true positives,
FP – false positives.

Secondly, the recall is defined as the capability of the model to detect all annotated bounding boxes or ground truths, and is calculated using the following equation [33]:

$$R = TP/(TP + FN) \tag{2.2}$$

where *R* – recall,
TP – true positive,
FN – false negative.

Finally, the Average Precision (AP) is calculated using both the precision and recall and is defined as the area under the precision-recall curve (PRC). Consequently, the mean Average Precision (mAP) is computed using all the values of the AP for the different classes as it's shown in the equation number 2.3, and it represents the model's accuracy to detect all the classes [33]. Usually, this parameter is evaluated over IoU thresholds, having two usual metrics, mAP 0,5, with an IoU of 50% and mAP 0,5:0,95 with an IoU between 50% and 95%.

$$mAP = \frac{1}{N}\sum_{i=0}^{N} AP_i \tag{2.3}$$

where mAP – mean average precision,

N – number of classes,

AP_i – average precision for each i^th class.

A benchmark is used to assess how an object detection model performs and evaluate the results for the metrics previously explained. The process is named benchmarking and it is defined as the process of evaluating the performance of a product or process by comparing it against those considered to be the best in the industry. In order to do so, there are several benchmark datasets already available, that contain a large variety of labelled images, with which we can train our model and analyse the results to choose the most accurate or appropriate algorithm for the needed use. The most popular and widely used datasets for benchmarking object detection algorithms are Microsoft Common Object in Context (MS COCO), PASCAL Visual Object Classes (PASCAL VOC), and ImageNet.

MS COCO is the most used dataset containing over 300 thousand images from everyday scenes, with over 90 different types of objects or classes and with a total of 2,5 million labelled instances. The dataset is available to be explored through an online interface or to be downloaded and used in the platform of the user's preference [34], [35].

PASCAL VOC contains 20 classes of different object categories and each image on the dataset has pixel-level segmentation, object class annotations and bounding box annotations. The dataset is already divided into three sections: training, validation, and testing sets [36].

Lastly, ImageNet is the largest out of the mentioned benchmarking datasets, with over 14 million images. The dataset was publicly released, dividing its contents into manually labelled training images and testing images without any annotations on them [37].

Using the aforementioned benchmarks and observing the values obtained for the evaluation metrics, a comparison between the most relevant deep learning methods for object detection is performed. The methods compared are divided into those capable of providing a real-time application, with an FPS value higher than 20, and those with lower speeds that don't allow to perform inference on real-time video input.

First, the R-CNN family is compared as deep learning methods with low speed, not able to provide real-time applications. They are assessed on the PASCAL VOC benchmark, with different versions from 2007, 2010 and 2012, and the results can be observed in figure 2.6 below:

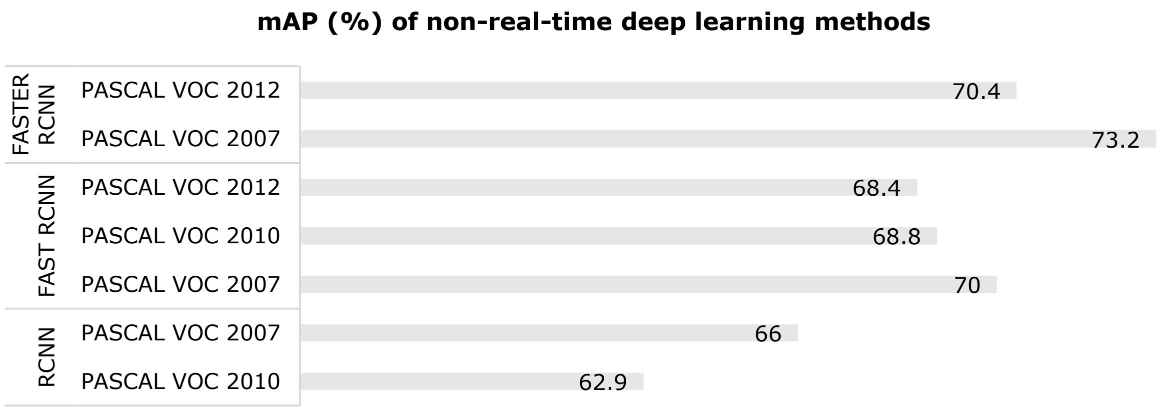


Figure 2.6 Comparison of R-CNN family algorithms (non-real-time deep learning methods) [33]

On the other hand, with an FPS value high enough as to be considered suitable for real-time applications, SSD and different versions of the YOLO family are compared on the benchmark MS COCO. As it can be observed in figure 2.7, the values of the mAP significantly decrease when dealing with real-time performances.

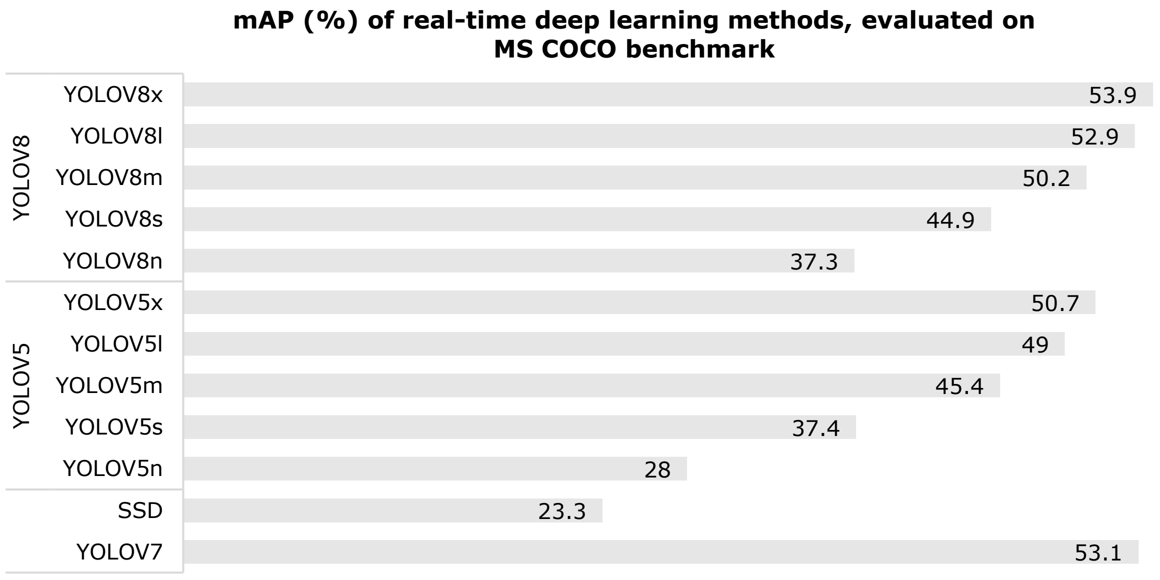


Figure 2.7 Comparison of real-time deep learning methods [26], [33], [38], [39]

2.3 Fiducial marker detection

Fiducial markers are 2D artificial landmarks used to extract pose estimation of an object. Some examples of fiducial markers can be observed in figure 2.6 below:

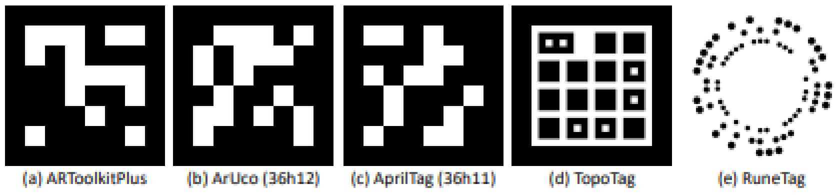


Figure 2.8 Examples of fiducial markers [40]

A fiducial marker system usually is composed of the markers, a detection algorithm, and a coding system. Since the color and shape of the used markers are defined beforehand and stored in a library, fiducial markers can be easily and stably extracted from an input image or video [40][41]. The most used fiducial markers are square-based, whose main advantage is that the corners of the markers can be easily used to extract the pose and location of the object, while the inner region is used for identification. This inner region can be a binary code or an arbitrary pattern [42].

ARToolKit is an open-source project based on square-based fiducial markers with an arbitrary pattern inside, which has been extensively used. The markers have a black border, and it is the inner region that is stored in the database of the system. Its main drawback is that by using template matching approaches for identification of the inner region, the amount of false positives obtained is very high and it’s very sensitive to lighting conditions when detecting the outer squares [42].

ArUco markers are one of the most popular and most used fiducial markers nowadays. ArUco markers are square-based fiducial markers and each of them is formed by a seven-by-seven binary grid, each cell called a bit, and a collection of markers is defined as a dictionary of ArUco markers. There are several open-source implementations of ArUco marker detection on OpenCV and Python [43], and it can also be implemented in combination with a deep learning method using a convolutional neural network in order to acquire a more reliable and accurate system [44].

2.4 Face recognition technologies

Face recognition is a subset of object recognition, which consists of, through artificial intelligence, processing the input face image, extracting the facial features and comparing the results with the existing face database to obtain an identification of the individual [45]. Since face recognition technologies began in the 1960s, it's been in constant development and their procedure has varied from being based on face structure features (1970-1990), to statistical features (1991-2000) to the current and developing technology based on big data and complex algorithms (2001-present). It has extensive use in daily applications, mostly related to biometric identification technologies, in numerous fields such as politics, military, economy and culture [46].

All face recognition systems involve three key steps or stages: face detection, feature extraction and identification. Sometimes these tasks aren't completely separated from each other, performing several tasks at once, but all of them have to be present to successfully achieve a working system [47].

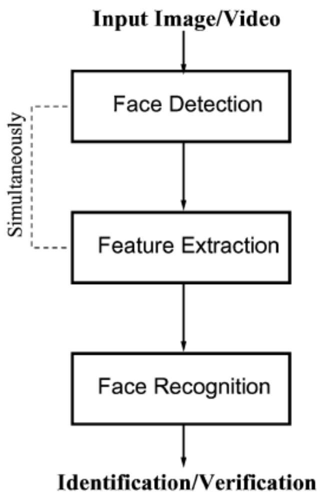


Figure 2.9 Face recognition stages [47]

Furthermore, there is a very important aspect to be taken into account when developing a face recognition system, which is the security and user privacy, needed in any biometric software implemented. According to the General Data Protection Regulation (GDPR) that came into effect in 2018, the processing of biometric data such as face images for identification purposes is prohibited unless there exists an explicit consent of the relevant person. Therefore, face images are classified as sensitive data and need to be protected to ensure the protection of privacy [48].

Among the most relevant and recent technologies developed in this field, we can find Amazon Rekognition. It was first released in 2016 as a cloud-based software as service computer vision platform capable of recognising both objects and faces using deep learning. It provides a series of pre-trained algorithms on data collected by Amazon, such as celebrity recognition, facial attribute detection, text detection and classification in images, among others. It also allows user to train their own pre-labelled data set for face search and verification [49]. Due to the large amount of data amazon has access to, the algorithms are able to correctly learn and identify the required object or person, however, if a user desires to create a customized recognition system importing and labelling numerous images is required, which is a disadvantage of this product as its price depends on its use and therefore on the number of images to be processed.

DeepFace is an open-sourced face recognition and facial attribute analysis library for Python, released in 2014 by Facebook. They developed a deep learning system, creating a CNN trained on 4 M images of over 4000 individuals, that approaches human performance in face recognition [50][51].

These systems are state-of-the-art in face recognition methodology; however, as stated before, the resources needed in both datasets and computational power are very extensive, reason why they are often unattainable to smaller companies or smaller projects. On the other hand, there exist numerous projects [52], [53] using the same principles, such as convolutional neural networks, implemented in accessible software like OpenCV and Python to implement and train a face recognition system with a limited database of images and using more reachable hardware [54].

The main method for testing the effectiveness and performance of the implemented facial recognition system is by means of benchmarking. For that goal, in the specific case of face recognition, there are several popular training datasets with which the system can be trained in order to compare the results with other implementations. The most used datasets in the last four years are LFW (Labeled Faces in the Wild) and VGGFace2. LFW contains over 13000 images of faces collected from the web, forming over 5000 identities with almost 2000 people with two or more images in the dataset [55]. VGGFace2 is formed by around 3,3 million images, divided into classes, representing over 9000 identities. The dataset is already divided into data for training and testing [56].

2.5 Influencing factors in machine vision technology

When developing a machine vision-based system, there are several factors that must be taken into consideration, and that might affect the end results of the implemented system. These factors can come from the specific component being used in the system, like the camera, or from environmental factors in the location, that may or may not be modified. Logically, these factors are divers and might affect the overall system differently whether its location is outdoor or indoor. In order to reflect in this chapter the information relevant to the system to be implemented, only those factors that affect a system in an indoor location are specified. In figure 2.10, the factors later explained are summarized:

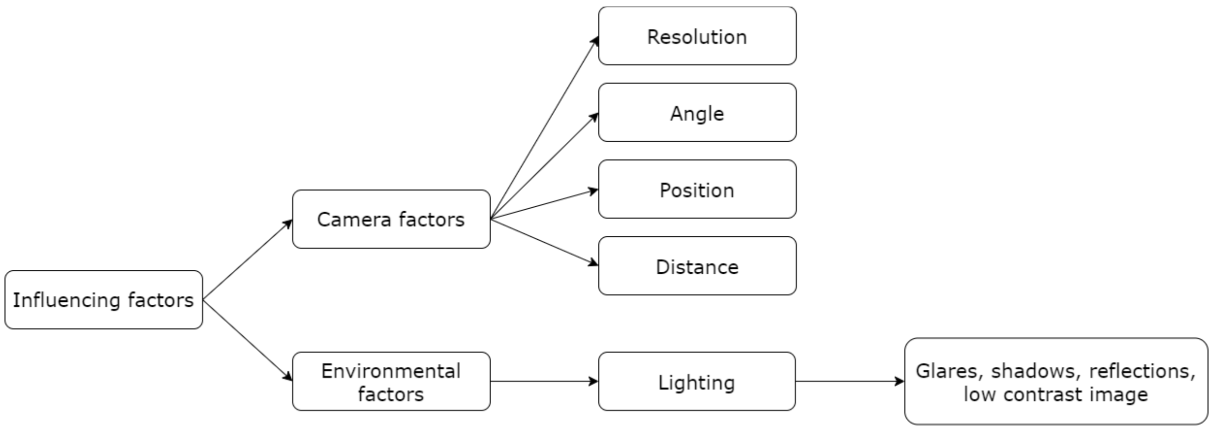


Figure 2.10 Influencing factors in machine vision technology

First, regarding those factors that may arise from the specific selection of the camera used and its location within the system:

- Resolution: this specific characteristic of the camera used may affect the detection and recognition of different items, depending on their type, size and features.
- Angle: the angle at which the camera is placed facing the items that have to be detected or recognised can influence enormously in the outcoming result. At an incorrect angle, the items' shape may be distorted, making it very difficult for the algorithm to recognise them.
- Position: the location of the camera must also be taken into account. An optimal position must be found in order to avoid obstructions from other items in the environment.
- Distance: the placement of the camera at an adequate distance from the detected items is fundamental. If the items are too far from the camera, the colour or details might be faded, making it harder to be recognised. In the opposite case, if the camera is too close, the items might be too large to fit on the frame, and if there

are several items, some of them might be undetected. The whole area in which the items are placed must be within the frame.

On the other hand, the main external factor that can affect the performance of the implemented system is lighting [57], [58]. Lighting is one of the most critical aspects of a machine vision-based system. Inadequate lighting can lead to glares, reflections, shadows and/or low contrast images, which cause problems like the saturation of the camera, undetected features, and distortion inferences (items might appear of different sizes or colour to the camera). Most of these effects can be corrected by the use of a proper source of lighting, and even the position of such source can lead to a better result. Furthermore, changes in the environment, like the background of the setup, can influence and reduce such effects.

However, although the changes in these factors can lead to better results, sometimes the situation doesn't allow for more or any modifications. In that case, image post-processing might be needed, through which the negative effects can be corrected before performing the detection or recognition of the items, facilitating the task for the algorithm.

2.6 Conclusion on literature review

Inventory management systems are an indispensable part of industrial manufacturing facilities and warehouses, that add reliability, security, and efficiency. The methodology or technologies used for their implementation are very extensive. Large companies with great development potential often use combinations of several technologies like artificial intelligence, sensory systems, computer vision methods, Radio-Frequency Identification (RFID) technology or computer vision with deep learning. Such implementations result in complex and expensive systems, unattainable for smaller companies.

However, the functionalities implemented in those complex systems can be achieved by the use of only vison-based approaches. Such an implementation would result in a simpler and cheaper system, as we would eliminate the need for any additional hardware, relying only on a multi-camera system.

As the literature is organized, the system is divided into three different sections, each one of them with a different goal and oriented to a certain type of recognition:

- For objects that can vary in position and orientation, as well as other set up parameters like lighting or background, the implementation of an object detection algorithm based on deep learning methodology is the best approach. This implementation allows the system to work on the detection in real-time, allowing variances in the placement and orientation of the required object to be detected. The specific algorithm is to be chosen, depending on the precision and accuracy needed as well as the speed, a very important characteristic when dealing with real-time systems.

- When there exist further limitations on the objects to be recognised, like very similar appearance or difficult visibility of the entire item, the use of fiducial marker detection gives you the advantage of a simpler recognition in spite of the limitations. ArUco markers are the most popular option nowadays for such implementations, allowing the creation of as many markers as needed and the following recognition by the creation of a library that contains them.
- Face recognition is probably the most complicated choice as it's a developing technology, with new advances being discovered daily and there's an important limitation to be taken into account, which is the limited dataset that the system has to be implemented with. Due to the complexity of face recognition, deep learning solutions tend to show better results.

The validation and evaluation of the system implemented will be done with real-time camera input to test the system in real life and benchmarking for the object and face recognition implementations, with web-available datasets, typically used in these developments.

3 DESIGN CONCEPT

The implemented system consists of an inventory management application that keeps track of the available material, constantly updating a registry list that informs for each of the items present in the inventory system, if they have been borrowed or they are available, and in the first case, the user that borrowed them. The material available consists of two different groups: the lab equipment and the camera kits, each of them with a tag displaying a unique fiducial marker.

With the use of two different external cameras, one dedicated to facial recognition of users and the other for the detection of available items from both sets of material, the general logic of the system consists of the following: while a user is detected, the system stores the name or ID of that user, and the inventory is not updated until the facial detection is negative, giving the necessary time for the user to make any necessary changes or returns. Once the user is out of the camera frame, the inventory starts to update, registering any changes in the available material and assigning the corresponding user as the borrower. Furthermore, at the initialization of the system, the user is asked to choose one from the two available functionalities:

1. Registration of a new user into the inventory management application.
2. Visualization of the inventory information (normal functioning of the system).

In figure 3.1, a rough outline of the design of the implemented system can be observed:

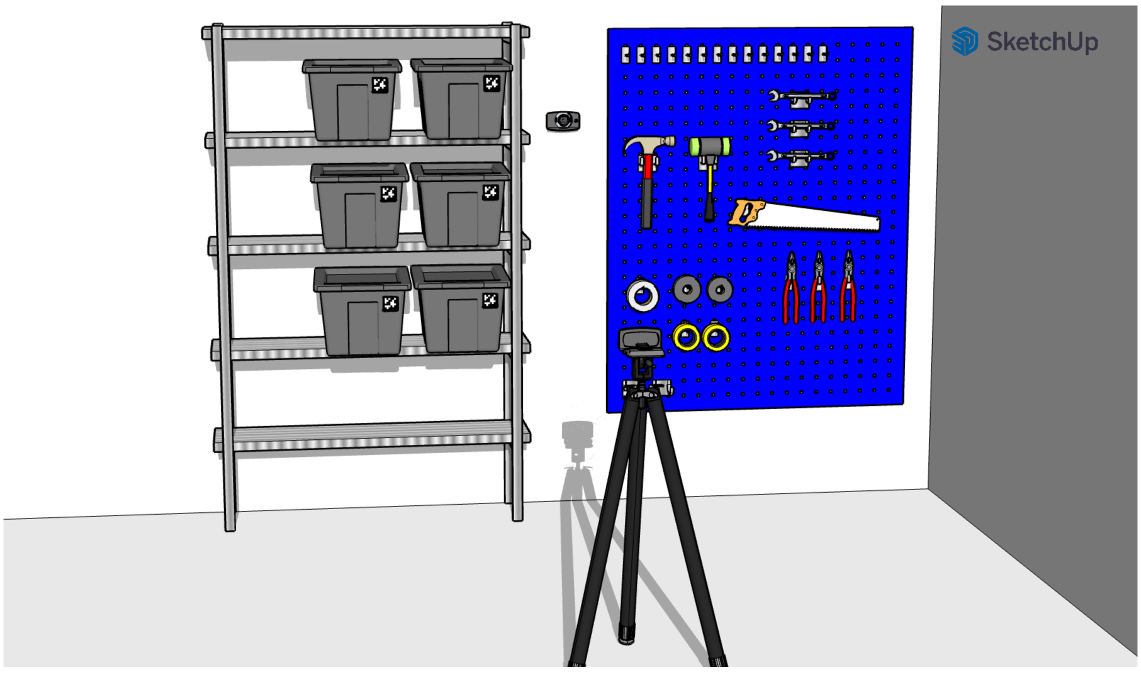


Figure 3.1 Sketch of the design of the system, implemented using SketchUp

3.1 Hardware set up

There are different hardware components involved in the development of the project and, as it was mentioned in the introduction, the approach to this project appeared as a response to the demand from the department to have a system to control the available materials and, therefore, most of the elements used were already available and there was no need for any purchases. The components are divided into camera and computer equipment used for the implementation of the detection methodology, and the elements to be detected, which together constitute the inventory of the department to be managed by the system.

3.1.1 Camera equipment

Due to the objective of implementing the system in the simplest and most accessible way for all types of companies, it was decided to use as video input for both the detection of tools and boxes and for the recognition of users the Logitech C615 model, a portable external webcam with autofocus that is connected via USB. This module has a frame rate of 30 FPS with a resolution up to 1920 x 1080 pixels, considered high definition, a diagonal field of view of 78º and autofocus and auto light correction features, that allow for the procurement of better-quality images even if the environment conditions are slightly modified due to external variations.

Figure 3.2 Logitech C615 webcam [59]

The camera used for object detection is placed on a tripod of the model PrimaPhoto PHTRBBK Gear, in which each leg uses a twist lock configuration to extend its length, providing a maximum height of 140 cm. The camera and tripod assembly with its dimensions and the field of view at the distance at which it is placed from the inventory equipment can be seen in figure 3.3.

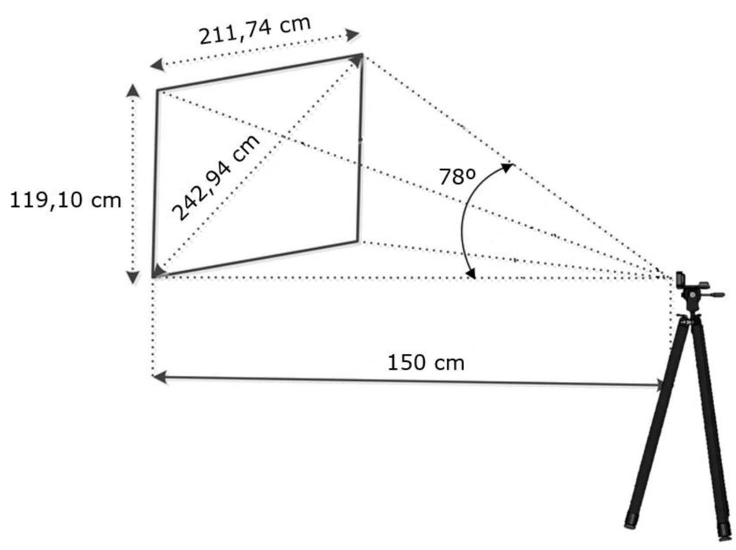


Figure 3.3 Field of View (FOV) of the camera assembly

The second camera, whose purpose is the face recognition of the users, is placed between the shelf containing the camera equipment boxes and the metallic rack in which the tools are placed. It is placed at a height of 190 cm, just above the height of the tool rack. This means that the camera is tilted slightly downwards to focus on approaching users, an action that is made possible by the fold-and-go design of the Logitech C615 webcam, which allows 360-degree rotation to position the camera at the best angle and different inclinations.

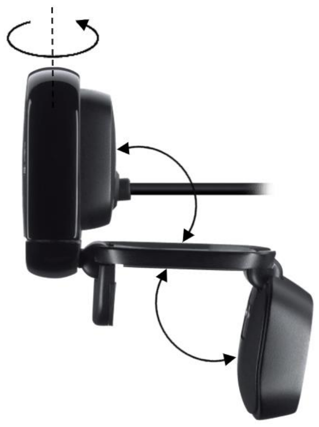


Figure 3.4 Logitech C615 webcam movement design

The communication between the cameras and the computer device in which the system is running is done via USB-type connectors, with the help of an extension cord for easy reach from the position in which they are both placed.

3.1.2 Computer equipment

Two different devices were used in the development of the system. A portable computer that can be easily positioned in different locations, making the system more flexible according to the structure of the environment or if the usual place of its position is temporarily occupied by another element. A secondary desktop computer with a GPU was initially intended to be used for the training of the YOLOv8 object detection model, since the training of object detection algorithms is a computationally intensive task and, therefore, can take a long time to be executed on a computer with only a traditional CPU. With the use of a GPU, the advantage of parallel processing can accelerate the training process and the ability to select a larger batch size results in an improvement of the accuracy of the model. However, the secondary computer had a GPU NVIDIA Quadro P4000, which is designed for professional graphics and visualization, not for machine learning and AI processes and therefore did not result in such a considerable improvement in training time. On the other hand, Google Colab, a cloud-based platform for running and developing machine learning models, provides with a Tesla T4 GPU, which is designed primarily for machine learning applications and, therefore, a more suitable choice for algorithm training. In the following table 3.1, the specifications of each device are listed and the resources available in Google Collab are listed:

Table 3.1 Computer equipment specifications

	Processor	**RAM**	**GPU**
Portable computer	Intel(R) Core(TM) i7-8550U CPU @ 1,80GHz 1,99 GHz	8,00 GB	-
Desktop computer	Intel(R) Xeon(R) W-2123 CPU @ 3,60GHz 3,6 GHz	32,00 GB	NVIDIA Quadro P4000
Google Colab	-	32,00 GB	Tesla T4

3.1.3 Inventory elements

The last elements of the hardware set up are those objects that form part of the inventory that the system is managing. These are divided into two groups: the tools and the camera equipment, which is stored inside boxes.

Among the tools can be found wrenches, hammers, pliers, different colour tapes, etc. All tools are placed in a metallic blue rack placed on the wall, using adjustable hooks that can be moved within the rack to place the tools at the desired location. Figure 3.5 shows a possible arrangement of the tools within the rack and its dimensions.

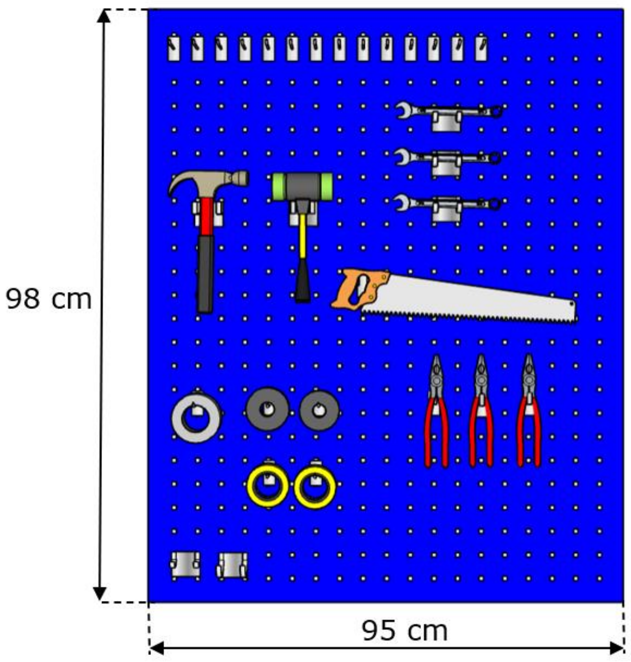


Figure 3.5 Possible arrangement of the tools (implemented using SketchUp)

On the other hand, the camera equipment available in the laboratory is stored in black plastic boxes of similar appearance. The ArUco tags are attached to the top of the box, where the opening system is located, since they are placed horizontally on their side on the shelf, as it is shown in figure 3.6.

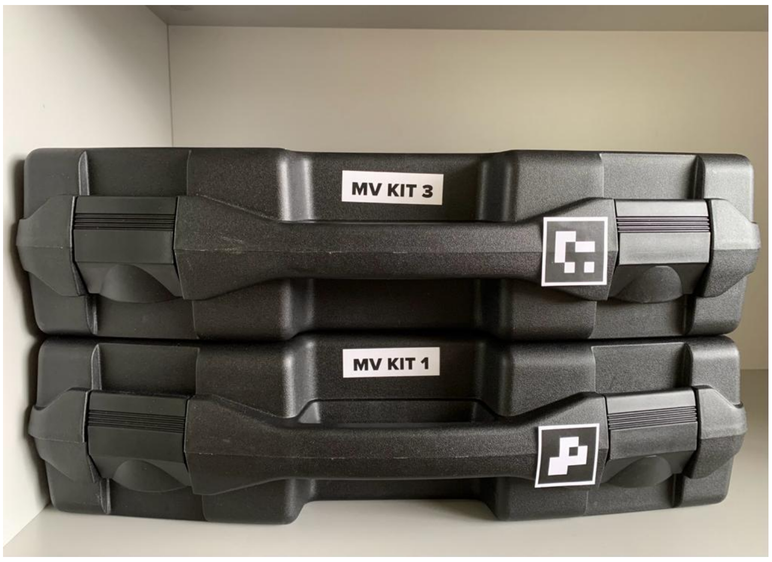


Figure 3.6 Camera equipment with ArUco tags

3.2 Software architecture

The development of the software architecture of the system implemented is divided into three different sections that correspond to the three types of detections, which are then combined into one program, in which the overall functioning of the inventory management system is achieved.

3.2.1 Object detection (tools)

The tools available in the laboratory are items that can vary in position and orientation, therefore, the optimal method for their detection is using an object detection algorithm based on deep learning. As stated in chapter 2, the YOLO family algorithms are among the most innovative with promising results, proven by renowned benchmarks such as the COCO dataset. The newest state-of-the-art YOLO model and the chosen method for the detection of tools, is YOLOv8, developed by Ultralytics and launched in January of 2023.

YOLOv8 counts with five different versions of different sizes which make each of them optimal for specific applications. These are YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l and YOLOv8x, sorted from smallest to largest. As it is shown in figure 3.7, the precision and accuracy of the versions increases as the size of the model increases; however, at the same time, the speed of each version decreases. The larger models take longer times for both training and inference speeds.

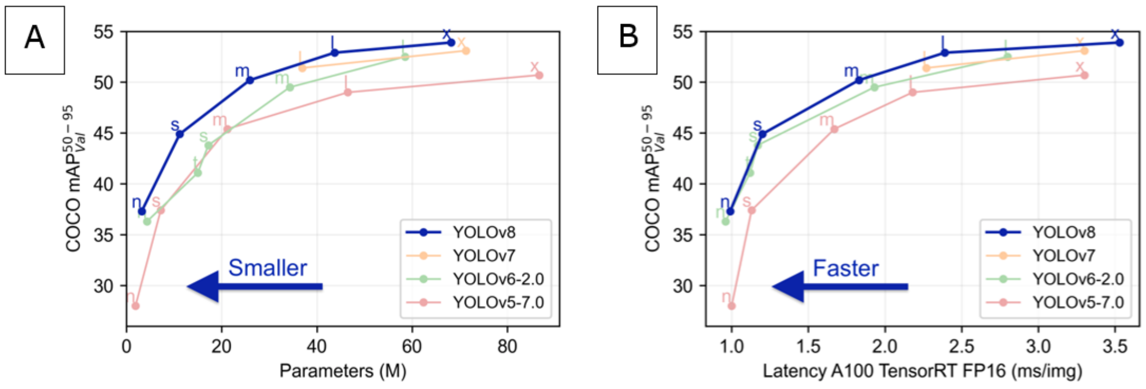


Figure 3.7 Comparison of YOLOv8 versions: YOLOv8n (n), YOLOv8s (s), YOLOv8m (m), YOLOv8l (l) and YOLOv8x (x)
A – Size comparison (mAP vs. parameters); B – Speed comparison (mAP vs. latency) [39]

Therefore, for an application in which real-time detection of the objects is needed, the suitable versions would be YOLOv8n and YOLOv8s, corresponding to the nano and small sizes, and in this project, the small version is used. The process of preparing and training the algorithm on a custom dataset it's carried out in different steps, whose results will be later analysed in chapter 4.

Image dataset generation: the first step consists of the recollection of several images of each tool to be detected by the system, in different orientations and positions. In deep learning object detection methods, a larger number of images guarantees better results. However, since the system is aimed to be developed with limited resources, the number of images per object is set to 15-20 plus a few additional ones of the metallic rack with all the available tools placed on it. With 10 different tools, the total number of images in the dataset is 226. In figure 3.8, a few examples of images from the dataset, of a specific tool and of the full rack can be observed:

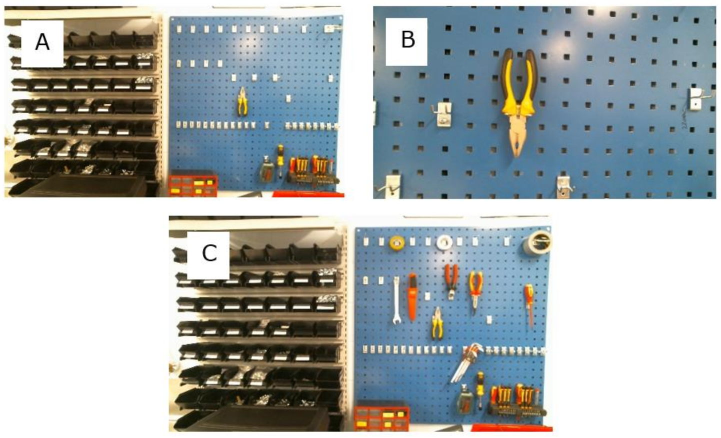


Figure 3.8 Dataset images example
A – Tool at system's set up distance; B – close-up tool; C – all tools at system's set up distance

Labelling of images: once the dataset is completed, the images have to labelled to indicate the presence of the objects of each class in the images. All objects have to be labelled, as precisely as possible. This step is performed with the help of the platform Roboflow, a cloud-based computer vision platform that allows users to manage datasets, train models and deploy them, among other features. Figure 3.9 shows a labelling example where all the tools are present in the metallic rack:

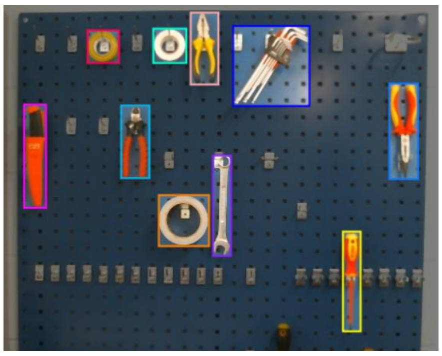


Figure 3.9 Example of labelling of images

Pre-processing: the first technique applied to the images it's its resizing to a smaller size, which results in smaller size files and, therefore, faster training. The images should maintain their aspect ratio, and since initially they have a size of 1920x1080 pixels, with an aspect ratio of 16/9, the final resize value is set to 640x360 pixels.

Data augmentation: is a technique used to increase the size and variety of an image dataset by the creation of new instances out of existing data samples. There are several techniques that can be performed, among which, the ones used for the tool's dataset are flip (horizontal, vertical), 90º rotation (clockwise, counterclockwise), brightness (between -30% and +30% for brighten and darken modifications), and blur (up to 1,5 pixels). The free version of the Roboflow platform supports the growth of the dataset up to 3 times its original size after applying data augmentation, with a final dataset size of 586 images. Figure 3.10 shows examples of brightness and blurriness data augmentation techniques:

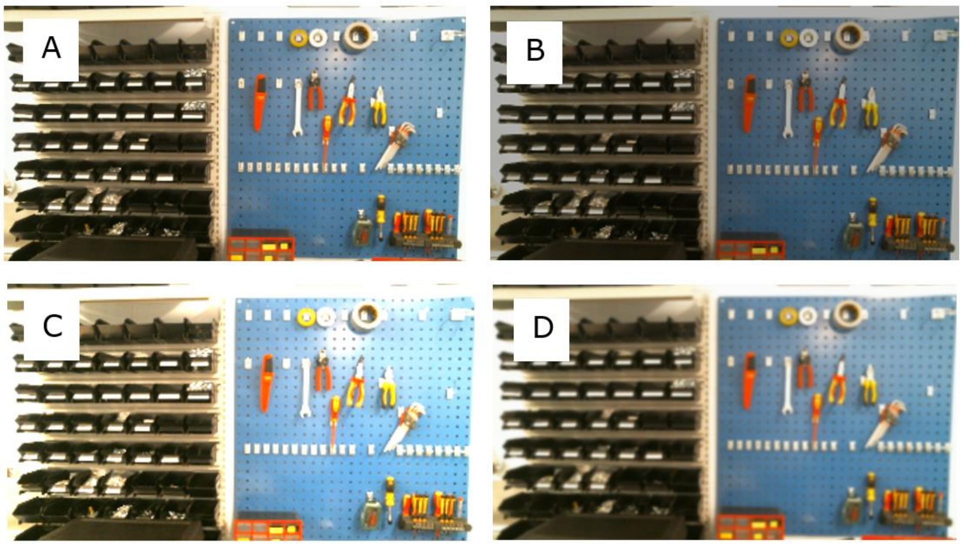


Figure 3.10 Examples of data augmentation techniques
A – original image; B – 30% darken image; C – 30% brighten image; D – 1.5 pixels blur

Dataset exportation: once the dataset is labelled and the data augmentation applied, it can be exported into different formats available in Roboflow. YOLOv8 format consists of a text file for each image that has an object (if no object is labelled in an image, there isn't a text file), that contains the class and bounding box data of the labelled objects. Furthermore, an additional data file is included that specifies the location of the training and validation data (both images and labels) and the number of classes and their corresponding names.

Training: the next step is the training of a YOLOv8 model on a custom dataset using the specified hyperparameters. This process involves adjusting those parameters to improve the model's ability to precisely identify the categories and positions of the objects within the frame, since the setting and tuning of the different values can affect the model's performance, speed, and accuracy. The most important hyperparameters, those that have been analysed for the optimal implementation of the system are:

- Model: it is the path to the model file, in this case, the small version YOLOv8s.
- Data: path to the data file that is included when the dataset is exported.
- Epochs: it is the number of iterations a dataset is trained for. If the number of iterations is too low, the model may not be able to learn the important features of the dataset and result in underfitting. On the other hand, if the number of epochs is too high, the model can develop the ability to memorize the training data and incorrectly perform on unseen data, resulting in overfitting.
- Batch: the batch size is the number of images that are processed in a single pass during training. A larger batch size means that more images are processed in parallel and, therefore, it can lead to faster training; however, it also requires more memory, which the system might not be able to provide, and may result in overfitting. The choice of the right size after experimenting with different batch sizes and evaluating the performance of the model can improve the stability of the model.
- Device: it is the device to run the training on, where it should be specified if a GPU, and which one, if there are multiple available, is being used during the training of the model.
- Patience: it is the number of epochs without any observable improvement to wait before early stopping the training of the model. This parameter helps to set the right number of epochs, since if the specified number of patience value is reached, it means that the number of epochs set is too high and might result in overfitting.

Validation: the validation data is a subset of the training data that is used to evaluate the performance of the model being trained and therefore, the correct election and tuning of the training hyperparameters. After each training epoch, the model's performance is assessed by using the validation data as unseen data for the model. The validation data must be independent of the training data and is obtained by dividing the final dataset version from Roboflow, after labelling and performing data augmentation. The division percentage depends on the number of images available, and as in this project the dataset is limited, the training and validation data is divided into 80% and 20% respectively, to have a bigger training dataset, resulting in 540 images for training and 46 for validation.

The training experimentation to set and tune the values of the hyperparameters, along with the result analysis and testing of the real-time detection of tools of the trained YOLOv8s model, are analysed in chapter 4.

3.2.2 ArUco marker detection (boxes)

The camera equipment available in the laboratory is stored in boxes of similar appearance and size, therefore making the process of their detection more challenging. The chosen methodology for this task is the detection of ArUco markers placed in each of the boxes as unique tags, each of them linked to the material available inside the container.

An ArUco marker is a square fiducial marker composed of a black border, that allows a fast detection of the marker within an image, and a binary matrix inside, whose codification allows the identification of each marker. The markers used for a specific application are called a dictionary of markers, which consists of a list of the ArUco markers and their codifications. There are two parameters to consider when choosing or generating a dictionary of markers: dictionary size, which is the number of markers in the dictionary, and marker size, which is the number of bits in each marker. When choosing the dictionary to use in the present application, both parameters were analysed as they influence the performance of the detection. First, as the maximum number of boxes in the inventory system is set to 10, the dictionary size was set to 50, which is the lowest number of markers present in a predefined dictionary. A choice of a dictionary with a higher number of markers would slow down the system since it has to find a match for each binary matrix detected from all the markers present in the dictionary. Secondly, the marker size was set to 4x4 bits, which along with the size of the image, set to 125 pixels, allows for the system to recognize the marker at the distance and resolution that the set up provides. In chapter 4, the different experiments performed in order to find the most suitable marker dictionary are shown. In figure 3.11, the markers printed and used for the identification of each box can be observed:

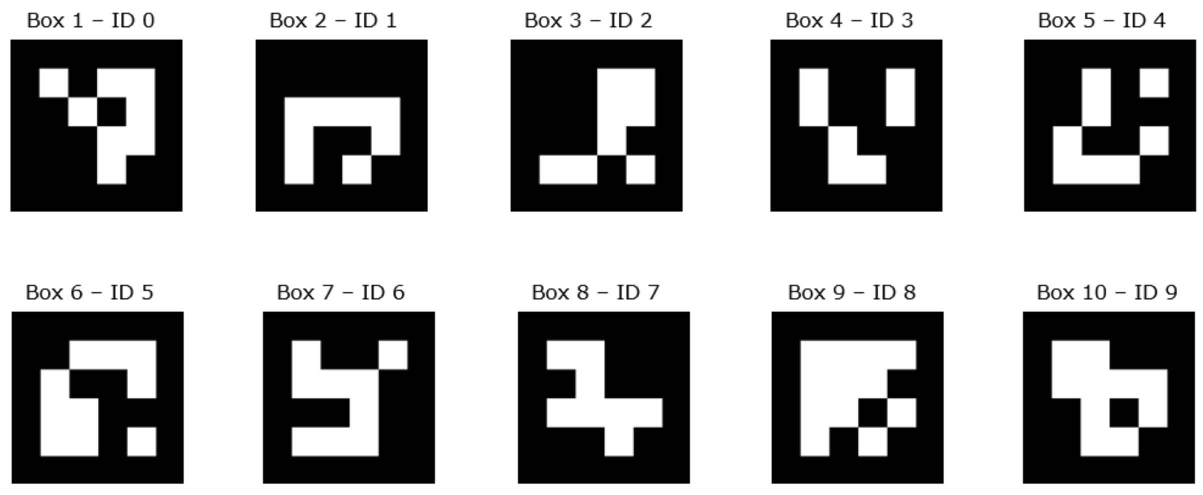


Figure 3.11 ArUco markers IDs linked to each box

The detection process of ArUco markers within an image return a list of detected markers, and for each of them, the position of its four corners in the image and the id of the marker, that allows its identification. This detection process is divided into two different steps:

1. Detection of potential markers: the image is analysed in order to find square shapes that have the potential to be an ArUco marker.
2. Detection of correct markers: by analysing the codification of the binary matrix, the real markers are selected.

3.2.3 Face recognition (users)

As it was mentioned in chapter 2, face recognition is one of the most challenging tasks, especially in view of the limited dataset condition, which makes it difficult to correctly train an object detection algorithm adapted to the users. Therefore, the methodology implemented is based on traditional computer vision techniques which require fewer resources for their development but still produce a satisfactory end result which fulfils the necessary tasks. The implementation of this section relies on the development of two main elements: a face detector and a face recognizer, that are used in the process as shown in figure 3.12:

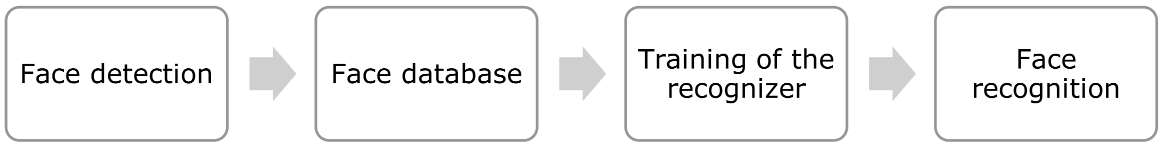


Figure 3.12 Steps in face recognition implementation

Both of the mentioned tools are available in the computer vision library OpenCV (Open Source Computer Vision). There are several algorithms or methods available in the library for both the detector and recognizer, and the selected methodology consists of face detection using Haar Cascades and face recognition using Local Binary Patterns Histograms.

Face detector

Haar feature-based cascade classifiers are an object detection method that can be adapted for face detection, which was first introduced by Paul Viola and Michael Jones in 2001. A Haar-like feature involves evaluating neighbouring rectangular regions within a detection window, where the intensities of pixels in each region are summed, and the difference between these sums is calculated. This value indicates a certain characteristic in that particular area of the image.

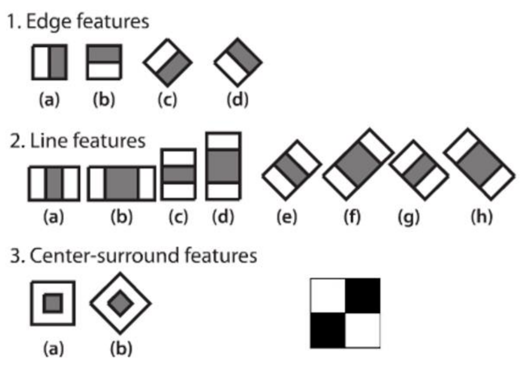


Figure 3.13 Haar-like features [60]

Then, the cascade classifier is trained on several positive and negative images and later used to detect the desired object in other images. OpenCV contains many pre-trained classifiers, among which we can find classifiers for faces, eyes, or smiles.

This approach is based on machine learning, where a cascade function is trained on several positive and negative images and later used to detect the desired object in other images.

Face recognizer

Local Binary Patterns Histograms (LBPH) is a feature extraction method that focuses on local regions of an image to avoid the high amount of input data and obtain a more robust recognizer. The main functioning of Local Binary Pattern (LBP) consists of summarizing the local structure of an image by labelling the pixels, examining the surrounding pixels and determining if they surpass a specific threshold. The resulting binary value is used as a label for that pixel. When it is combined with histograms, the images can be represented with a simple data vector.

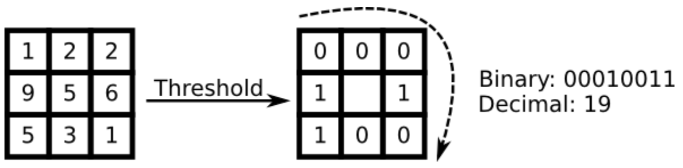


Figure 3.14 LBPH feature extraction method [61]

There are different steps taken in the use of the algorithm:

1. Set parameters. The LBPH employs four parameters, namely: radius, neighbours, grid X and grid Y. Radius is the distance from the central pixel used to create the circular local binary pattern. Neighbours refer to the number of sample points considered in such pattern. Grid X and grid Y represent the number of cells in the horizontal and vertical directions (the higher the number of cells, the finer the grid and therefore, a resulting feature vector with higher dimensions).

2. Training the algorithm. The next step consists of the collection of a dataset of the user that we want to recognize and the assignment of an identifier to each image (a number of a name). All images of the same person must have the same identifier.
3. Application of the LBPH operation. The binary value of the pixels is obtained for each of the images and the histograms are extracted, resulting in data vectors that represent the characteristics of the image.
4. Face recognition. For each input image or frame, the same process is applied to obtain an individual histogram which is later compared to the pre-trained algorithm in order to find the closer match, that returns the identification of the user.

3.2.4 Programme functioning

The implemented program is based on object-oriented programming and is written in Python. The set is formed by four different classes and a main program where the general logic of the system is developed. The classes correspond to the three types of detections already mentioned, corresponding to the tools, boxes, and users of the inventory system, and an additional supporting class. Each of the classes is next described, with the main attributes and methods needed for the correct understanding of the functioning of the program.

The **class for tool detection** has two main attributes corresponding to the YOLOv8 model loaded (already pre-trained) and an empty tools list that will contain the objects detected by the algorithm and used later in the main program. The only method that this class contains, obtains the detections made by the algorithm and filters them by only adding to the tools list the object with the highest confidence score of each class detected. Therefore, if multiple objects of the same class are detected, only the one with the highest probability of belonging to such class with be forwarded as a tool detected to the main program.

The **class for ArUco detection** has four important attributes: the ArUco dictionary used in the application, the parameters for the detection of the fiducial markers, a list corresponding to the boxes of camera equipment that each marker identifier is linked to, and an empty list where the detected boxes are stored for their later use in the main program. The class contains two methods, one for the detection of the markers in which the identifiers of the present markers within the frame are linked with the box they correspond to and stored in the list, and a secondary method that allows for the graphical visualization of the detection of the markers.

The **class for face recognition** has three main attributes: the Haar cascade classifier that acts as the face detector, already pre-trained with the frontal face feature detector available from OpenCV; the face recognizer based on Local Binary Patterns Histograms (LBPH) and a list with the user's names that are being used in the system, so that each face recognized identifier can be linked with the name of the user. The three methods correspond to the recollection of the user's images for the creation of a dataset, the training of the LBPH recognizer using the dataset previously created and finally, the face recognition method, in which using the pre-trained detector and recognizer, the system returns the name of the user present in the frame.

Finally, an additional **class for frame configuration** is created, that aims to set the frames that are used in each of the above detections: for face recognition, the whole frame is used (as the user can move along the whole field of view of the camera) while for tool and box detection, the input is divided into two different areas to avoid the unnecessary search for a certain type of objects where they cannot be found. Figure 3.15 shows the division of the frames for tools and boxes:

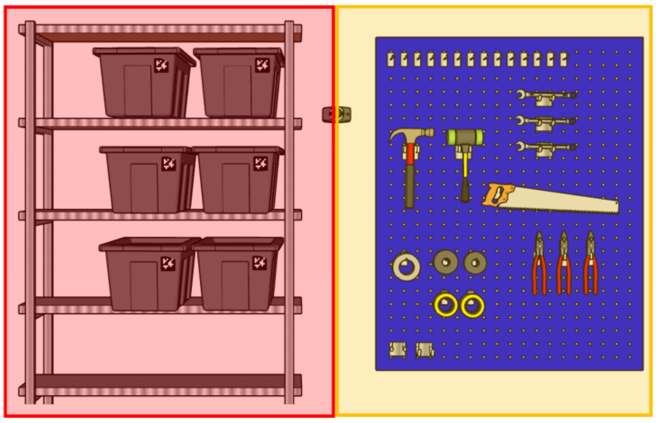


Figure 3.15 Division of frames

In figure 3.16, a summary of the different classes, their attributes and methods, can be observed:

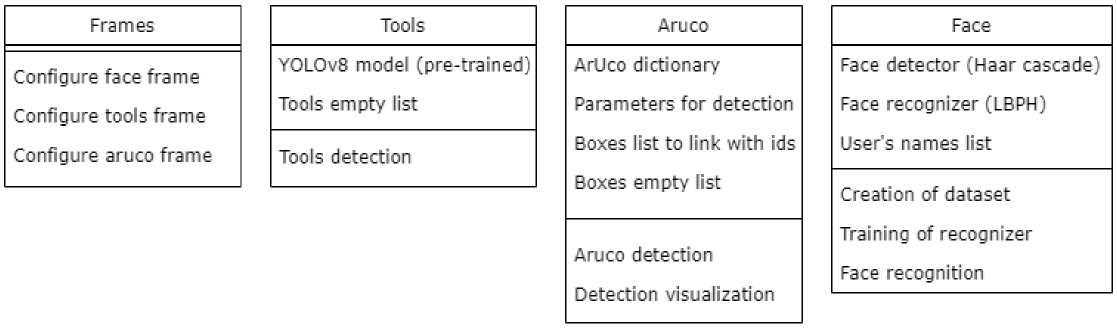


Figure 3.16 Summary of program's classes

The main program is first initialized by the creation of four different objects of the different classes previously explained, capturing the input video from both webcams, and creating an inventory dictionary variable. This dictionary has three different keys corresponding to the tools, boxes, and users of the system. For each of the tools and boxes, there are two variables that indicate the availability of the object within the inventory system and, if not available, the name of the user that borrowed the item. The third key corresponds to the users registered in the system and, for each of them, the items in their possession at the time.

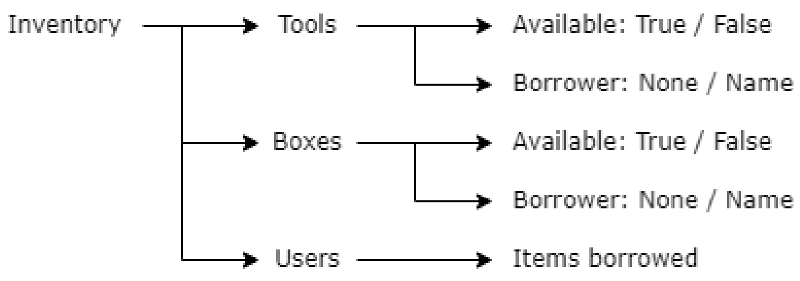


Figure 3.17 Structure of inventory dictionary variable

Then, the user is asked to choose between the functionalities available in the program, first one being the registration of a new user into the inventory management system, second one, the normal functioning of the system that allows obtaining the information about the items available or not, and third one, normal functioning to obtain the information and the additional visualization of the detections or recognitions. In figure 3.18, the flowchart or this first section of the program is shown, including the steps involved when choosing the first functionality, the addition of a new user to the system:

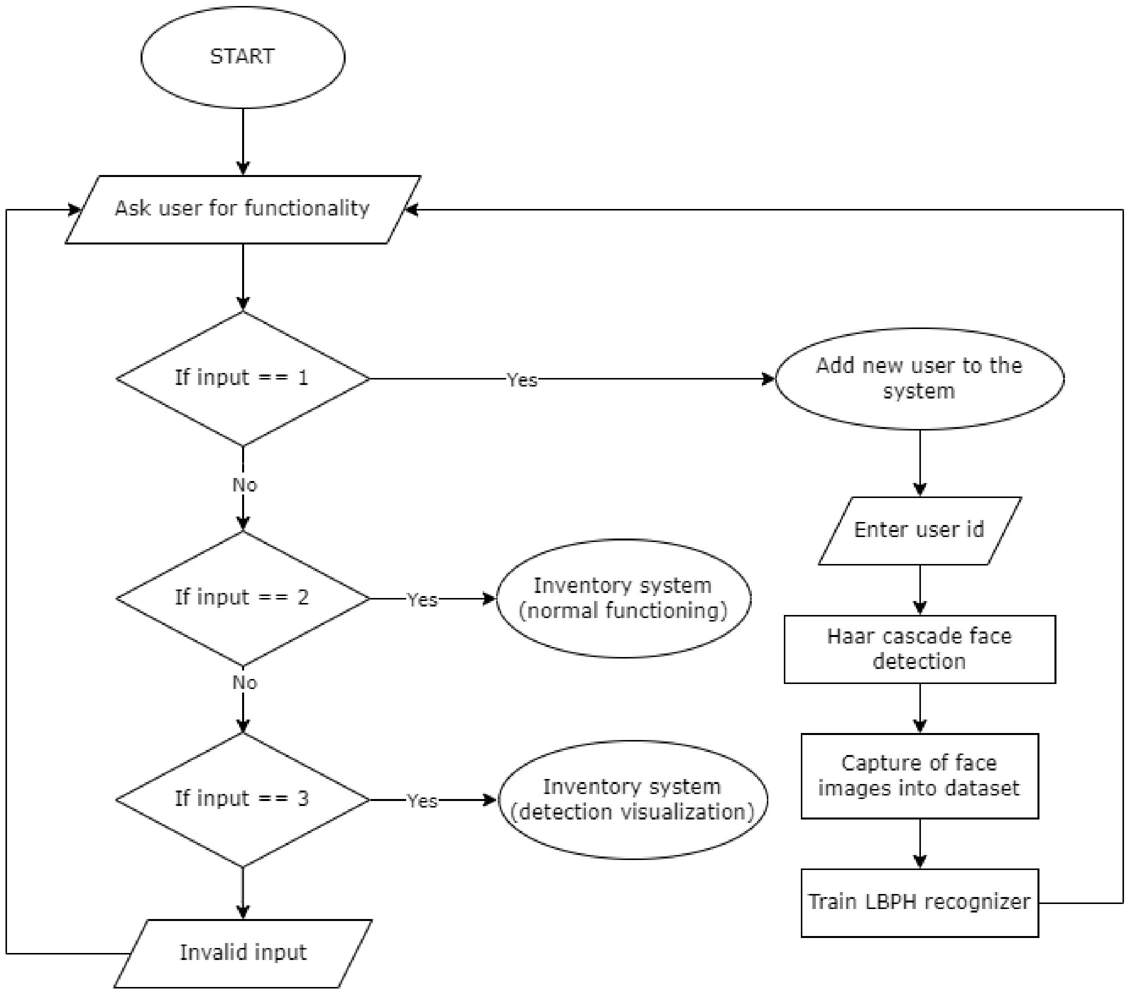

START
Ask user for functionality
If input == 1
Yes
Add new user to the system
No
Enter user id
If input == 2
Yes
Inventory system (normal functioning)
Haar cascade face detection
No
If input == 3
Yes
Inventory system (detection visualization)
Capture of face images into dataset
Train LBPH recognizer
Invalid input

Figure 3.18 Flowchart of functionality choice and addition of new user

On the other hand, if the user chooses option number two, the normal functioning of the inventory management system is set into motion. As briefly stated at the beginning of the present chapter, the logic implemented consists of updating the inventory registry only when the user is no longer in the camera frame, therefore giving the necessary time for the user to make any necessary changes or returns. As long as a face is detected, the system will only save the identification of the user, to later be set as the borrower of the tool or box that is no longer being detected. The updating of the inventory is done in a very similar way for both the tools and the boxes: by detecting these objects, two different lists are obtained, the objects detected in the image and those that are not detected and for each of the items in this list the correct values of the availability and the person borrowing the item are assigned. At the same time, the user’s key of the inventory dictionary is filled with the items that each user is borrowing at the time. In figure 3.19 it is shown the flowchart of the general logic behind the inventory management program:

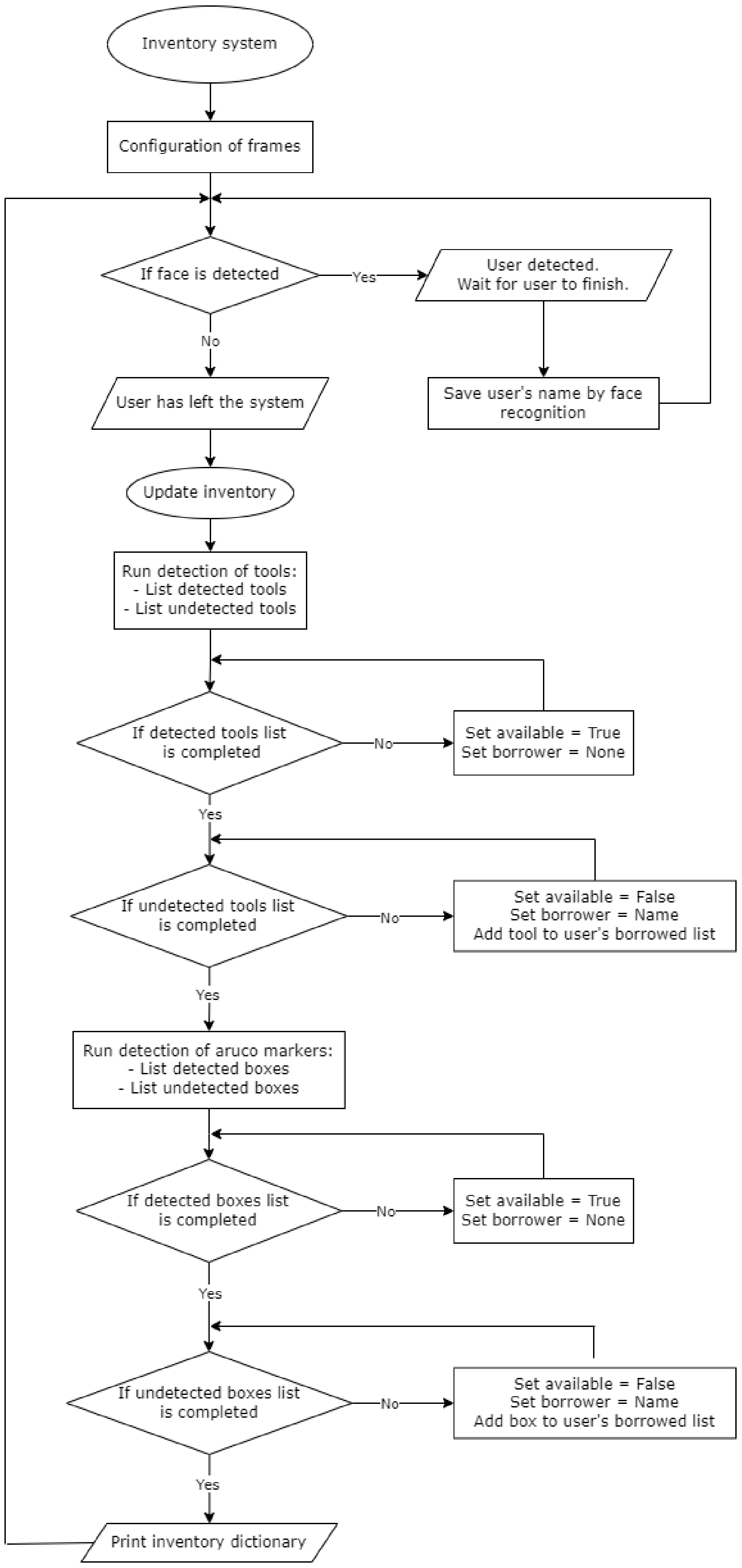
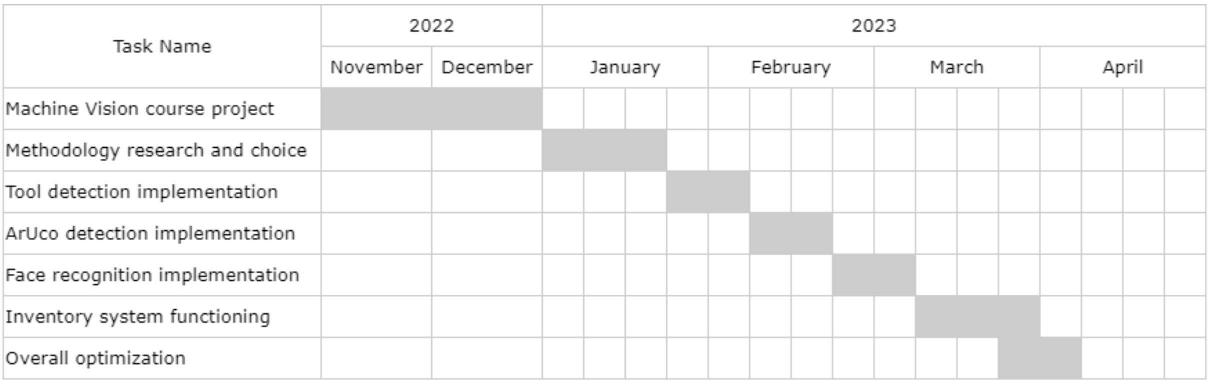


Figure 3.19 Flowchart of general logic behind inventory management

4 EXPERIMENTATION AND RESULTS

The development of this project was initially considered as a response to the demand from the department to have a system to control the available materials, and a first basic approach was implemented during the Machine Vision course as the final project of the subject. This project consisted in the detection of different tools using the YOLOv5 model in a different set up than the one implemented in the present system and allowed the familiarisation with the algorithms of the YOLO family.

The overall workflow and approximate time periods of each section's implementation of the system can be observed in the following figure 4.1:



Task Name	2022		2023																
	November	December	January				February				March				April			
Machine Vision course project	▨	▨																
Methodology research and choice			▨	▨	▨													
Tool detection implementation						▨	▨											
ArUco detection implementation								▨	▨									
Face recognition implementation										▨	▨							
Inventory system functioning												▨	▨	▨				
Overall optimization														▨	▨			

Figure 4.1 Gantt chart of the system's implementation

This chapter is divided into the results and analysis of each section of the system's implementation, finalising with an overall analysis of the complete system's functioning.

4.1 Results of object detection (tools)

This section of the project was started earlier, during the development of the Machine Vision course final project, and such implementation can be considered a first prototype. As previously stated, the object detection algorithm used was YOLOv5, another version of the YOLO family, also developed by Ultralytics as the finally used YOLOv8. Even though there are some significant differences between the two algorithms, the realization of this project served to become familiar with the object detection algorithms, being the first approach to this methodology, and facilitated the future implementation of the system proposed in this document. The Machine Vision project had the same functionality as the tool detection section of this system, although smaller in size, as only 3 tools were trained to be detected and with less system flexibility, as almost no variation in the environment was allowed for the system to function correctly.

The training results of the system reached a mAP for an IoU between 0,5 and 0,95 of 0,963 for the hacksaw, 0,908 for the pliers, 0,951 for the wrench and an overall value for all classes of 0,941. The training process observing the changes in the mAP 0,5:0,95 parameter over the training iterations can be observed in figure 4.2:

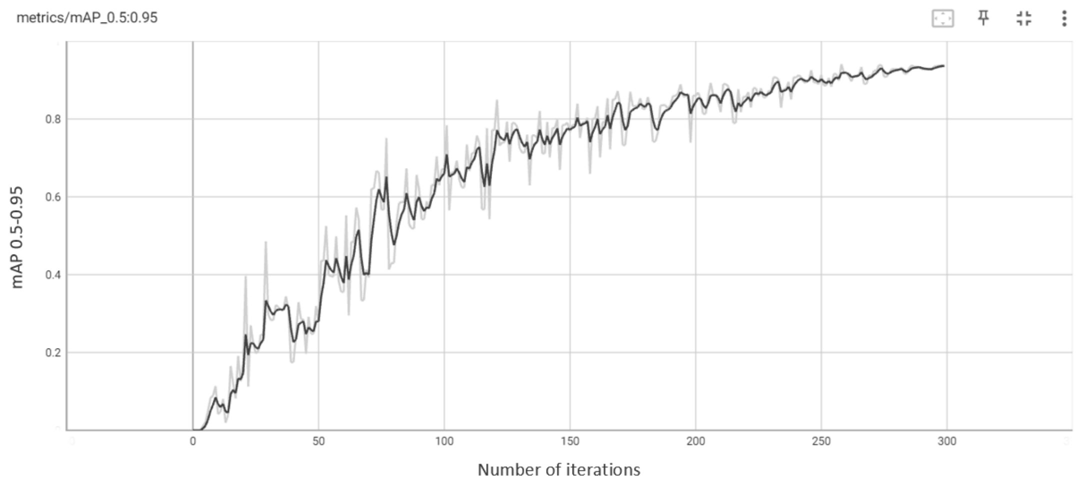


Figure 4.2 Machine vision's project mAP results

Although numerically the results are satisfactory, as the system was trained very strictly in terms of variations within the positions and orientations of the tools, the visual results when testing the system in real-time using the webcam, show the existing limitations. Figure 4.3 shows the result of running inference of the model when the tools are placed in the same positions as the training dataset (picture A) and the misdetections when the tools are placed in different positions or partially hidden (pictures B and C).

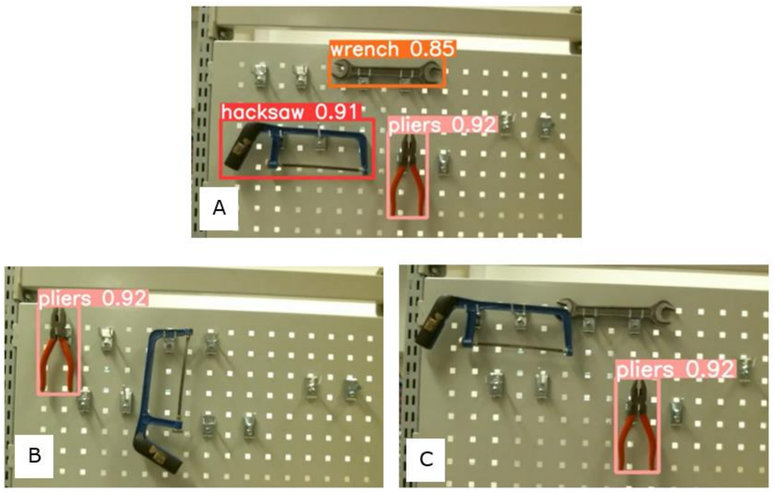


Figure 4.3 Inference runs of Machine Vision's project
A – correct detection; B, C – misdetections

The training and validation process for the new object detection algorithm using YOLOv8 was studied and analysed more extensively, and numerous tests were carried out until the optimal model was found. As previously stated, the implementation of the object detection used for tools using the YOLOv8 algorithm requires a series of steps. The preparation of the dataset was already explained and analysed in chapter three, however, although the final result of the balance between training data and validation data is 80% and 20% respectively, since the system is being implemented with a limited and relatively small dataset, as an initial approach, the division of 90% and 10% was attempted, in order to have a higher number of images used for training. After starting the training experiments with this balance of data, it became clear that the amount allocated for validation was insufficient as the results showed that the model did not converge at any time to a relatively stable value (even if it was low) and thus it is not possible to assess the performance of the model, in order to choose the optimal hyperparameters for the final training. Furthermore, initially, the amount of data augmentation techniques introduced was higher, which resulted in overfitting, as the model became too specialized to the augmented data and lost the ability to function correctly when new unseen data was assessed. Figure 4.4 shows the mAP results for an example of the previously explained case, using a batch size of 16:

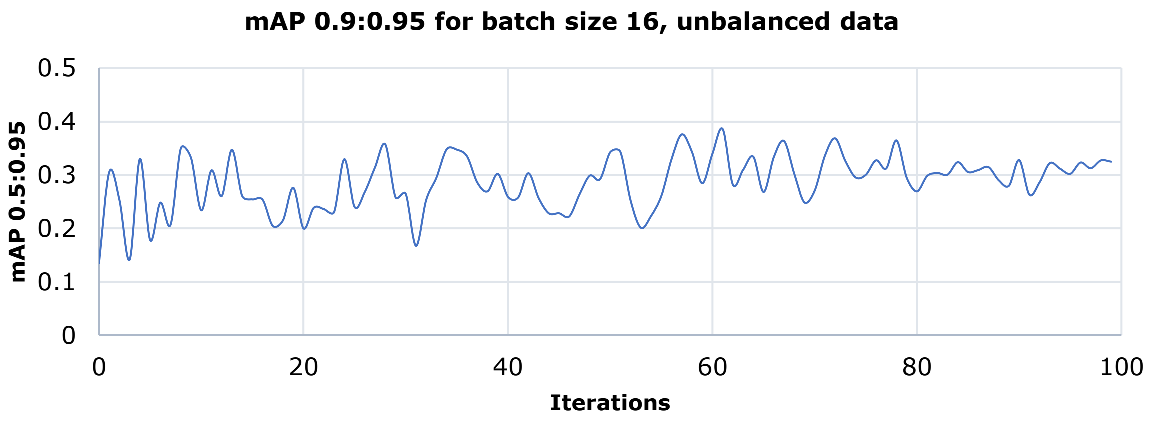


Figure 4.4 mAP results for an unbalanced model of training and validation data

Once the new and final version of the dataset was prepared, with fewer data augmentation techniques and a training-validation balance of 90% and 10% respectively, a series of training attempts were made to determine the optimal values for the hyperparameters of the batch size and epochs. First, to conclude on the batch size to be used, the model was trained with a low number of iterations (50) and the batch size was increased gradually, resulting in four different training with batch values of 8, 16, 32 and 64. After each training, the model’s performance was assessed to determine the optimal value, comparing the values of different parameters: training time, mAP 0,5, mAP 0,5:0,95 precision and recall. In table 4.1, a summary of the training results for batch selection is shown:

Table 4.1 Metrics results of batch selection trainings

Batch	**Epochs**	**Time (mins)**	**mAP 0,5**	**mAP 0,5:0,95**	**P**	**R**
8	50	14,04	0,995	0,921	0,973	0,996
16	50	14,04	0,995	0,931	0,973	0,999
32	50	14,7	0,995	0,938	0,985	1
64	50	16,2	0,995	0,932	0,98	1

The first value discarded for use corresponds to the highest one, with a batch size of 64. As already mentioned, a larger batch size requires more memory to be processed, and in this case, with a batch size of 64, the system, although capable of performing the set number of iterations and obtaining the highest mAP value, is at the limit of its memory capacity and will have trouble or even not be able to complete a higher number of iterations. A higher number of samples in each pass means that the model requires more memory to store the larger batch which slows down the training process, as it can also be seen in the time results, being the slowest training. The lowest value corresponding to a batch size of 8, although being one of the two fastest in training, obtains the worse results in mAP, precision, and recall. Furthermore, the graphs showing the training process show a slow and unsteady convergence into the final values of the parameters and is therefore also discarded for the final training. For the last two values of 16 and 32, the results in all parameters are slightly better in the second case but nevertheless, it was decided that both batch sizes would be further analysed in the process of choosing the optimal number of iterations, and asses which one results on the best performance model.

For the selection of the number of epochs, the model was attempted to be trained with 1000 iterations, and with the use of the patience parameter (number of epochs to monitor for lack of progress before stopping the training early) the optimal value could be determined. The initial value of 1000 iterations is chosen since when training with a limited dataset, it is suggested to train for a large number of epochs. If the training is stopped early because the patience value is reached, which is set to 50, the model can be retrained with a lower number, using as a reference the iteration at which the initial training was stopped. If, on the other hand, the training reaches its end, concluding the 1000 iterations, and the performance of the model is still showing gradual improvement, the model needs to be retrained with an even higher number of epochs. This process is to be repeated until the optimal performance of the model is found. In table 4.2, a summary of the training results for iteration selection is shown, using both of the batch sizes chosen:

Table 4.2 Metrics results for epochs selection

Batch	Epochs	Time (mins)	mAP 0,5	mAP 0,5:0,95	P	R
16	1000	45,36	0,995	0,932	0,964	1
16	150	41,94	0,995	0,945	0,989	1
32	1000	79,5	0,995	0,936	0,987	1
32	250	70,98	0,995	0,946	0,982	1

The first try was carried out with a batch size of 16, and after 169 iterations, with a training time of 45 mins and 21 seconds, it was stopped early since the patience value was reached. The best results were obtained at epoch 119 and therefore the model was retrained once again with an iteration value of 150 epochs. For the batch size of 32, the same situation occurred, and the training was stopped early at 283 epochs, obtaining the best results at iteration number 233. It was therefore retrained with 250 epochs. The final training for both batch sizes obtains the same results in recall and very similar results in mAP 0,5:0,95, with a very small advantage when using a batch size of 32. However, the precision value was notably better for the smaller bath size, which means the model is more likely to correctly identify true positives and less likely to generate false positives. Nevertheless, as the obtained metrics are very similar for both models, they were both further tested, checking their performance when running inference on the trained tools, using a real-time video feed from the set up’s camera. Although the camera was placed at the usual distance from the equipment, and the frame included the ArUco marker’s area to simulate the normal use of the system, the pictures have been cropped to avoid unnecessary content that does not contain any relevant information. Figure 4.5 contains four examples of detections using the model with batch size 16. As it can be observed in pictures A, B and C the model is able to detect the tools in different positions and placements within the rack, and picture D shows that the addition of untrained tools into the rack doesn’t disturb the results and no detection failures occur.

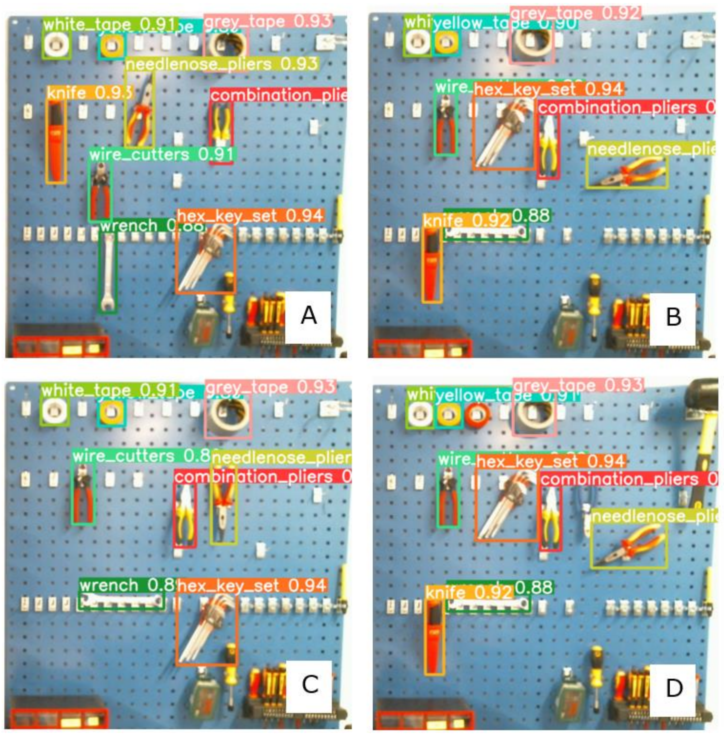


Figure 4.5 Inference detections for batch size 16
A – normal placement; B, C – different orientations and positions; D – additional tools placed

On the other hand, figure 4.6 shows the results for the detection results for the bigger batch of size 32, and in this case, a few misdetections can be observed. Pictures A and B show that the model is able to detect the tools in their usual orientations, even with a few minor changes, and doesn't perform any misdetections when untrained tools of very different structures are placed in the rack. However, picture C shows that when introducing a similar-looking object (red tape), a wrong detection is produced, labelling it as yellow tape as well. Furthermore, some misdetections are produced, not being able to detect the wrench when it is placed horizontally instead of its usual orientation.

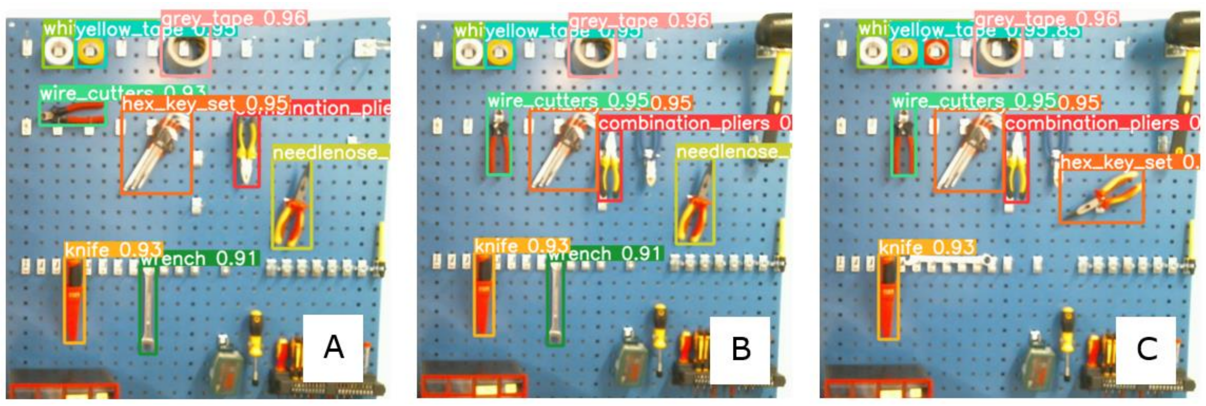


Figure 4.6 Inference detections for batch size 32
A – normal placement; B – additional different tools; C – additional similar tool

Even though the smaller batch size model already showed better performance results, a final test was carried out, calculating the average inference speed for 150 detections for both models. The model with batch 16 resulted in an average speed inference of 301,69 ms and the model with batch 32 had a slower average speed inference with a value of 304,93 ms. For this and all the previous reasons already explained, the final model chosen corresponds to a batch size of 16, trained for 150 epochs. In the following figure 4.7, the evolution of the different metrics during training for such model are shown:

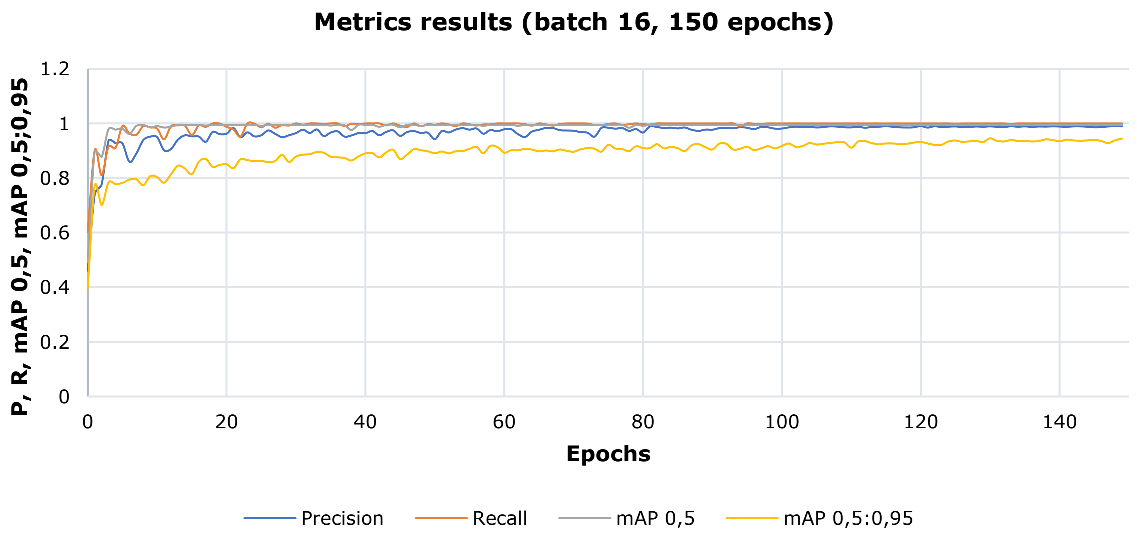


Figure 4.7 Metrics results for batch 16 and 150 epochs

Furthermore, when the training is completed, we can obtain other information about the training results, obtained using the validation dataset. Figure 4.8 shows the confusion matrix of the classification model, where the columns represent the actual classes and the columns the predicted classes of the objects. Each cell represents the amount of times an object of a particular class was correctly or incorrectly classified by the model. As it can be observed, the confusion matrix shows a perfect result for the trained model.

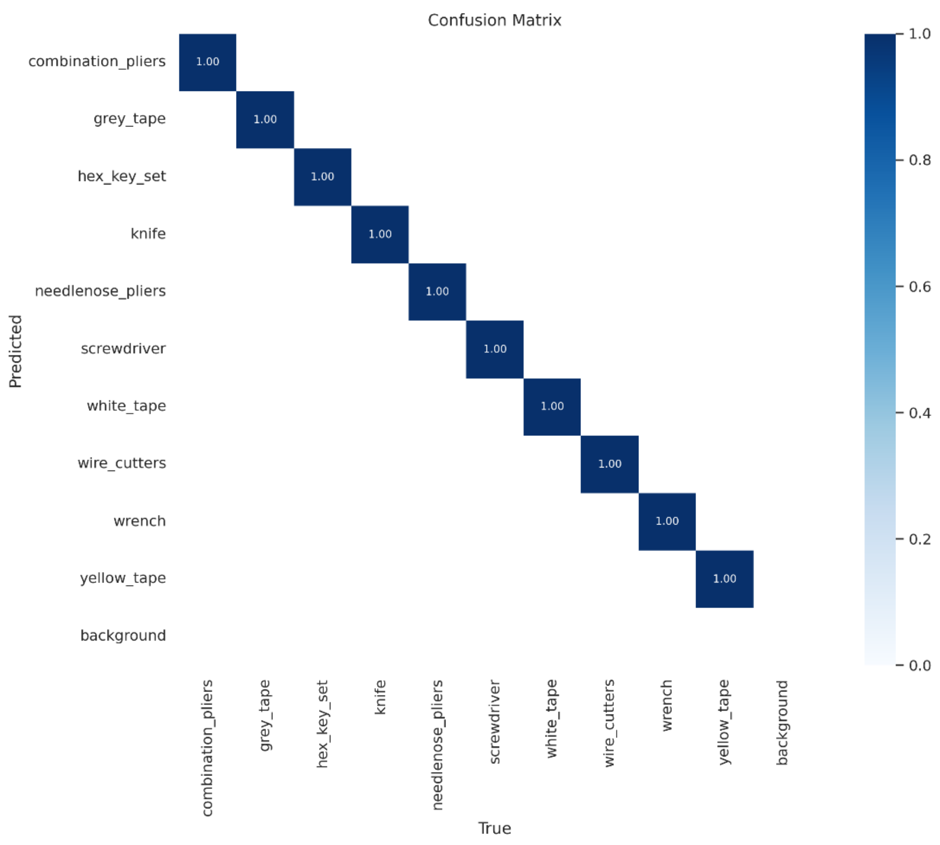


Figure 4.8 Confusion matrix for class classification (batch 16, 150 epochs)

Finally, the training and validation losses can be analysed. There are three types of losses calculated: box losses, class losses and Detection Loss Function (DFL). The box losses assess how well the model predicts the bounding box for the detected objects, the class losses measure how well the model predicts the class of the object within the bounding box and the detection loss function is the total loss of the model, calculated using the two previous losses. The model’s losses are calculated for both the training and validation dataset, and the optimal fit is that both losses decrease over time with a small or no difference between them, which means the model is achieving a good fit to the training data, and also working correctly when new unseen data is introduced. Figure 4.9 shows the comparison of the three different losses for training and validation:

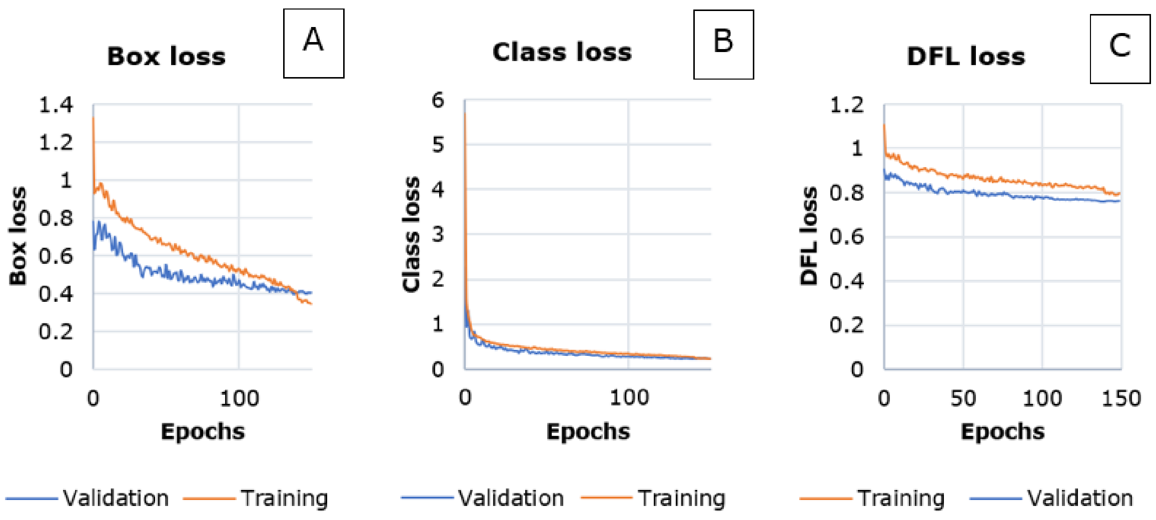


Figure 4.9 Training and validation losses (batch 16, 150 epochs)
A – box loss; B – class loss; C – DFL loss

The box loss shows a bigger difference between the validation and training (picture A); however, the class losses are very similar to each other (picture B) and therefore, the total losses or DFL have quite similar values, with a small difference between training and validation, and the model's losses analysis concludes with a good result.

It should be noted that although the results are generally satisfactory, and the model works correctly when real-time video inference is performed, it is trained and, above all, validated with a very limited dataset of only 46 images. Since one of the main objectives of this project is the use of a limited dataset, and that the tool detection system works correctly, it is a very satisfactory result. However, it is important to consider that the performance results of the trained YOLOv8 model may not generalize well to larger datasets or different use cases.

In appendix 1, the metrics and losses result graphs for all the different trainings are shown, along with the confusion matrices.

4.2 Results of ArUco detection (boxes)

The experimentation performed using ArUco markers is divided into two different steps: first, the choice of a marker dictionary, suitable for the needed application at the given conditions; and secondly, the correct detection of such markers when they are placed within the system's set up.

For the first step, a series of marker's dictionary examples were printed, of different marker size and image size, measured in pixels, to determine which one is the most suitable with the camera's resolution, lighting conditions, and geometrical characteristics present in the real system set up. Equal to the number of markers that can be used in the system, sets of 10 markers were printed. From the beginning, considering the distance from the camera to the placement of the boxes and the camera's resolution, it was determined that the image size of the markers had to be equal to 150 pixels, as any smaller size would not be recognisable for the system at such given conditions, and as any larger size image could not be placed correctly in the boxes. Therefore, the testing was performed with images of 150 pixels, and the marker size (number of bits in each marker) varies from 4x4 to 7x7 bits. Figures 4.10, 4.11, 4.12 and 4.13 show the results for the marker dictionaries for the marker size of 4x4, 5x5, 6x6 and 7x7 respectively, placed at different locations within the system's working frame, and where the misdetections have been marked in a red bounding box.

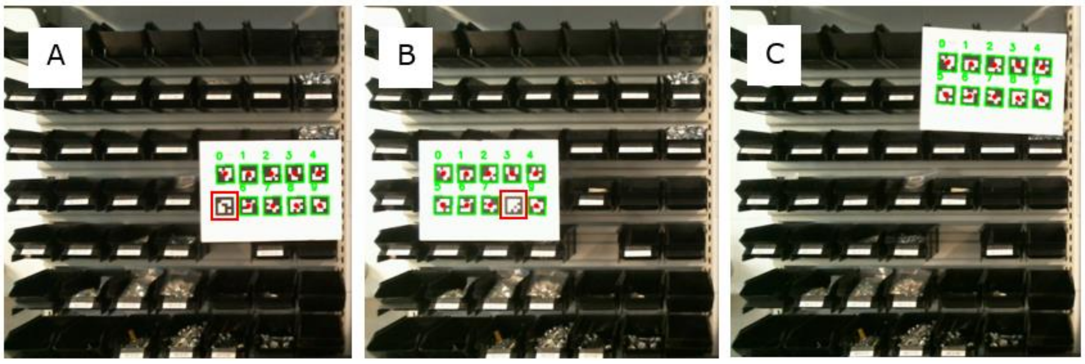


Figure 4.10 Testing of marker dictionary size 4x4

The 4x4 bits markers were detected in most of the occasions, however, a there were a few misdetections as it can be seen in pictures A and B. Having only 16 bits of information, the 4x4 markers have less redundancy. If a part of the markers is undetected, the system doesn't have enough information to conclude the ID of the marker using only the remaining bits, therefore, it's not robust enough to correctly detect all of the markers.

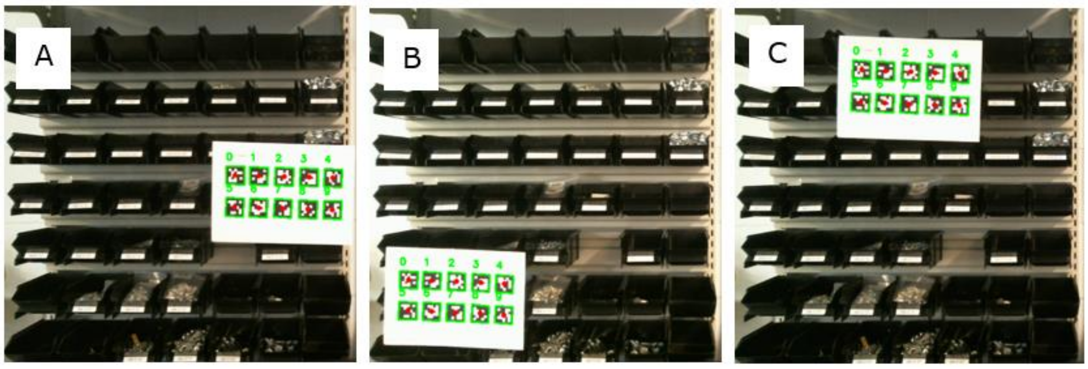


Figure 4.11 Testing of marker dictionary size 5x5

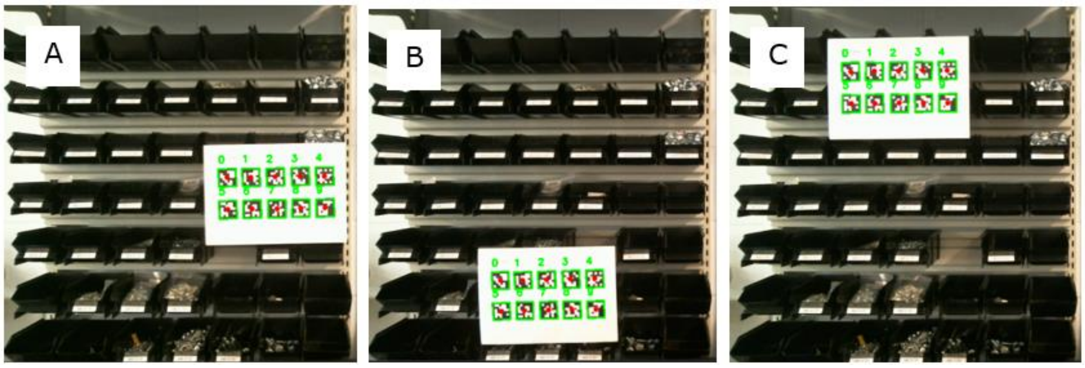


Figure 4.12 Testing of marker dictionary 6x6

The marker dictionaries with 5x5 and 6x6 bits resulted in perfect detections, with all of the markers being detected in both cases and placed in different positions within the frame. Having 25 and 36 bits of information respectively, they have higher redundancy and error correction capabilities in case some parts of the markers aren't correctly seen. Both of them are further tested below to select the final marker dictionary used.

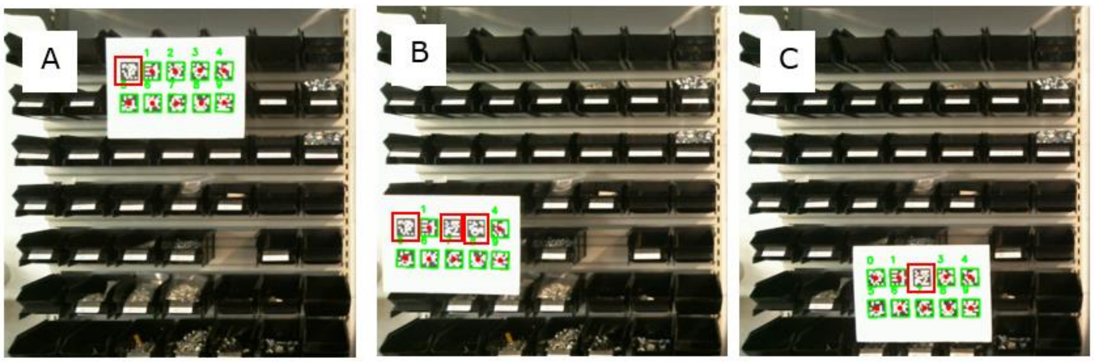


Figure 4.13 Testing of marker dictionary 7x7

Finally, the 7x7 markers, although they should result in higher redundancy and error correction capabilities, showed the most misdetections out of the tested markers. This is due to the resolution of the camera that is not able to distinguish the different patterns, being the size of the bits smaller and therefore resulting in a blurrier image.

The markers dictionaries of 5x5 and 6x6 bits are then tested using separated markers placed in different locations in the frame, and checking if once a marker is removed, it is no longer detected, and the rest of the markers still are. Figure 4.14 shows the detection for 5x5 markers and figure 4.15 shows the detection for 6x6 markers:

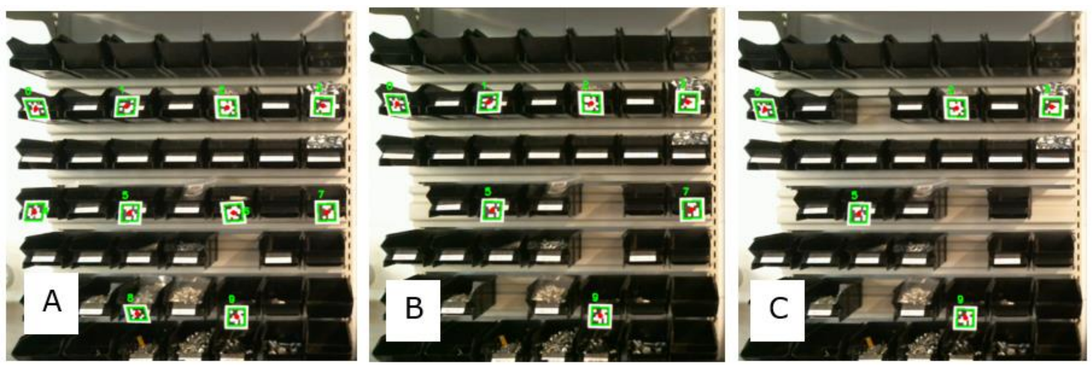


Figure 4.14 Detection of 5x5 bits markers

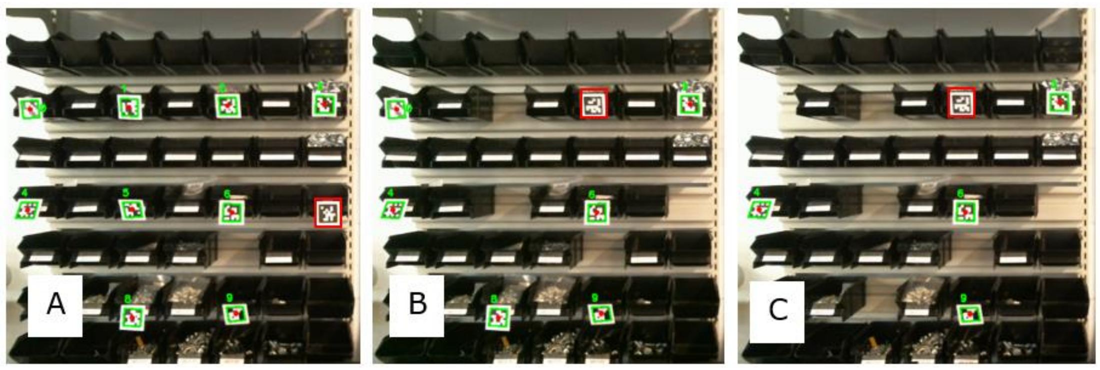


Figure 4.15 Detection of 6x6 bits markers

As it can be seen in the figures above, the 5x5 markers result in a perfect performance of the system with no misdetections and the 6x6 markers, although detecting correctly in most of the cases, still have some misdetections. Therefore, the markers of size 5x5 bits are the final used for the inventory management system, having a good combination between the number of bits of information (that provides more redundancy and error detection capabilities) and having the right bit size for the resolution of the camera used.

4.3 Results of face recognition (users)

As stated in the previous chapter, the face recognition task of the system is performed in three different steps: recollection of an image dataset of the user, training of the recognizer and lastly, detection of faces within the frame and its recognition to perform the identification of the user.

The recollection of the image dataset consists of first, transforming the image into grayscale and using the face detector, only the portion of the image that contains the face of the user is saved. Once the user has entered the identifier number through the terminal, the system captures a certain number of images. This number can be configured, and in the system is set to 50, which although it may seem a high number, it only takes the system 12 seconds to capture that number of images. The user is asked to slowly rotate the head to capture different angles of the face, to add more flexibility to the recognition system. Figure 4.16 shows the head rotation performed by the user and a few examples of images from a user's dataset, in different positions.

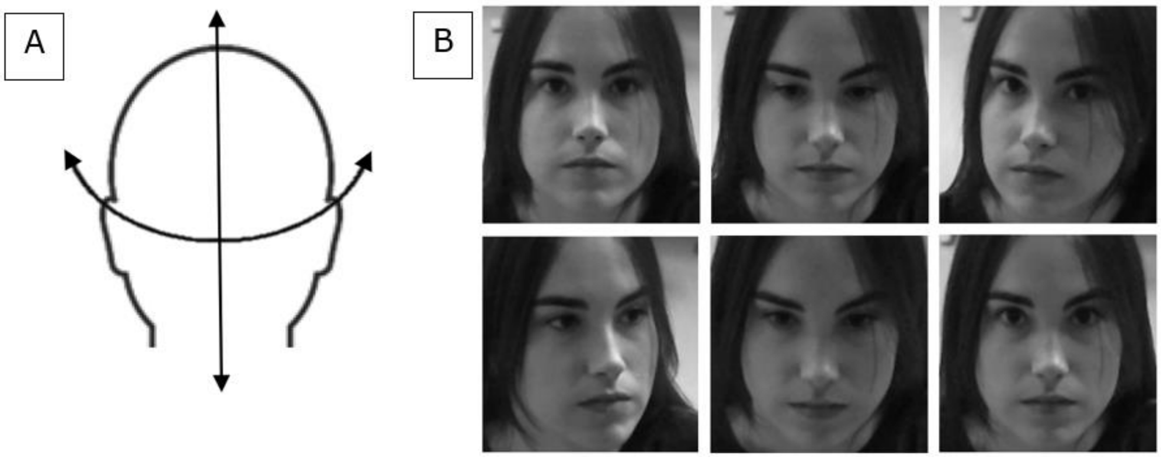


Figure 4.16 A - Rotation of the user's head, B - Example images from dataset

Once the dataset is completed, the recognizer is trained with such images, and the user's identification can be performed through face recognition. In figure 4.17 the recognition of the same previous user can be observed, in a normal position as the user approaches to borrow an item:

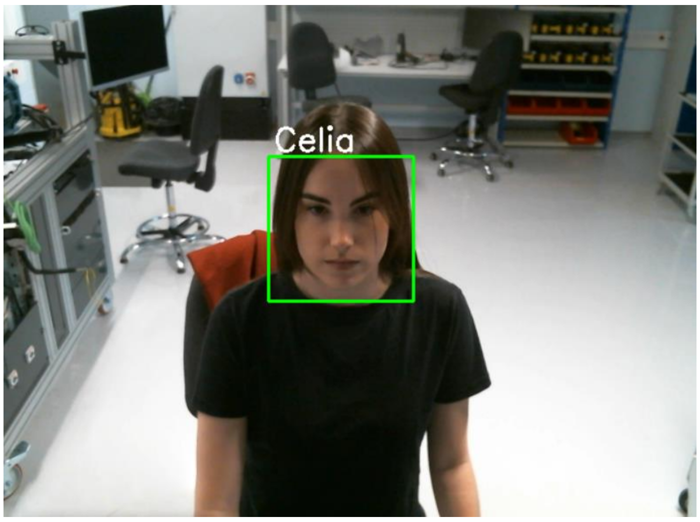


Figure 4.17 User's recognition in a normal position

Different scenarios were tried out, in order to test the flexibility and performance of the system, and to be able to identify its limitations, when slight changes are introduced like light variations, different appearances of the user or the orientation of the user’s head and the position within the frame. In all cases, the user is looking at the objects that can be borrowed, instead of looking at the camera, to simulate the situation that would occur in the real system.

In figure 4.18, the results with the same light adjustment but different head orientations and positions within the frame are shown. Pictures A through D show the user in the center of the frame looking at different sides. As it can be observed, in both cases looking left and right, if the head is rotated at a slight angle, the recognition is still successful (pictures A and C). However, if the angle is more severe, both the recognition and detection of the face fail, with the system no longer being able to identify the user (pictures B and D). On the other hand, when the user is located at the edges of the frame, shown in pictures E through H, the results are successful, with the system being able to correctly identify the user, only if the user's head is slightly oriented towards the centre, closer to the origin of the image (pictures E and G). If the user is positioned with the head facing toward the edges of the frame, the system isn’t able to perform the user’s identification (pictures F and H).

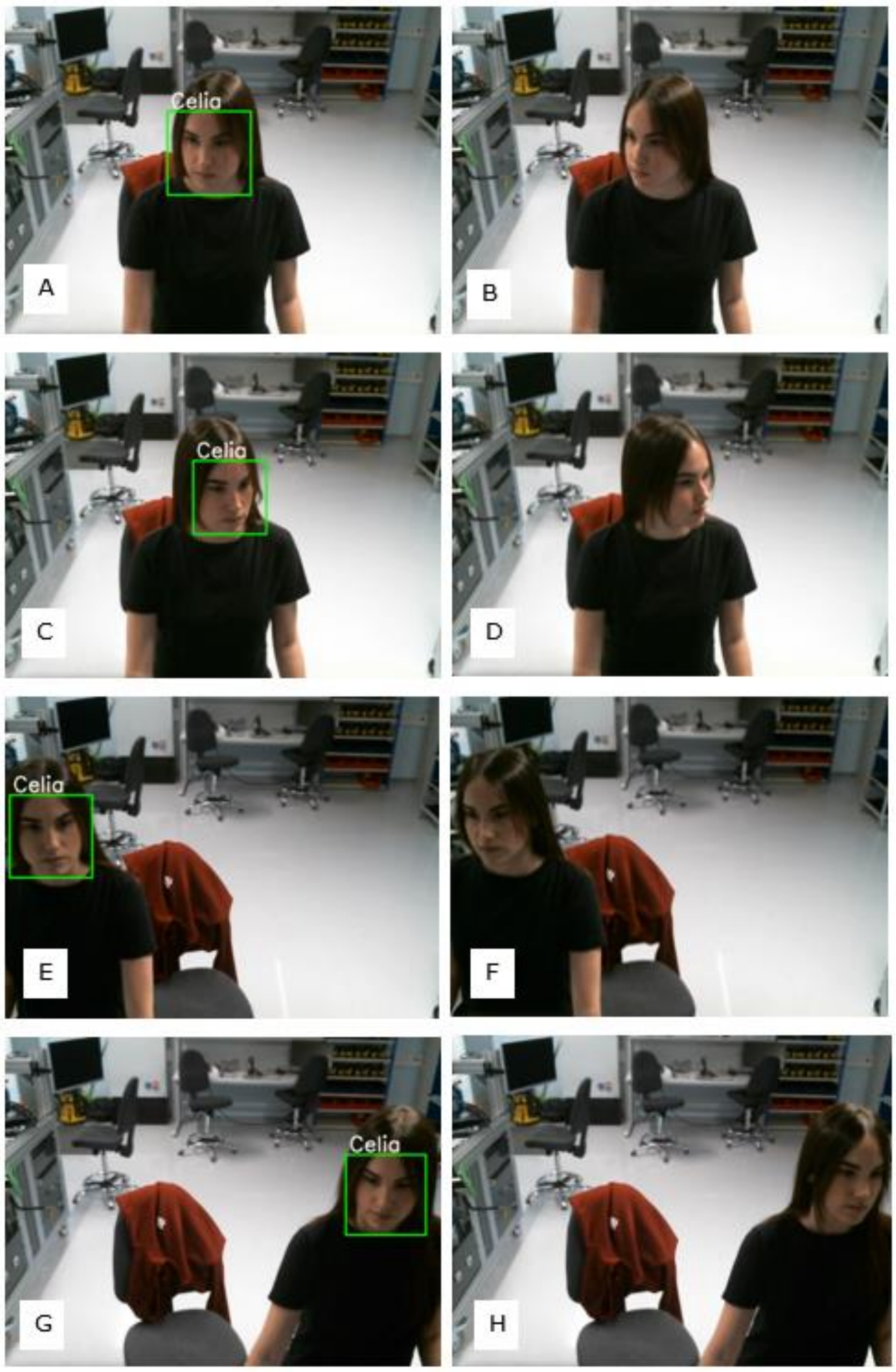


Figure 4.18 Face recognition in different conditions
A, B, C, D – user's head rotations; E, F, G, H – user's position in the edges of the frame

In figure 4.19, the same light conditions are applied, but the user's appearance is changed. The detections are in most cases satisfactory but as it can be observed in picture B, in some cases with even a small rotation of the user's head, the recognition fails. On the edges of the frame (pictures E and F) the system works correctly, as before if the user is facing slightly toward the camera.

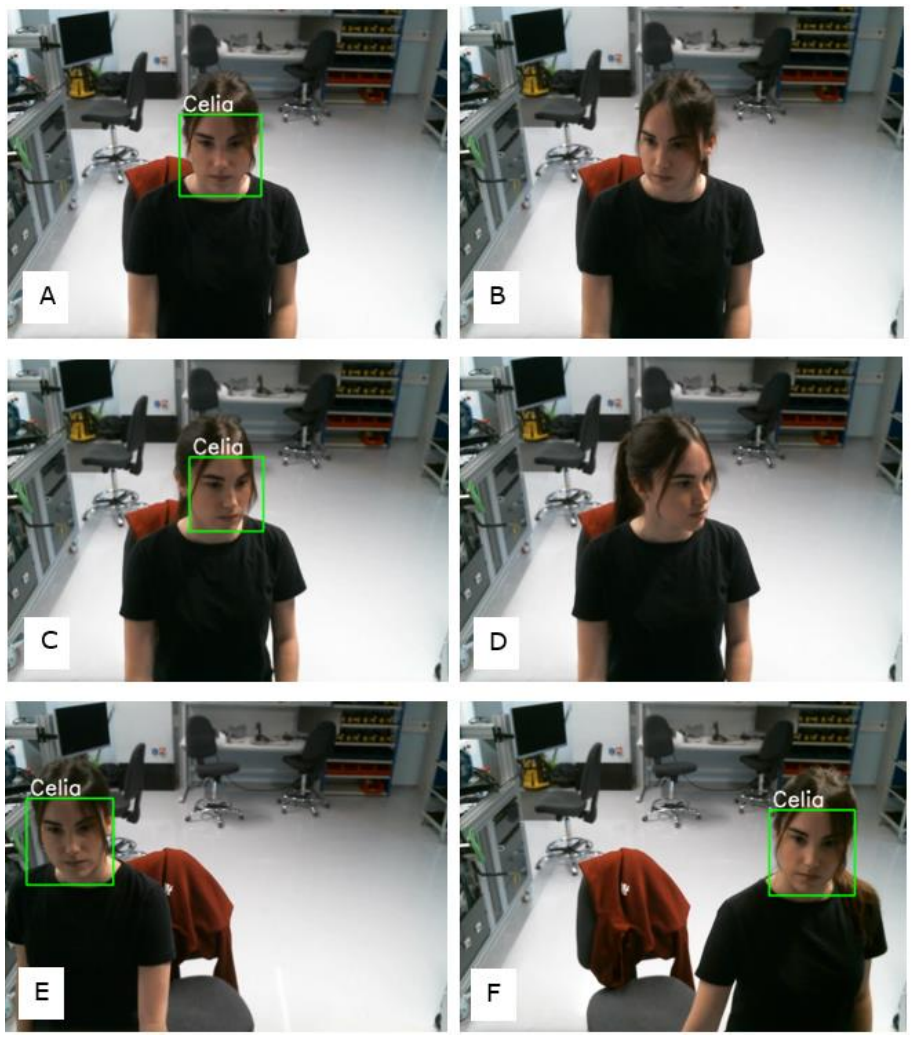


Figure 4.19 Face recognition in user's different appearance
A, B, C, D - user's head rotations; E, D - user's position in the edges of the frame

Lastly, the system's performance was tried with the lowest light condition available at the place of the system's set up. As it can be observed in figure 4.20, the results are mostly unsatisfactory, with the system having major problems in recognising facial features (pictures B and C) unless the user is facing toward the middle of the frame (picture A) in which case, the system identifies the user correctly. Furthermore, with these conditions, the system in certain cases is able to detect that a face is found in the frame, but it is classified as unknown, as it cannot relate the features to those based on the dataset with which the recogniser is trained (picture D).

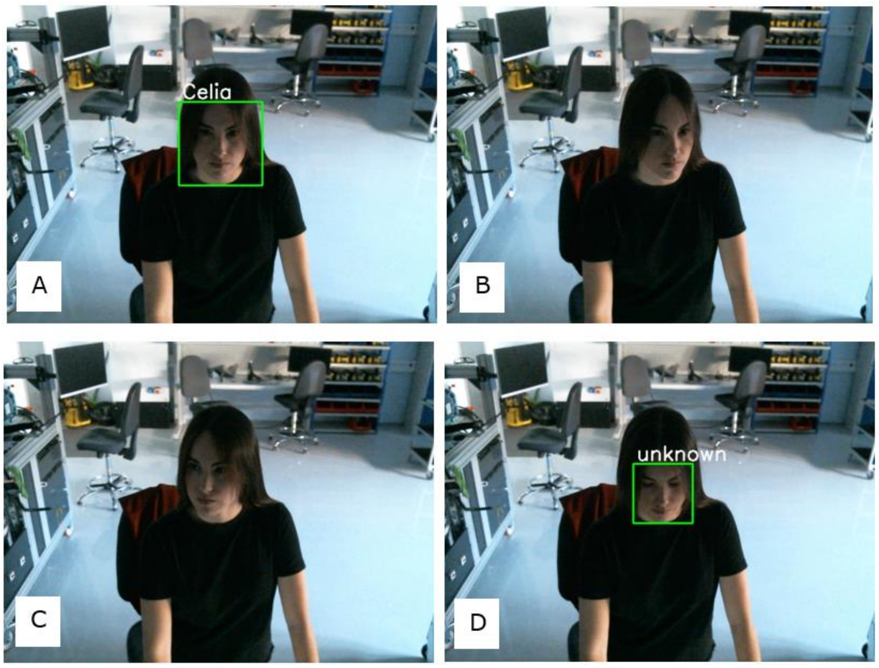


Figure 4.20 Face recognition with a different light condition

Overall, the results of the face recognition implementation are considered satisfactory when the lights conditions are appropriate and as long as the user is in a minimum part facing towards the centre of the frame, which is the usual situation that occurs when the user is picking up items from the available area. Although it may not recognise the user continuously at all times, due to the structure of the designed code, the system will store the name of the user once it is recognised for the first time and will be assigned as the borrower of the items that are no longer within the frame after the user's action is done.

4.4 Results of inventory management system

The final implementation of the inventory management system consists of: tool detection by YOLOv8 small algorithm trained with a batch size of 16 and for 150 epochs; boxes detection using ArUco markers of 5x5 bits and image size of 150 pixels; and face detection and recognition using a Haar feature-based cascade classifier and LPBH respectively. Once each section was correctly implemented and tested and using the program logic previously explained in chapter 3, the functioning of the whole system was tested. To that end, the set up shown in figure 4.21 was implemented and several scenarios were simulated consisting of the following:

1. The whole inventory is available.
2. A user borrows two tools (screwdriver and wrench) and a box of camera equipment (box 0).
3. The user returns one of the tools (wrench).
4. The user returns the remaining items in her possession.

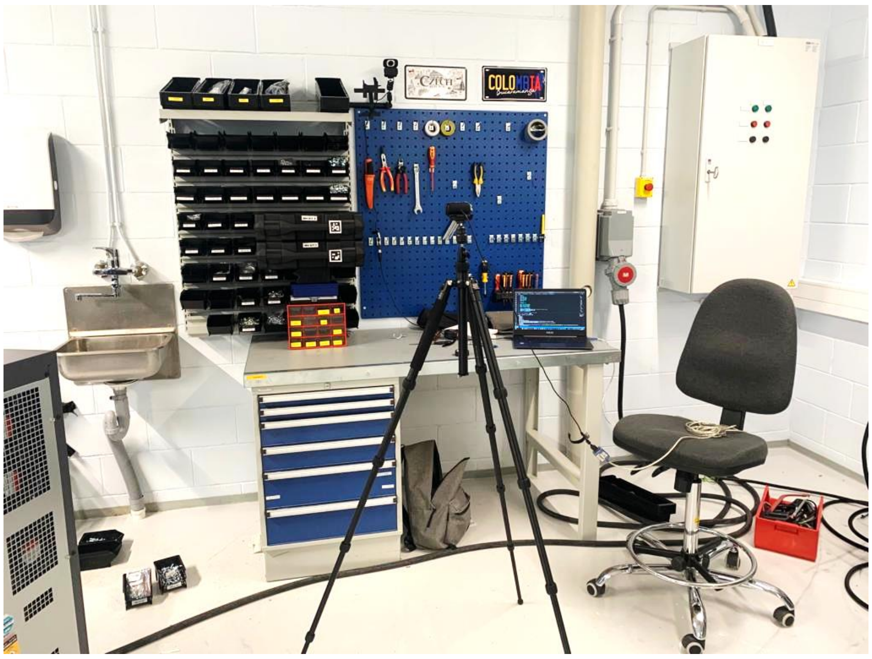


Figure 4.21 Final testing set up

At the beginning, the inventory equipment is all accounted for, and both all of the tools and boxes are within the working frame and detected. Figure 4.22 shows the real-life detection of the inventory equipment, both tools and boxes in pictures A and B respectively:

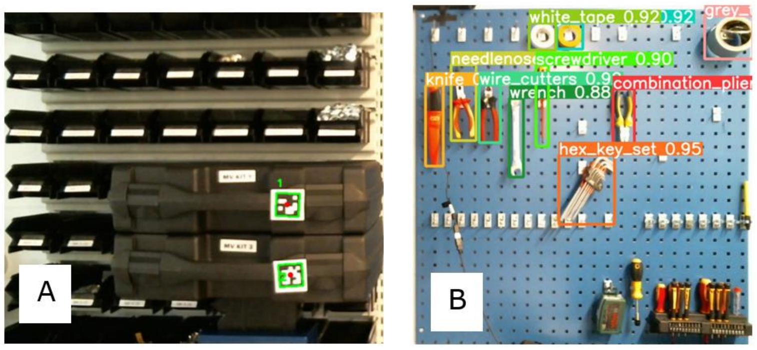
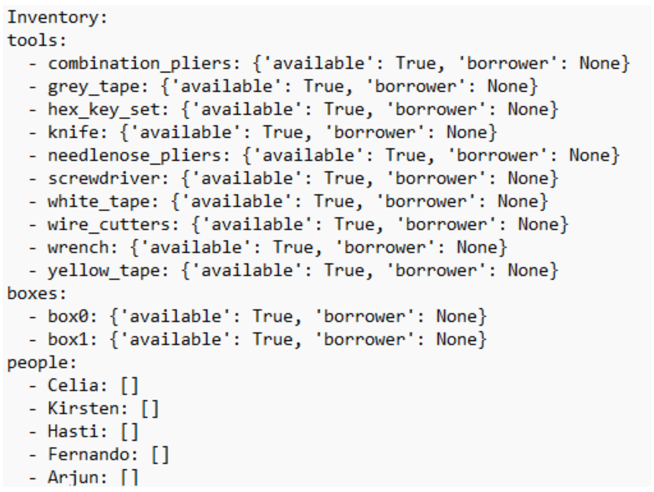


Figure 4.22 Full inventory detection

For the given scenario, and as no face is detected with the second camera, the inventory system is updating and shows the following information:



```
Inventory:
tools:
  - combination_pliers: {'available': True, 'borrower': None}
  - grey_tape: {'available': True, 'borrower': None}
  - hex_key_set: {'available': True, 'borrower': None}
  - knife: {'available': True, 'borrower': None}
  - needlenose_pliers: {'available': True, 'borrower': None}
  - screwdriver: {'available': True, 'borrower': None}
  - white_tape: {'available': True, 'borrower': None}
  - wire_cutters: {'available': True, 'borrower': None}
  - wrench: {'available': True, 'borrower': None}
  - yellow_tape: {'available': True, 'borrower': None}
boxes:
  - box0: {'available': True, 'borrower': None}
  - box1: {'available': True, 'borrower': None}
people:
  - Celia: []
  - Kirsten: []
  - Hasti: []
  - Fernando: []
  - Arjun: []
```

Figure 4.23 Inventory information displayed when all equipment is available

Once a user enters the frame of the second camera, the updating of the inventory is stopped, and the message "User detected. Waiting for user to finish" is displayed. At the same time, the system performs face recognition on the face detected and saves the name to be set as the borrower for any tools or boxes that might be missing after the user leaves the frame. Figure 4.24 shows the user's recognition when borrowing the tools:

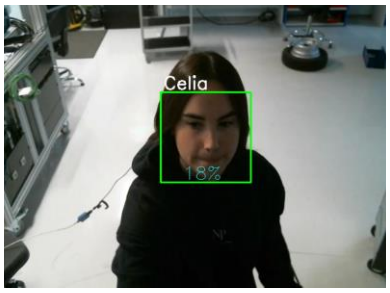


Figure 4.24 User's face recognition when borrowing equipment

Once the user is no longer within the frame, the message "User has left the system." is displayed, and the updating of the inventory list is resumed. In this scenario, the user takes a tool (screwdriver) and a box (number 0) as it can be seen in the corresponding detections in the following figure 4.25:

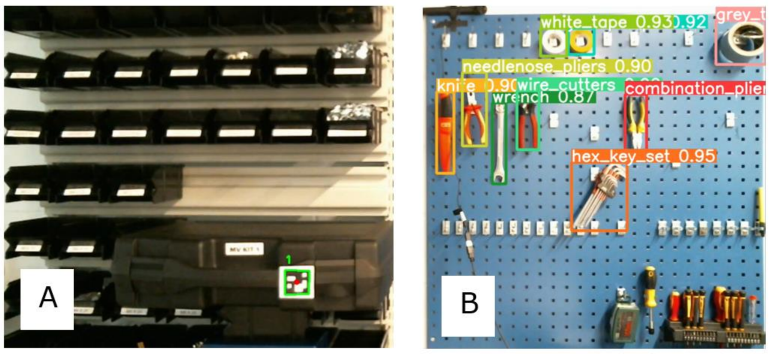
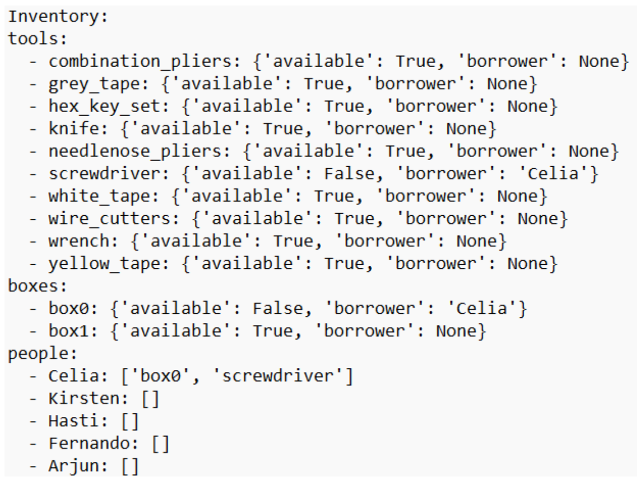


Figure 4.25 Detection of available equipment

And once the update is restored, the information shown in the system is the following:



```
Inventory:
tools:
  - combination_pliers: {'available': True, 'borrower': None}
  - grey_tape: {'available': True, 'borrower': None}
  - hex_key_set: {'available': True, 'borrower': None}
  - knife: {'available': True, 'borrower': None}
  - needlenose_pliers: {'available': True, 'borrower': None}
  - screwdriver: {'available': False, 'borrower': 'Celia'}
  - white_tape: {'available': True, 'borrower': None}
  - wire_cutters: {'available': True, 'borrower': None}
  - wrench: {'available': True, 'borrower': None}
  - yellow_tape: {'available': True, 'borrower': None}
boxes:
  - box0: {'available': False, 'borrower': 'Celia'}
  - box1: {'available': True, 'borrower': None}
people:
  - Celia: ['box0', 'screwdriver']
  - Kirsten: []
  - Hasti: []
  - Fernando: []
  - Arjun: []
```

Figure 4.26 Inventory information displayed when equipment is borrowed

After that, the user is placed once more within the frame, the updating is stopped while the tools and box are returned, and the system starts updating and shows a full inventory available as in figure 4.23 when the user leaves the system's frame.

5 SUMMARY

5.1 Conclusions

Inventory management systems are of great importance for industrial manufacturing facilities, warehouses and companies of any size. Inventory management plays a critical role in adding reliability, security and efficiency to the system, controlling all elements present, from raw elements to finished products or assets used and produced by the company. Current solutions use combinations of different technologies such as artificial intelligence, sensory systems, computer vision, deep learning or RFID, among others. These solutions result in complex and expensive systems, which are only suitable for companies with extensive resources, and are unaffordable for smaller companies.

In this thesis, a multi-camera vision-based inventory management system of greater simplicity and therefore, suitable for smaller companies with fewer resources has been implemented. The approach to the development of this project appeared to meet the department's need for a system to control and monitor the available materials and the resulting system creates a registry keeping a record of the users' use of such equipment. For that purpose, four goals were initially set, corresponding to the main sections implemented in the project: development of a face recognition application to detect the user, development of two object recognition systems for the lab equipment and the camera kits, and finally, implementation of a program that combines all the previous solutions and provides the information collected by the inventory system.

Each section was researched, analysed, and tested in order to find the most suitable methodology, considering the particular conditions of the system's environment as well as the material available for its implementation. The final implementation of the inventory management system consists of: tool detection by YOLOv8 small algorithm trained with a batch size of 16 and for 150 epochs; boxes (camera equipment) detection using ArUco markers of 5x5 bits and image size of 150 pixels; and face detection and recognition using a Haar feature-based cascade classifier and LPBH respectively. Finally, the functioning of the whole system was tested. Several scenarios were simulated in which, starting with a full inventory, a user borrows certain equipment, and it is later returned.

After the different tests carried out both in each section separately and in the system as a whole, and as it is shown in this work, the different objectives have been met and the results are satisfactory, with the system being able to keep the inventory managed and updated. Logically, the scope of the project is limited and moreover, this project has been developed as a concept idea. Therefore, further implementations would need to be made in order to be used in real life on a regular basis. Furthermore, since one of the main objectives was to keep the developed system as simple as possible, the resources are limited and the results, although satisfactory, are bounded by the conditions of the environment. In this sense, in the following section, some future works are proposed to enhance the capabilities and give more flexibility to the system that has been developed.

5.2 Future works

As previously stated, the current system is bounded by its limited flexibility, and therefore a series of future implementations are proposed for its further development. Firstly, the current set up is not the most convenient for the department and the working area where the equipment is available, as it is in the middle of user movement zones and secondly, the use of a tripod means that it can be accidentally moved at any time, which would affect the functioning of the system. Hence, a rough idea for a structure to hold the camera monitoring the equipment is proposed, attached to the roof of the laboratory as it counts with metal bars already installed, where it could be easily placed. In this way, the material could be monitored continuously without the risk of the camera disturbing users or being displaced and unable to perform its function. Figure 5.1 shows an approximate outline of the proposed structure:

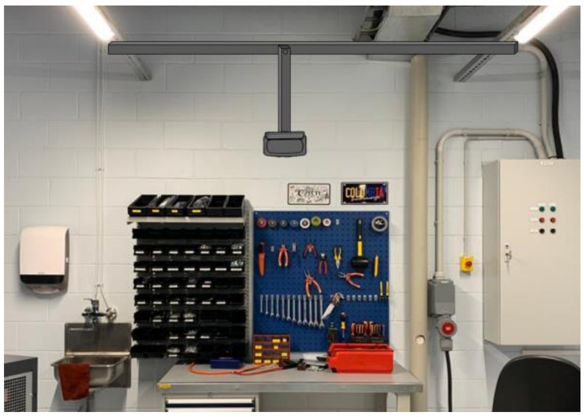


Figure 5.1 Outline of the proposed future structure

Also related to hardware material, the use of a camera with higher resolution would greatly facilitate the object's detection and user's recognition. Furthermore, if the camera can be connected to the system without the need for any physical connections to the computer, an independent system could be obtained, and the inventory could be managed remotely from different locations.

On the other hand, as future implementations related to the software, different ideas are proposed for each developed section. The implementations for tool detection and user recognition are both constrained by the goal of using limited resources, which means only a certain number of images are used for each section. The accuracy and performance of the system would notably improve only with an increase in the number of images used, training each corresponding algorithm in a more flexible manner. However, if this objective is to continue to be met, different detection or recognition methods could be combined to determine with greater certainty the classification of an object or the identification of a user. The detection of ArUco markers used in the boxes of camera equipment is the most accurate implementation, whose greatest weakness is the reflections caused by both artificial and natural lighting. As a future implementation to improve this situation, two methods are proposed: the use of a coverage of the markers with an anti-reflective material or image processing to reduce said reflections to a minimum.

Finally, the addition of a user-friendly interface as well as the implementation of the program with a reliable memory to store the information gathered by the inventory management system, would make it more interactive and appropriate for use in real life and on a regular basis.

KOKKUVÕTE

Varude haldamise süsteemid on väga olulised tööstuslikes tootmisrajatistes, ladudes ja igas suuruses ettevõtetes. Varude haldamine mängib kriitilist rolli süsteemi töökindluse, turvalisuse ja tõhususe lisamisel, kontrollides kõiki olemasolevaid elemente alates toorelementidest kuni valmistoodete või ettevõtte poolt kasutatavate ja toodetud varadeni. Praegused lahendused kasutavad erinevate tehnoloogiate kombinatsioone, nagu tehisintellekt, sensoorsed süsteemid, arvutinägemine, süvaõpe või RFID. Nende lahenduste tulemusel valmivad keerulised ja kallid süsteemid, mis sobivad vaid suurte ressurssidega ettevõtetele, väiksematele ettevõtetele pole need aga jõukohased.

Käesolevas lõputöös on kasutatud mitme kaameraga visioonipõhist laohaldussüsteemi, mis on lihtsam ja seetõttu sobilik väiksematele ja väiksemate ressurssidega ettevõtetele. Selle projekti arendamise lähenemisviis vastas osakonna vajadusele olemasolevate materjalide kontrollimise ja jälgimise süsteemi järele. Sellest tulenev süsteem loob registri, mis hoiab arvestust selliste seadmete kasutajate kasutustegevuste kohta. Selleks seati algselt neli eesmärki, mis vastavad projekti põhiosadele: näotuvastusrakenduse väljatöötamine kasutaja tuvastamiseks, kahe objektituvastussüsteemi väljatöötamine laboriseadmete ja kaamerakomplektide jaoks ning lõpuks programmi kasutusele võtmine, mis ühendab kõik varasemad lahendused ja annab inventuurisüsteemi kogutud teabe.

Iga sektsiooni uuriti, analüüsiti ja testiti, et leida kõige sobivam metoodika, arvestades nii süsteemi keskkonna eritingimusi kui ka selle rakendamiseks saadaolevat materjali. Varude haldamise süsteemi lõplik juurutamine koosneb tööriista tuvastamisest YOLOv8 väikese algoritmi abil, mis on koolitatud partii suurusega 16 ja 150 epohhi jaoks; kastide (kameraseadmete) tuvastamisest, kasutades ArUco markereid suurusega 5x5 bitti ja pildi suurust 150 pikslit; ning näotuvastusest ja tuvastamisest, kasutades vastavalt Haari funktsioonipõhist kaskaadiklassifikaatorit ja LPBH-d. Lõpuks testiti kogu süsteemi toimimist. Simuleeriti mitmeid stsenaariume, mille puhul kasutaja laenab alates täielikust laoseisust teatud seadmed ja tagastab need hiljem.

Pärast erinevaid teste, mis on viidi läbi nii igas jaotises eraldi kui ka süsteemis tervikuna ja nagu käesolevas töös on näidatud, on erinevad eesmärgid täidetud ja tulemused rahuldavad, kuna süsteem suudab hoida laoseisu hallatuna ja ajakohasena. Loogiliselt võttes on projekti maht piiratud ja pealegi on see projekt välja töötatud ideeideena. Seetõttu tuleks reaalses elus regulaarseks kasutamiseks teha täiendavaid rakendusi. Lisaks, kuna üks peamisi eesmärke oli hoida väljatöötatav süsteem võimalikult lihtsana, on ressursid piiratud ja tulemused, kuigi rahuldavad, on piiratud keskkonnatingimustega. Selles mõttes pakutakse järgmises jaotises välja mõned tulevased tööd, et täiustada võimalusi ja anda väljatöötatud süsteemile rohkem paindlikkust.

LIST OF REFERENCES

[1] "Inventory Management Defined Plus Methods and Techniques." https://www.investopedia.com/terms/i/inventory-management.asp (accessed Nov. 27, 2022).

[2] "Inventory Tracking Simplified: Steps, Methods and Efficiency Tips | NetSuite." https://www.netsuite.com/portal/resource/articles/inventory-management/inventory-tracking.shtml (accessed Nov. 27, 2022).

[3] "Amazon.com: Amazon Go." https://www.amazon.com/b?ie=UTF8&node=16008589011 (accessed Nov. 27, 2022).

[4] "Amazon One." https://one.amazon.com/ (accessed Feb. 21, 2023).

[5] R. Lin, "The Importance of Successful Inventory Management to Enterprises-A Case Study of Wal-Mart," 2019, doi: 10.25236/mfssr.2019.154.

[6] "How Does Walmart Inventory Management System Works?" https://www.urtasker.com/walmart-inventory-management-system/ (accessed Nov. 27, 2022).

[7] "The first unmanned store in Estonia was opened." https://taltech.ee/en/news/first-unmanned-store-estonia-was-opened (accessed Nov. 27, 2022).

[8] "What Is Object Detection? - MATLAB & Simulink." https://se.mathworks.com/discovery/object-detection.html (accessed Nov. 27, 2022).

[9] "Detection, Recognition, Identification Ranges FAQ - HGH Infrared." https://hgh-infrared.com/detection-recognition-identification-ranges-faq/ (accessed Nov. 27, 2022).

[10] "A Gentle Introduction to Object Recognition With Deep Learning - MachineLearningMastery.com." https://machinelearningmastery.com/object-recognition-with-deep-learning/ (accessed Nov. 27, 2022).

[11] "Deep Learning vs. Machine Learning: Beginner's Guide | Coursera." https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide (accessed Nov. 27, 2022).

[12] "What Is Deep Learning? | How It Works, Techniques & Applications - MATLAB & Simulink." https://se.mathworks.com/discovery/deep-learning.html (accessed Nov. 27, 2022).

[13] A. S. Jubair, A. J. Mahna, and H. I. Wahhab, "Scale Invariant Feature Transform Based Method for Objects Matching," *Proceedings - 2019 International Russian Automation Conference, RusAutoCon 2019*, Sep. 2019, doi: 10.1109/RUSAUTOCON.2019.8867657.

[14] "HOG (Histogram of Oriented Gradients): An Overview | by Mrinal Tyagi | Towards Data Science." https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f (accessed Nov. 27, 2022).

[15] "SIFT | How To Use SIFT For Image Matching In Python." https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/ (accessed Nov. 27, 2022).

[16] P. D. Wardaya, "Support vector machine as a binary classifier for automated object detection in remotely sensed data," *IOP Conf Ser Earth Environ Sci*, vol. 18, no. 1, 2014, doi: 10.1088/1755-1315/18/1/012014.

[17] "Object Detection in 2022: The Definitive Guide - viso.ai." https://viso.ai/deep-learning/object-detection/ (accessed Nov. 27, 2022).

[18] "Object Detection Explained: R-CNN | by Ching (Chingis) | Towards Data Science." https://towardsdatascience.com/object-detection-explained-r-cnn-a6c813937a76 (accessed Nov. 27, 2022).

[19] "Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN - MATLAB & Simulink - MathWorks Nordic." https://se.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html (accessed Nov. 27, 2022).

[20] "R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms | by Rohith Gandhi | Towards Data Science." https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e (accessed Nov. 27, 2022).

[21] T. Diwan, G. Anirudh, and J. v. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimed Tools Appl*, 2022, doi: 10.1007/S11042-022-13644-Y.

[22] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 6517–6525, Nov. 2017, doi: 10.1109/CVPR.2017.690.

[23] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018, doi: 10.48550/arxiv.1804.02767.

[24] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, doi: 10.48550/arxiv.2004.10934.

[25] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You Only Learn One Representation: Unified Network for Multiple Tasks," May 2021, doi: 10.48550/arxiv.2105.04206.

[26] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul. 2022, doi: 10.48550/arxiv.2207.02696.

[27] "Ultralytics YOLOv8 Docs," 2023. https://docs.ultralytics.com/ (accessed Feb. 09, 2023).

[28] "What is YOLOv8? The Ultimate Guide." https://blog.roboflow.com/whats-new-in-yolov8/ (accessed Feb. 09, 2023).

[29] "YOLOv8 - Ultralytics | Revolutionizing the World of Vision AI." https://ultralytics.com/yolov8 (accessed Feb. 09, 2023).

[30] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, Dec. 2015, doi: 10.1007/978-3-319-46448-0_2.

[31] "Getting Started with SSD Multibox Detection - MATLAB & Simulink - MathWorks Nordic." https://se.mathworks.com/help/vision/ug/getting-started-with-ssd.html (accessed Nov. 27, 2022).

[32] "Object Detection Metrics With Worked Example | by Kiprono Elijah Koech | Towards Data Science." https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e (accessed Feb. 13, 2023).

[33] J. Kaur and W. Singh, "Tools, techniques, datasets and application areas for object detection in an image: a review," *Multimedia Tools and Applications 2022 81:27*, vol. 81, no. 27, pp. 38297–38351, Apr. 2022, doi: 10.1007/S11042-022-13153-Y.

[34] "COCO - Common Objects in Context." https://cocodataset.org/#home (accessed Feb. 14, 2023).

[35] T.-Y. Lin *et al.*, "LNCS 8693 - Microsoft COCO: Common Objects in Context," 2014.

[36] "PASCAL VOC Dataset | Papers With Code." https://paperswithcode.com/dataset/pascal-voc (accessed Feb. 14, 2023).

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," pp. 248–255, Mar. 2010, doi: 10.1109/CVPR.2009.5206848.

[38] "ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite." https://github.com/ultralytics/yolov5 (accessed Feb. 17, 2023).

[39] "ultralytics/ultralytics: YOLOv8 in PyTorch > ONNX > CoreML > TFLite." https://github.com/ultralytics/ultralytics (accessed Feb. 17, 2023).

[40] Z. Zhang, Y. Hu, G. Yu, and J. Dai, "DeepTag: A General Framework for Fiducial Marker Design and Detection," *IEEE Trans Pattern Anal Mach Intell*, no. 01, pp. 1–1, May 2022, doi: 10.1109/TPAMI.2022.3174603.

[41] H. Uchiyama and E. Marchand, "Object Detection and Pose Tracking for Augmented Reality: Recent Approaches", Accessed: Nov. 27, 2022. [Online]. Available: https://hal.inria.fr/hal-00751704

[42] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014, doi: 10.1016/J.PATCOG.2014.01.005.

[43] "OpenCV: Detection of ArUco Markers." https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html (accessed Nov. 27, 2022).

[44] B. Li, J. Wu, X. Tan, and B. Wang, "ArUco Marker Detection under Occlusion Using Convolutional Neural Network," *Proceedings - 5th International Conference on Automation, Control and Robotics Engineering, CACRE 2020*, pp. 706–711, Sep. 2020, doi: 10.1109/CACRE50138.2020.9230250.

[45] M. J. Wu, Y. C. Chen, Y. S. Liao, J. A. Chen, and H. H. Lin, "Face-recognition System Design and Manufacture," *Proceedings - 22nd IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2021-Fall*, pp. 158–161, 2021, doi: 10.1109/SNPD51163.2021.9705014.

[46] E. Jiang, "A review of the comparative studies on traditional and intelligent face recognition methods," *Proceedings - 2020 International Conference on Computer*

Vision, Image and Deep Learning, CVIDL 2020, pp. 11–15, Jul. 2020, doi: 10.1109/CVIDL51233.2020.00010.

[47] ZhaoW., ChellappaR., PhillipsP. J., and RosenfeldA., "Face recognition," *ACM Computing Surveys (CSUR)*, vol. 5, pp. V-305-V–308, Dec. 2003, doi: 10.1145/954339.954342.

[48] "General Data Protection Regulation (GDPR) – Official Legal Text." https://gdpr-info.eu/ (accessed Nov. 27, 2022).

[49] "The Facts on Facial Recognition with Artificial Intelligence." https://aws.amazon.com/rekognition/the-facts-on-facial-recognition-with-artificial-intelligence/ (accessed Nov. 27, 2022).

[50] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, Mar. 2021, doi: 10.1016/J.NEUCOM.2020.10.081.

[51] Y. Taigman, M. Y. Marc', A. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification".

[52] P. Chandrakala MTech, Bs. Assistant Professor, and Ma. Kumar Professor, "Real Time Face Detection and Face Recognition using OpenCV and Python," vol. 13, 2022, Accessed: Dec. 02, 2022. [Online]. Available: www.jespublication.com

[53] M. G. Galety, F. H. al Mukthar, R. J. Maaroof, F. Rofoo, and S. Arun, "Marking Attendance using Modern Face Recognition (FR): Deep Learning using the OpenCV Method," *8th International Conference on Smart Structures and Systems, ICSSS 2022*, 2022, doi: 10.1109/ICSSS54381.2022.9782265.

[54] P. A. Harsha Vardhini, S. P. R. D. Reddy, and V. P. Parapatla, "Facial Recognition using OpenCV and Python on Raspberry Pi," *2022 International Mobile and Embedded Technology Conference, MECON 2022*, pp. 480–485, 2022, doi: 10.1109/MECON53876.2022.9751867.

[55] "LFW Dataset | Papers With Code." https://paperswithcode.com/dataset/lfw (accessed Dec. 02, 2022).

[56] "VGGFace2 Dataset | Papers With Code." https://paperswithcode.com/dataset/vggface2-1 (accessed Dec. 02, 2022).

[57] B. G. Batchelor, "Machine vision handbook," *Machine Vision Handbook*, pp. 1–2272, Jan. 2012, doi: 10.1007/978-1-84996-169-1/COVER.

[58] "Environmental factors for object detection - IBM Documentation." https://www.ibm.com/docs/en/video-analytics/1.0.6?topic=analytics-environmental-factors-object-detection (accessed Feb. 22, 2023).

[59] "Logitech C615 Full HD Webcam." https://www.logitech.com/en-us/products/webcams/c615-webcam.960-000733.html (accessed Apr. 14, 2023).

[60] "Cascade Classifier." https://apmonitor.com/pds/index.php/Main/CascadeClassifier (accessed Apr. 14, 2023).

[61] A. Ahmed, J. Guo, F. Ali, F. Deeba, and A. Ahmed, "LBPH based improved face recognition at low resolution," *2018 International Conference on Artificial Intelligence and Big Data, ICAIBD 2018*, pp. 144–147, Jun. 2018, doi: 10.1109/ICAIBD.2018.8396183.

APPENDICES

Appendix 1 Metric's results for YOLOv8 trainings

A1.1 Results for batch selection trainings

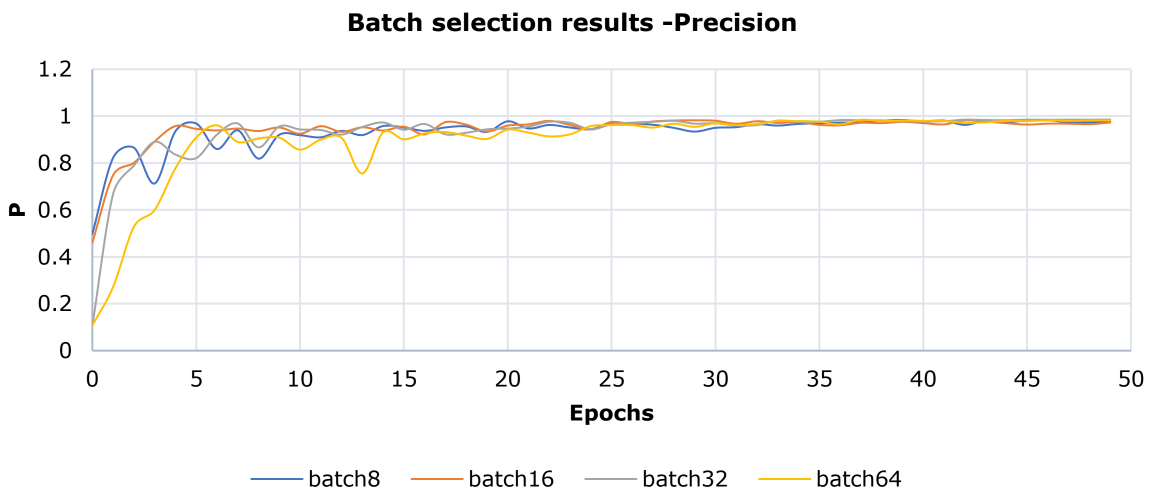


Figure A1.1 Precision results for batch selection trainings

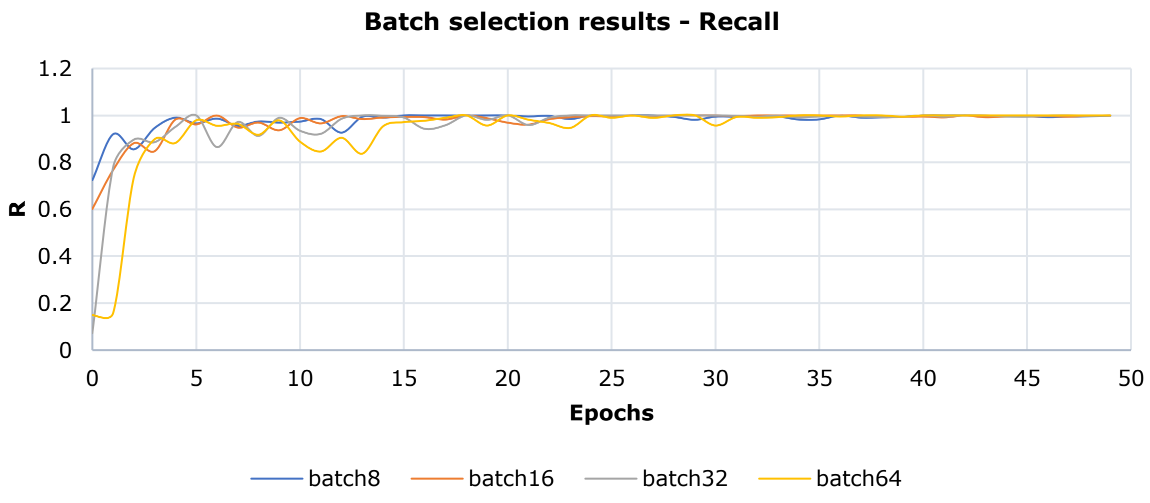


Figure A1.2 Recall results for batch selection trainings

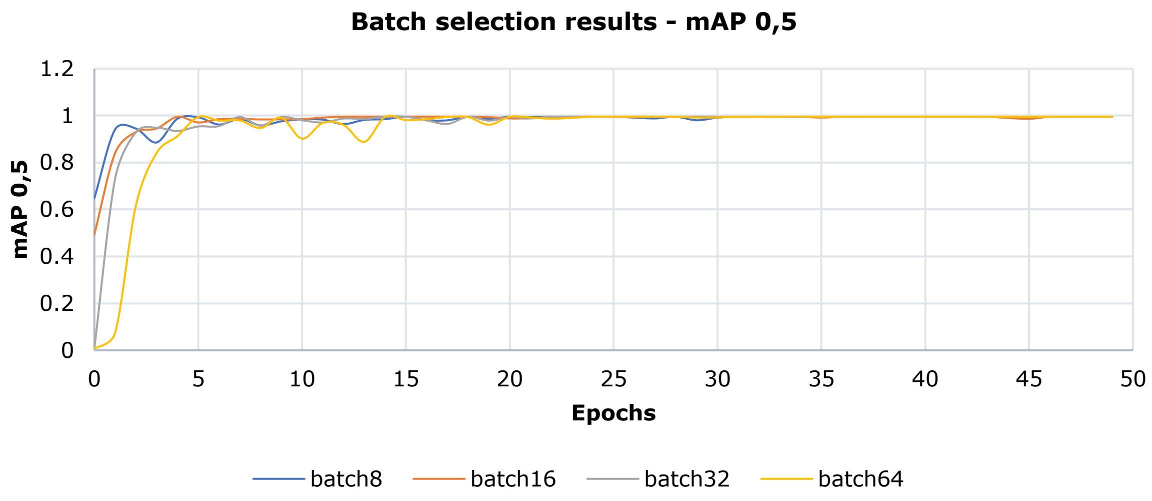


Figure A1.3 mAP 0,5 results for batch selection trainings

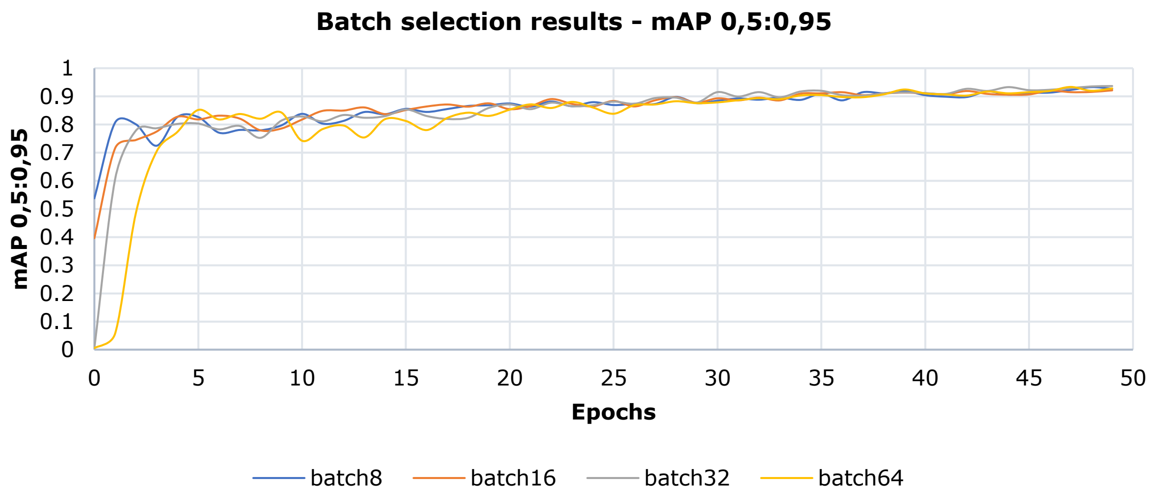
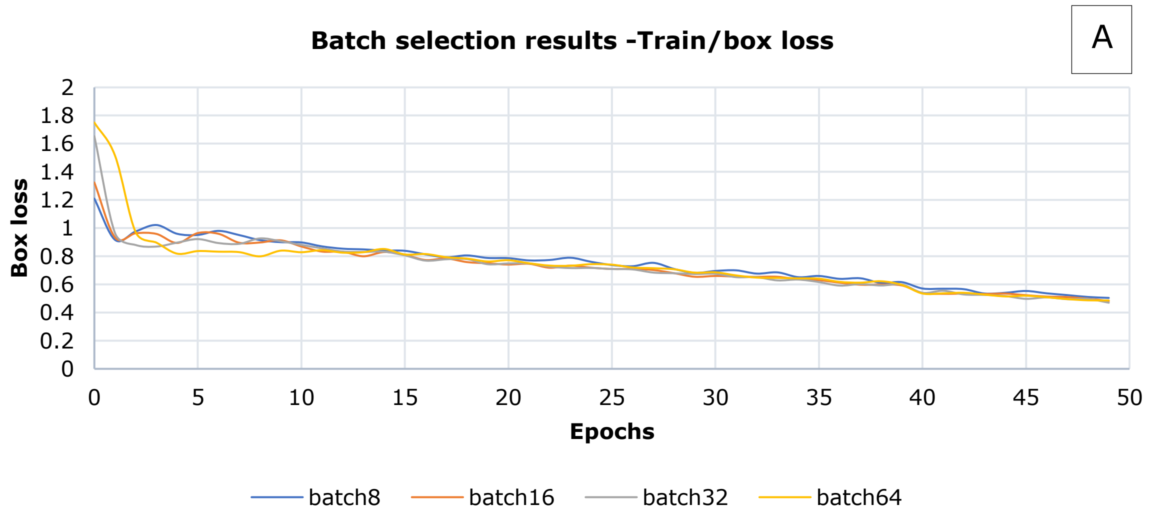
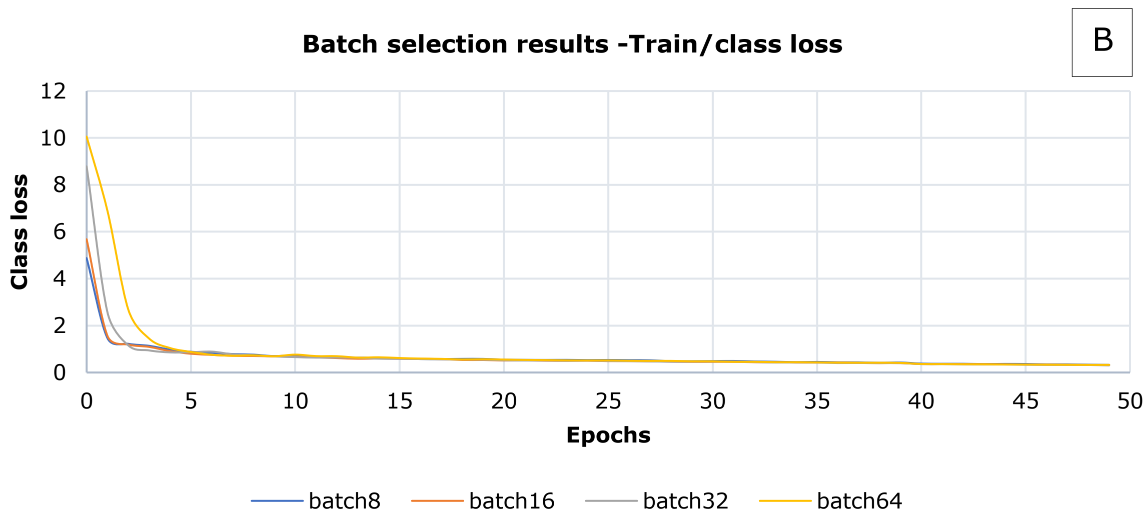


Figure A1.4 mAP 0,5:0,95 for batch selection trainings

A





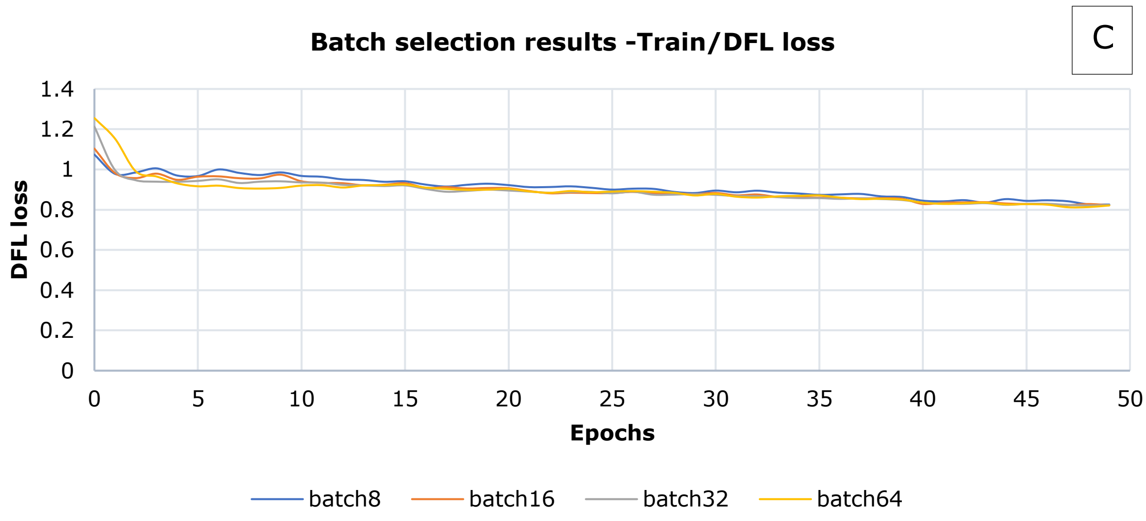
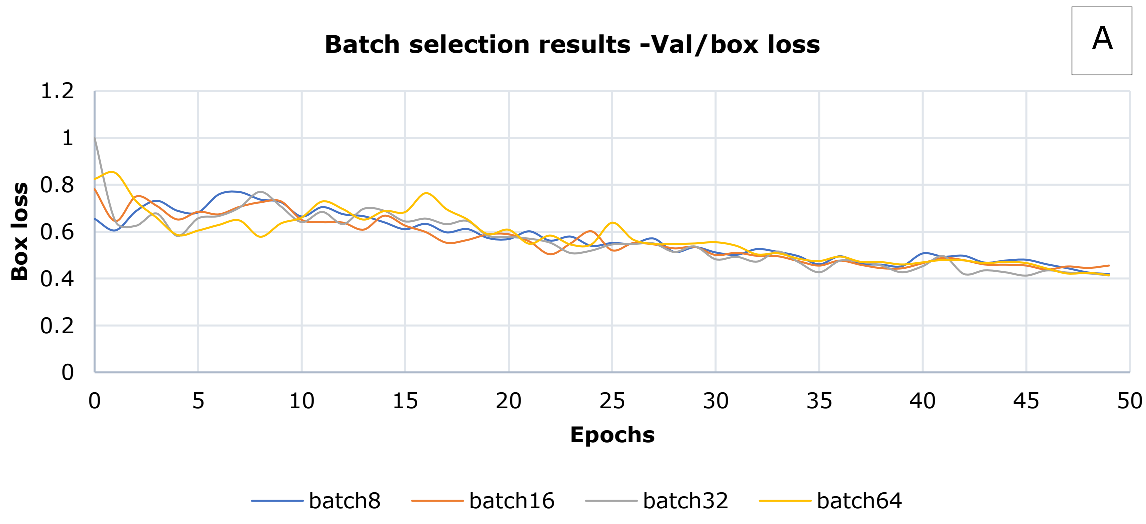
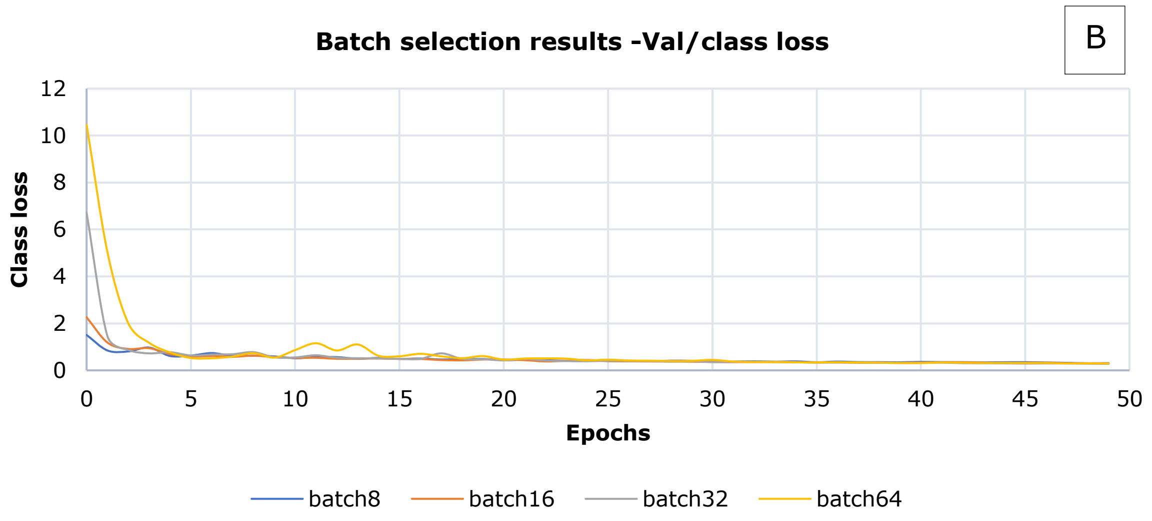


Figure A1.5 Training losses for batch selection trainings
A – Box loss; B – class loss; C – DFL loss





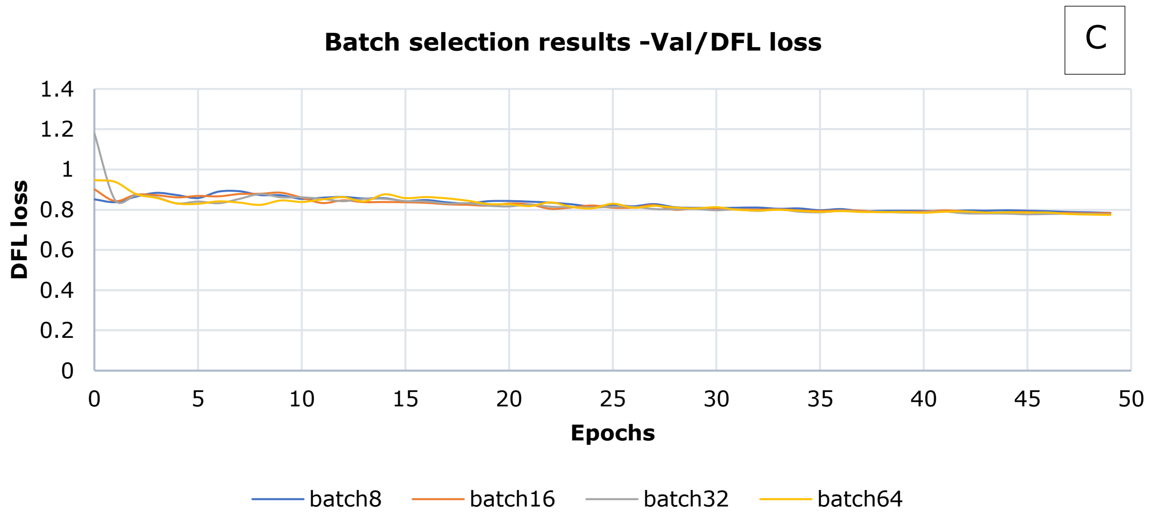


Figure A1.6 Validation losses for batch selection trainings
A – Box loss; B – class loss; C – DFL loss

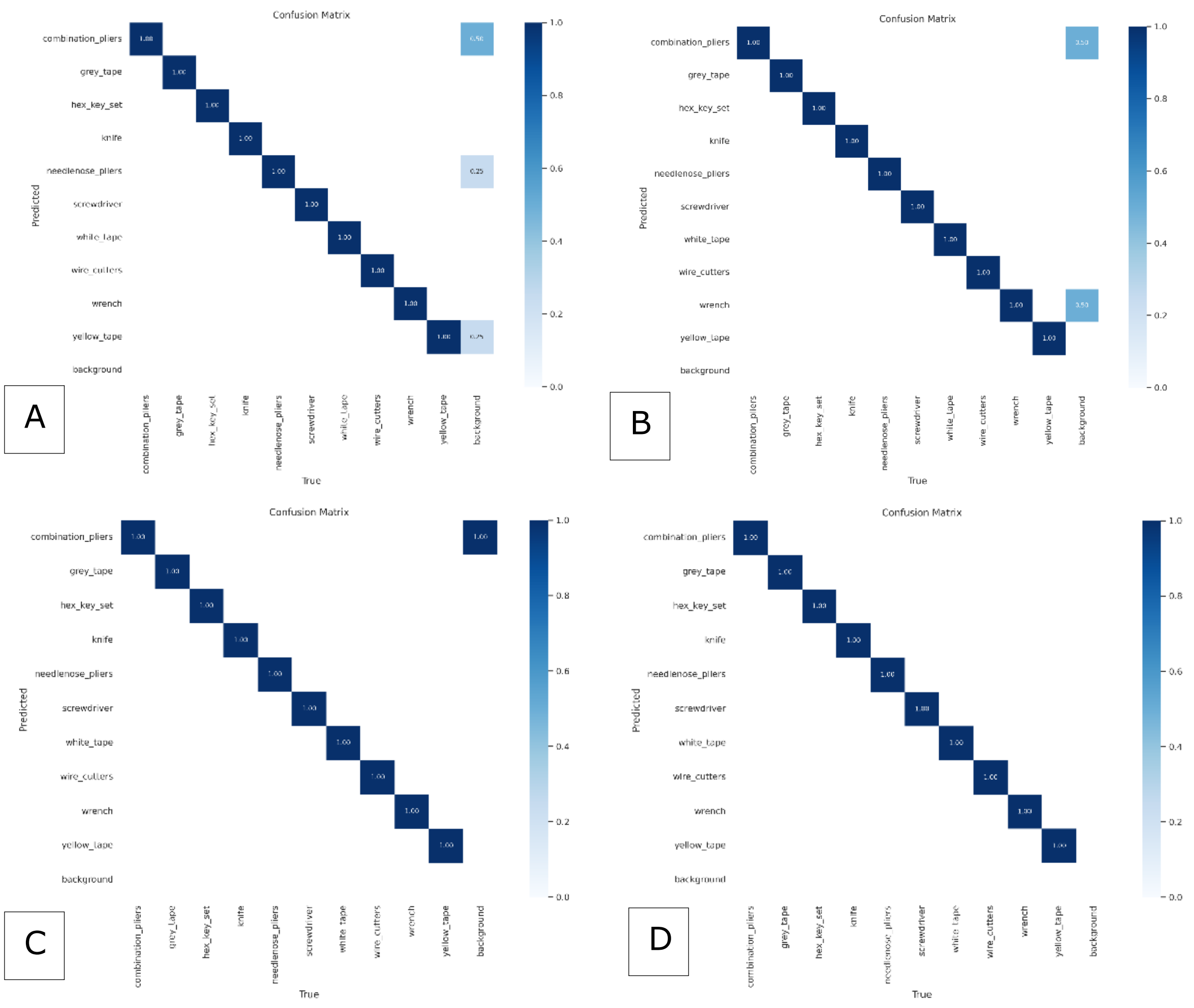


Figure A1.7 Confusion matrices for batch selection trainings
A – Batch 8; B – batch 16; C – batch 32; D – batch 64

A1.2 Results for batch size 32 and 250 epochs

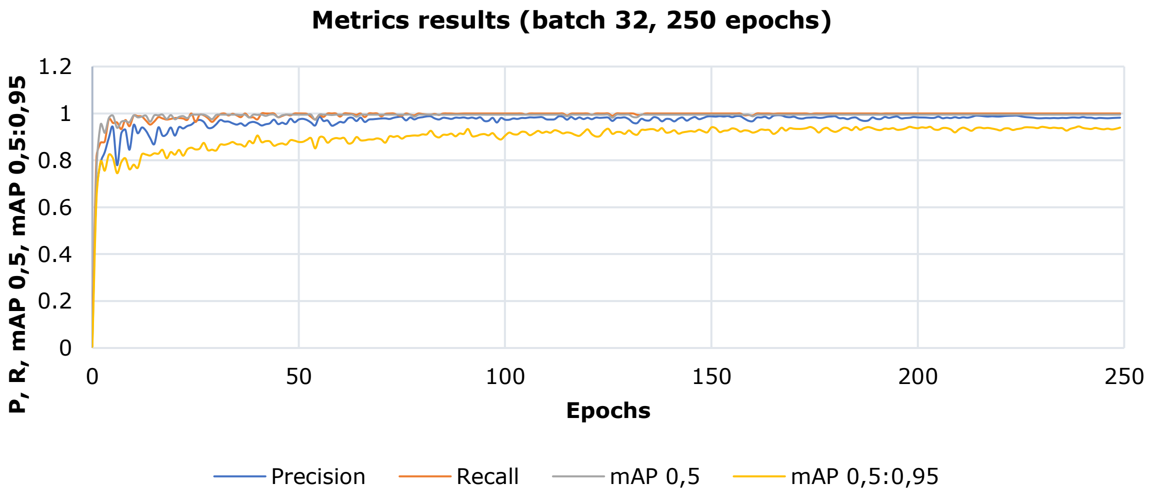
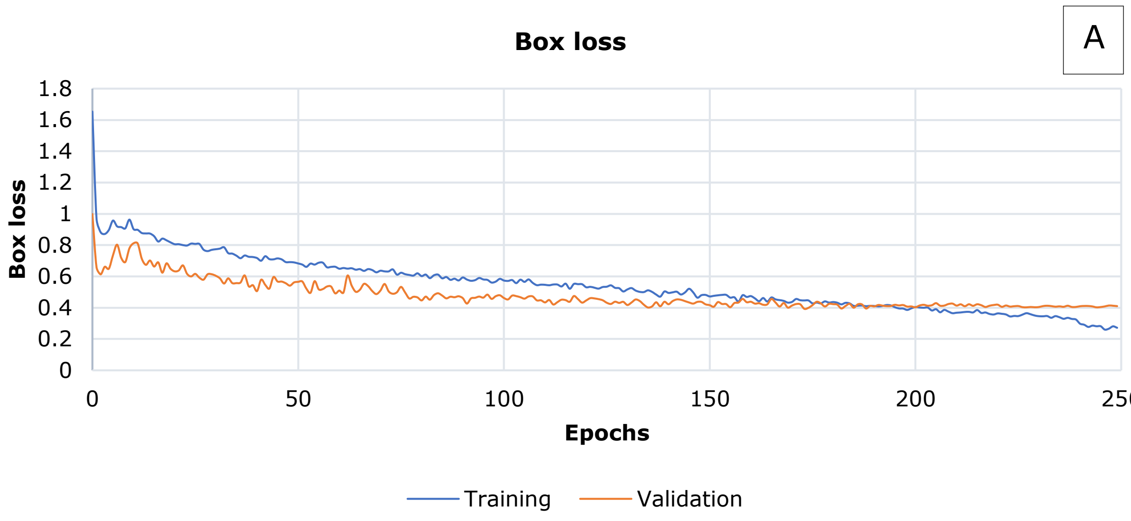
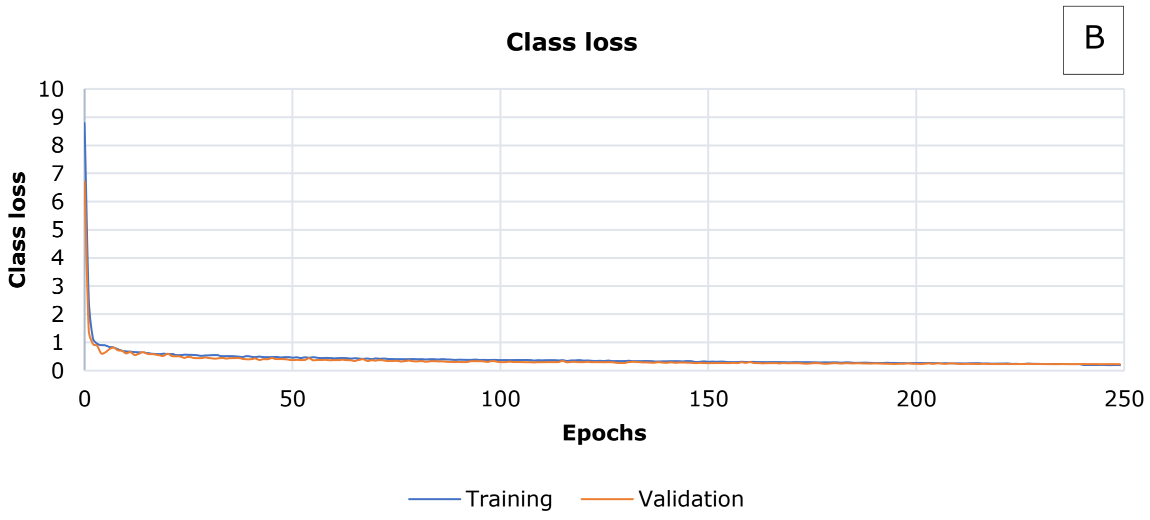


Figure A1.8 Metrics results (batch 32, 250 epochs)





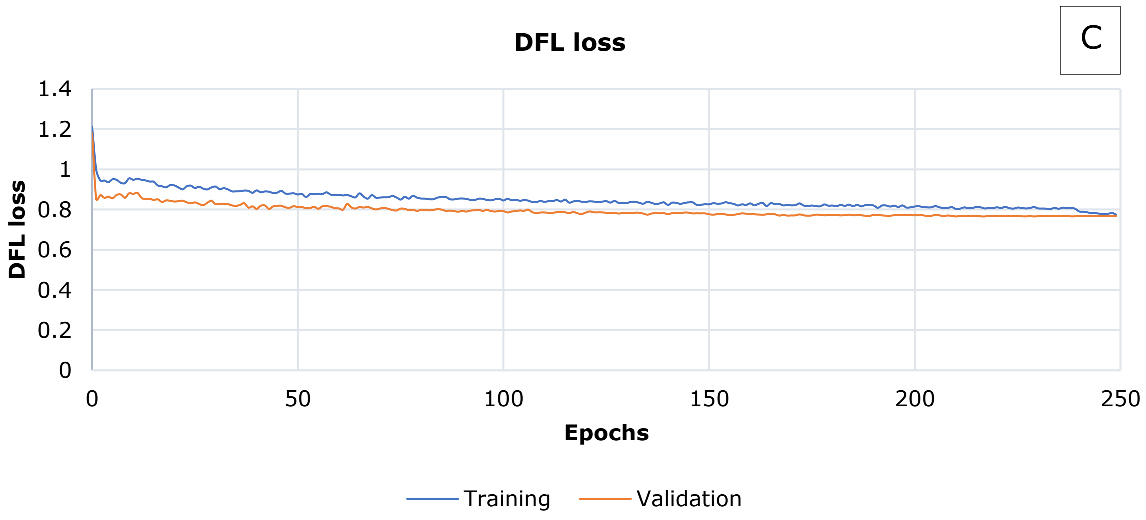


Figure A1.9 Training and validation losses (batch 32, 250 epochs)
A – box loss; B – class loss; C – DFL loss

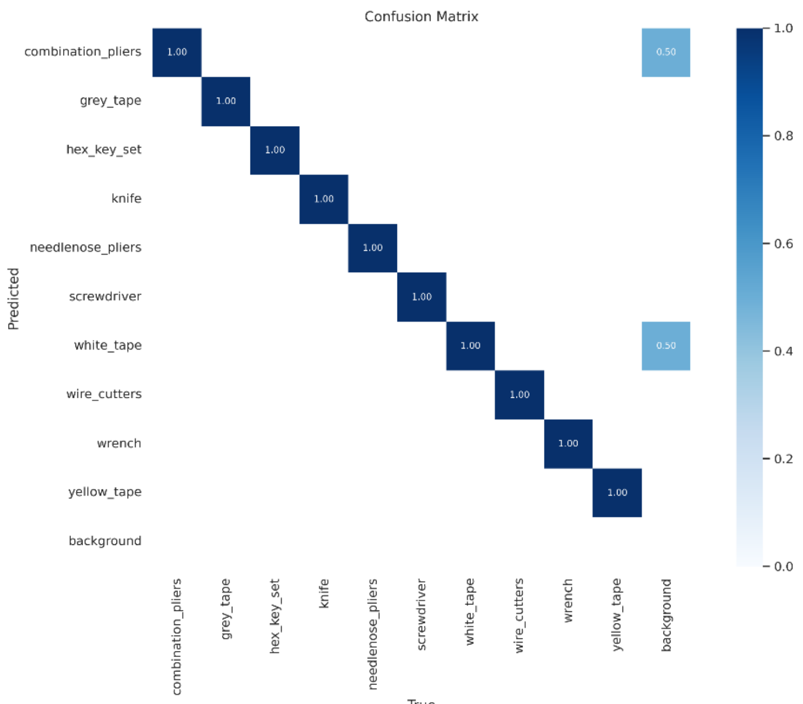


Figure A1.10 Confusion matrix (batch 32, 250 epochs)