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

MOTION ANOMALY DETECTION FOR INDUSTRIAL EQUIPMENT

TÖÖSTUSSEADMETE SEIRE KASUTADES LIIKUMISPÕHIST ANOMAALIA TUVASTAMIST

MASTER THESIS

Student:	Robert Reiska
Student code:	221937MAHM
Supervisor:	Daniil Valme, Early Stage Researcher

Tallinn 2024

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.

13. May 2024

Author: */signed digitally /*

Thesis is in accordance with terms and requirements.

13. May 2024

Supervisor: */signed digitally /*

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.*

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student:	Robert Reiska, 221937
Study programme:	MAHM, Mechatronics
Supervisor(s):	Daniil Valme, Early Stage Researcher
Consultants:	

Thesis topic:

(in English) Motion anomaly detection for industrial equipment

(in Estonian) Tööstusseadmete seire kasutades liikumispõhist anomaalia tuvastamist

Thesis main objectives:

The objective of this thesis is to develop a standalone device that can:

1. Observe the motion in industrial equipment
2. Construct a model of the equipment using the gathered motion data.
3. Detect anomalies in the equipment by observing and comparing the data to the constructed model.

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Perform background research of the topic	01.02.2024
2.	Buying hardware and construct a device	01.03.2024
3.	Writing software	01.04.2024
4.	Testing on benchmarks	07.04.2024
5.	Constructing first draft of the thesis	22.04.2024
6.	Thesis submission	13.05.2024

Language: English **Deadline for submission of thesis:** 13.05.2024

Student:	Robert Reiska	(signed digitally)	"13" May 2024
Supervisor:	Daniil Valme	(signed digitally)	"13" May 2024

Head of study programme:

Anton Rassõlkin (signed digitally) "......." May 2024

CONTENTS

PREFACE

This work was initiated by the author to assist personnel in reducing the debugging and repair times for industrial equipment. The objective was to develop a device capable of autonomously monitoring and analysing equipment, thus eliminating the need for specialized personnel for this specific task. The continuation of this thesis will be pursued separately, focusing on developing a marketable product.

The research and completion of this project were conducted under the supervision of Daniil Valme, Early-Stage Researcher, at Tallinn University of Technology. His guidance was instrumental in the realization of this work. The testing phase was carried out at Proven OÜ, where the author is currently employed. I would like to sincerely thank Daniil Valme for his dedicated help and the personnel at Proven OÜ for their support and the provision of equipment.

LIST OF ABBREVIATIONS AND SYMBOLS

AI - Artificial Intelligence

ANN - Artificial Neural Network

CUDA - Compute Unified Device Architecture

CPU - Central Processing Unit

CSI - Camera Serial Interface

GPU - Graphics Processing Unit

GPIO - General-Purpose Input/Output

HDMI - High-Definition Multimedia Interface

IoT - Internet of Things

LCD - Liquid Crystal Display

LSTM - Long Short-Term Memory

MLP - Multilayer Perceptron

MSE - Mean Squared Error

PU - Processing Unit

RAM - Random Access Memory

UI - User Interface

USB - Universal Serial Bus

1 INTRODUCTION

The manufacturing industry stands as one of the cornerstones of any economy. In Estonia, for instance, manufacturing accounted for 15% of the Gross Domestic Product in 2022, making it the largest contributor, followed by wholesale and retail trade at 12%, and real estate activities at 10% [1]. This sector is in a constant state of evolution, with automation emerging as one of its most significant trends. Increasingly, machinery is replacing human labour, offering cost-effectiveness, scalability, and easier management.

Within the European Union, machinery is defined by Directive 2006/42/EC as "an assembly, fitted with drive system other than directly applied human or animal effort, consisting of linked parts or components, at least one of which moves, and which are joined together for a specific application" [2]. This broad definition encompasses various equipment critical to the production line. As companies seek to add, upgrade, and maintain their hardware, a specialized industry has emerged to design and maintain these machines.

A significant aspect of machine building and maintenance involves the observation of machine operations to confirm their operational condition during development, testing, installation, or deployment. During system performance testing, the operator's task is to evaluate the system's performance using the appropriate machinery documentation, as well as monitoring the physical parameters of the system using measuring instruments, as well as using sensors. Without measuring equipment, the operator can evaluate the work visually using his eyes, as well as assess the level of vibration and associated noise.

Automation of the machinery condition monitoring is an important task, as scaling up observation becomes challenging due to factors such as the need for multiple observation positions, time constraints, maintaining quality, and ensuring safety. One potential solution to the scalability problem is the use of cameras to record machine operations for later analysis by experts. Cameras can access small and hazardous positions where humans cannot, record at higher frame rates and resolutions, and capture a wider spectrum of information, thus enhancing data collection. While advancements in technology offer solutions to certain challenges in machine observation and analysis, addressing scalability and minimizing downtime remain ongoing concerns for the industry.

This thesis proposes an enhancement in the use of cameras for industrial equipment monitoring through the development of a portable device that incorporates artificial intelligence (AI)-based machine vision for real-time motion analysis. The goal of this device is to automate the observation process, thereby reducing the time and manpower typically required for such tasks.

The device comprises three main components: a camera, a processing unit (PU), and a user interface (UI) in the form of a touchscreen. The camera captures footage of the machine during its normal operations and transmits the data to the PU. The processing unit then performs three primary tasks: data collection, machine model creation, and analysis. Firstly, the PU utilizes image processing techniques to identify the moving parts of the machine and measure their relative positions. This data is used to construct a model of the machine using a neural network in the PyTorch framework. Once the model is established, real-time measurements can be compared against it to conduct the analysis. If inconsistencies between the measurements and the predictions are detected, the PU can notify the user through the UI of a potential point of interest. The user can access the data gathering and analysis results through the user interface in real-time, allowing them to monitor trends over time and intervene as necessary.

As a result of this thesis, a prototype of a real-time machine monitoring device has been developed. This device is designed to detect and issue warnings about anomalous motion behaviour in machinery, such as parts binding, breaking, or moving unexpectedly. It was tested on a sample machine in two different environments: bottles moving on a conveyor belt and an indexing wheel turning. The conveyor belt scene demonstrated clearer and simpler movements, whereas the indexing wheel scene displayed more complexity. The primary objective is to enhance the device's autonomy concerning model creation and machine observation, aiming to make it user-friendly and efficient in operation.

The main body of the thesis is divided into five chapters. The second chapter explores existing solutions within the industry, as well as key concepts in machine vision and neural networks relevant to this thesis. The third chapter details the design approach and methodology of the device. The fourth chapter is dedicated to the hardware design of the device, while the fifth chapter covers its software design. The sixth chapter presents and discusses the results obtained from the thesis.

Keywords: Industrial, monitoring, AI, vision, master thesis

2 LITERATURE REVIEW

2.1 Machine monitoring

2.1.1 Background

Machine health monitoring is vital in today's industries, as it helps keep machines running efficiently and safely by spotting problems before they cause breakdowns or accidents. This method saves money by reducing the need for expensive repairs and unplanned downtime, making maintenance more effective. For example, an hour of downtime costs more than $2M for an automotive company in 2022 [3]. It also supports better decision-making, improving overall performance and sustainability by using data to understand and predict machine behaviour. This approach ensures that machines last longer, work better, and pose fewer risks, which is key for any business aiming to stay competitive and productive.

Various techniques have been developed to monitor the condition of machines, each with distinct approaches and diagnostic capabilities:

- Visual inspection uses visible light to detect abnormalities such as corrosion, misalignment, and structural damage, offering a quick and cost-effective assessment method.
- Vibration analysis is a field of structural dynamics, which analyses phenomena like resonance in structures and the effects of coupling structures together.
- Oil analysis is a technique that examines the lubricants in machinery to understand wear and tear inside engines, gearboxes, and hydraulic systems.
- Electrical Signature Analysis assesses the health of electric motors, generators, and overall power quality by analysing the electrical characteristics of machines.
- Performance monitoring tracks key performance indicators to optimize machine efficiency and detect faults, linking to the machinery's effectiveness.
- Thermal imaging and infrared analysis use thermal energy variations to assess machine health, ideal for monitoring electrical systems and mechanical components.

In this thesis, visual inspection is selected as the monitoring technique due to its widespread use, real-time applicability, cost-effectiveness, and simplicity of deployment. Unlike thermal imaging, visual inspection incurs low equipment costs and can be applied in real-time settings. Additionally, it offers the advantage of being deployable independently, without the necessity for integration into existing systems.

2.1.2 Existing solutions

In the realm of visual inspection tools, a large selection of products is available, each tailored for specific tasks. For instance, Cognex’s In-Sight D900 vision system [6], along with many other inspection camera systems, focuses on quality control along the production line. These systems typically capture an image of the object to assess its acceptability. However, they do not incorporate dynamics into their analyses. Single-shot applications do not account for the varying velocities and trajectories exhibited by industrial machinery, rendering them unsuitable for tasks that require adaptation to such conditions. Teledyne FLIR also offers a wide range of cameras and AI analysis tools [4], but like many other similar product, require specialized knowledge for their implementation.

There are also tools available for machine monitoring. FourJaw offers a machine monitoring solution using the MachineLink hardware [5], which employs sensors and algorithms to identify productive and unproductive machine states. It provides real-time utilization and Everall Equipment Effectiveness metrics, energy usage, carbon footprint, automatic alerts, downtime reasons, and work booking functionalities. While useful data, it doesn’t take into account the machines actual state, just the resulting energy usage. Festo’s AX system [6] is an AI and machine learning-based platform that takes in inputs from various sensors installed onto a machine, analyses the data, and provides information about the machine's health. This platform is utilized for diagnosis, predictive maintenance, and predictive energy management, which very flexible but does not offer an off-the-shelf solution. Festo AX's implementation requires support from Festo or qualified personnel.

Clarify AI [7] provides software that combines computer vision visual inspection with predictive analytics. This software is utilized for equipment monitoring, maintenance scheduling, improving safety, and protecting expensive assets. These solutions are entirely based on their cloud software, and they do not provide the hardware for cameras. They do provide cloud computing, which can accelerate the analysis process, but this necessitates the uploading of images to their platform for processing. Therefore, an internet connection is required, and real-time processing is not feasible.

With the widespread adoption of IoT (Internet of Things) devices, monitoring machines and their operations has become commonplace in production facilities. In the field of industrial equipment monitoring, a variety of solutions are available, each tailored to meet specific needs and functions.

2.2 Image processing

Image processing plays a vital role in standardizing or generalizing image data. This involves converting images into formats that contain uniform quantities of data. The recognition process is thus a two-stage scheme: initially, images are transformed into more manageable forms with consistent data volumes; subsequently, they are classified, reducing their data content. Recognition, in essence, is a form of data abstraction where the final data bears little resemblance to the original. For example, an image of a handwritten letter 'A' which might begin as a 20x20 bit array, could be recognised a simple 7-bit ASCII representation of the letter 'A' (Figure 2.1), 1000001 in binary, which might appear as a random pattern unrelated to the original image [8].

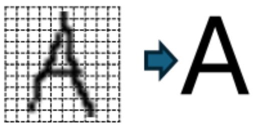


Figure 2.1 Image of a handwritten letter "A" converted to ASCII format.

The classical paradigm for object recognition involves two primary stages: preprocessing and abstract pattern recognition. During preprocessing, image processing techniques are used to suppress noise and other artifacts, as well as to regularize the image data. This creates a cleaner, more standardized set of data for the subsequent stage. The second stage involves applying statistical pattern recognition methods to extract the minimal bits of data needed to classify the object (Figure 2.2). This process reduces the vast amounts of raw image data to essential information that categorizes or identifies the object [8].

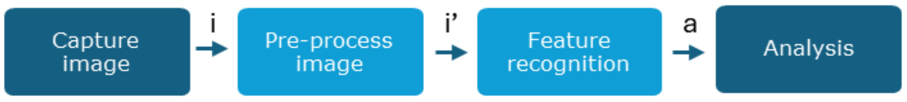


Figure 2.2 Image processing pipeline, where *i* is the captured image, *i’* is the pre-processed image and *a* are the abstract features.

2.2.1 Preprocessing

Image preprocessing plays a crucial role in the field of computer vision, serving as the foundation for subsequent analysis and model performance. Before delving into tasks such as object detection, recognition, or tracking, images must be conditioned to enhance their quality and ensure consistency across datasets [8]. This step is for removing noise, normalizing sizes, and converting formats, which increases the effectiveness of later algorithms, which extract meaningful information. By standardizing the input data, preprocessing ensures that models are not misled by irrelevant variations, thereby improving accuracy, reducing computational complexity, and facilitating more reliable interpretations of visual data.

Scaling down an image significantly reduces the computational load by decreasing the number of pixels that need to be processed. When an image is scaled down by a specific factor, the total number of pixels in the image is reduced by the square of that factor. For instance, an image with dimensions of 1920x1080, which contains 2,1 MP. If this image is scaled down by a factor of 2, the new dimensions become 960x540. This results in a total of about 0,5 MP. Thus, by reducing the image size in this manner, the computational requirements for processing the image are lowered.

Converting an image grayscale from RGB (Red-Green-Blue) reduces the computational load. An RGB image consists of three colour channels, with each channel representing a colour intensity value for every pixel. By converting to grayscale, these three channels are merged into a single channel, condensing the colour information into shades of grey that represent the luminance of each pixel. This reduction from three channels to one reduces the amount of data to be processed by a factor of three, decreasing the computational requirements for subsequent image analysis tasks.

Removing noise is required in suboptimal lighting conditions, as it enhances image quality by minimizing random variations in brightness or colour. Filtering or blurring techniques are utilized for this purpose, each tailored to address specific types of noise [8]. Gaussian filtering averages pixel values using a Gaussian kernel, softening the image, and reducing Gaussian noise. Median filtering, on the other hand, replaces each pixel's value with the median of its neighbours, making it useful for eliminating salt-and-pepper noise while preserving edges. Mode filtering replaces pixel values with the most frequent value among neighbouring pixels, suitable for categorical data. These methods improve image clarity, preparing them for more analysis and processing.

2.2.2 Feature recognition and movement detection

Background subtraction is a widely used technique in image processing for detecting moving objects in a scene by assuming that the background exhibits regular, predictable behaviour. This method involves creating a statistical model of the scene's background and then identifying any deviations from this model as potential intrusions. A probability density function is assigned to each pixel, allowing for the individual assessment of whether a pixel in a new image belongs to the background based on its fit with the density function. For static scenes, a simple model could be an image of the scene without any intruding objects. The next step involves estimating the variances in pixel intensity levels, which are necessary because these variances can differ significantly from one pixel to another [9]. In this thesis, a MOG2 [9] and a KNN [10] algorithms were used to determine these density functions.

The outcome of the background subtraction process is a grayscale image, in which the brightness levels indicate areas of motion. To minimize noise and isolate most significant areas, thresholding is applied. In this process, pixels with values above a specified threshold are set to the maximum value, while those below the threshold are reduced to zero, resulting in a binary mask. This mask enables the tracking of object movements across frames, enabling the system to identify and monitor moving elements (Figure 2.3).

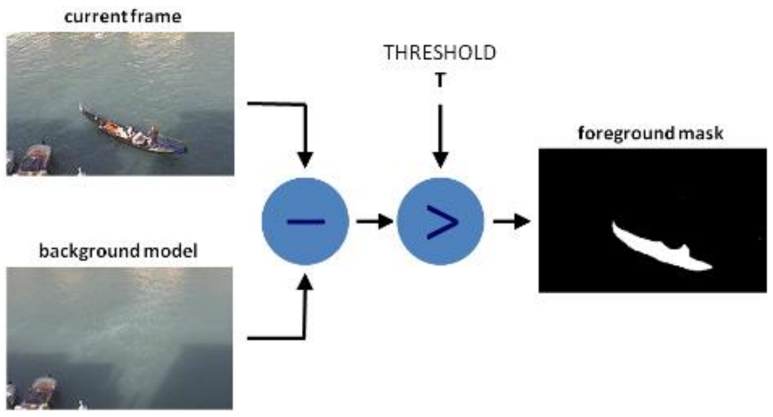


Figure 2.3 Example of background subtraction. The background model is subtracted from the current frame, then a certain threshold is applied, which calculates the foreground mask [11].

With standard thresholding techniques, we can detect which parts of an image are moving, but not the direction of their movement. To address this, optical flow can be calculated. Optical flow is a concept in motion analysis that computes a motion vector field across an entire image. This is done by analysing local pixel variations over time, assuming that changes in motion are due to alterations in image intensity[8].

Optical flow detects movement and captures the direction of that movement. To visually represent this direction on screen, a colour wheel is used. In this scheme, the hue represents the direction of the movement, while the intensity indicates its magnitude.

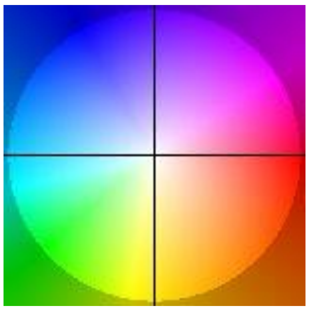


Figure 2.4 Colour wheel demonstrating the direction and magnitude of movement [12]. The closer the colour is to the edge of the circle, the larger the movement.

Once a mask is collected from the background subtraction, the objects need to be identified. These objects are regions of pixels, which are close to each other. To remove noise and isolate objects, the mask needs to be processed. This is done through a combination of erosion and dilation.

Morphological erosion is a process used in image processing to reduce the size of foreground objects in a binary image. It works by comparing a structuring element, a small shape or template, to the neighbourhood of each pixel in the image. If the structuring element fits completely within the foreground object at a specific location, the pixel under the structuring element's origin is set to the foreground value; otherwise, it's set to the background value. This operation effectively erodes away the boundaries of foreground objects, making them smaller and removing small, isolated objects (Figure 2.5) [13].

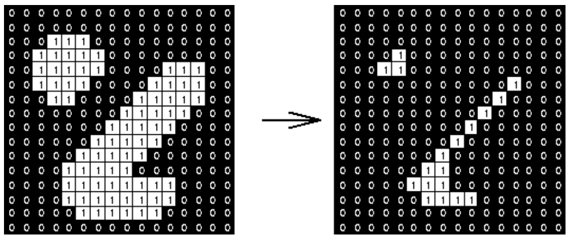


Figure 2.5 Effect of erosion using a 3×3 square structuring element [13].

Dilation is an image processing technique used to expand the size of foreground objects in a binary image. It involves the use of a structuring element, which is a predefined shape or template, to probe and modify the image. During dilation, the structuring element is centred over each pixel, and if any part of the structuring element overlaps with a foreground pixel, the pixel under the centre of the structuring element is set to the foreground value. This operation enlarges the boundaries of foreground objects, fills in small holes within those objects, and can join nearby objects (Figure 2.6) [14].

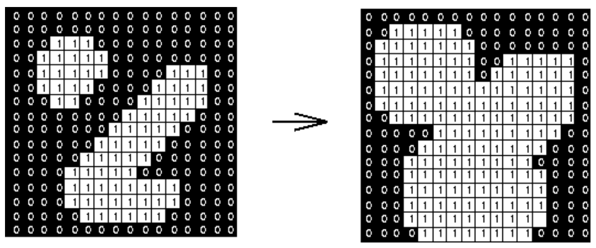


Figure 2.6 Effect of dilation using a 3×3 square structuring element [14].

By applying both erosion and dilation, small and irrelevant particles detected during motion detection are removed, enhancing the clarity of object identification within the mask. After the noise has been removed, the binary image contours can be detected and object coordinates can be found [15].

When objects are detected, tracking their positions using object tracking techniques is required. This step is required because, for the model of the machine to function, it needs to understand the contextual information regarding the movement of objects, specifically from where and to where they are moving. Since it is already known, where the object was in the last frame and where an object is in the current frame, there are several algorithms available, that perform object tracking.

2.3 Modelling

2.3.1 Approach

To determine whether the machine is operating correctly, it's required to define what constitutes correct behaviour. This definition could be established manually by the device user or automatically. To enhance the device's usability, automatic identification of correct behaviour is preferred. To develop a model that represents machine behaviour, its underlying mechanics of the machine itself should be understood. This understanding can be approached through three primary modelling strategies [16]:

White Box Modelling (also known as First Principles Modelling) seeks to construct the model by starting from the foundational "ground truths" or basic principles governing the system's operation. This method relies on established mathematical relationships and known physical laws to simulate the system's behaviour.

Grey Box Modelling represents a middle ground, wherein the model is built upon a combination of known relationships and empirical data. While some system parameters and relationships are assumed based on theoretical knowledge, others are derived through observation and estimation, blending theory with empirical insights.

Black Box Modelling takes a purely empirical approach, eschewing any presuppositions about the system's internal workings. In this method, the model is developed entirely from input-output data, using statistical or machine learning techniques to infer the system's behaviour without any assumed knowledge of its inner mechanisms.

If the device is used on previously unknown machines and for user convenience, it is needed, that the model is identified mostly automatically, then it is necessary to define the machines model as a black box model. These different black box modelling options will be explored in the next section.

2.3.2 Anomaly Detection

An anomaly, in the broadest sense, is an outcome or value that deviates significantly from what is expected or considered normal. It is not merely an outlier, which might represent a value that's extreme yet still within the bounds of what's considered normal behaviour or variation. Instead, an anomaly suggests something more significant – it could indicate an error, an unusual event, or a condition that warrants further investigation because it falls outside the normal operating parameters or expected patterns.

The definition of what constitutes an anomaly can vary widely depending on the context or the specific system being observed. For example, in the context of observing swans by a lake, a black swan in a population predominantly consisting of white swans would be considered an anomaly (Figure 2.7). This is because, based on prior observations, the expectation has been set that swans in that location are white, making the appearance of a black swan a significant deviation from the norm [17].

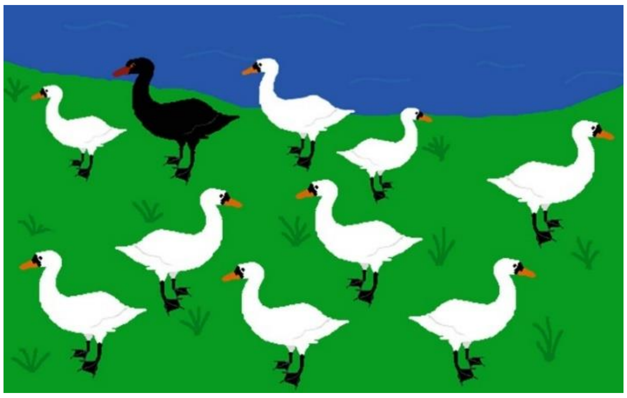


Figure 2.7 A black swan appears next to white swans. [17]

Anomaly detection is the process in which an algorithm identifies certain data or data patterns to be anomalous. Anomaly detection plays an important role across various sectors by identifying deviations from normal patterns in data, improving proactive interventions and quality assurance. In the industrial context, this technology is used for maintaining high standards of production, safety, and efficiency [17].

Anomaly detection can be categorized into three styles: Supervised, where models are trained with labelled data distinguishing normal from anomalous points; semi-supervised, utilizing partially labelled data, often only marking normal data to help models identify anomalies by deviation; and unsupervised, which does not rely on labelled data, expecting models to learn the distinction between normal and anomalous data through the data structure itself [17].

2.3.3 Artificial neural networks (ANNs)

Deep learning, a subset of machine learning, focuses on artificial neurons, inspired by biological neurons, to process information. It involves structuring neural networks as sequences of layers, where each layer consists of neurons that process inputs and produce outputs. These networks have an input layer that receives training data, output layers that make predictions, and intermediate hidden layers that handle complex data transformations (Figure 2.8). This structure enables the ANN to learn complex patterns through data analysis, without knowing the actual patterns or the underlaying principles. Because of this feature, ANNs are suitable for this thesis.

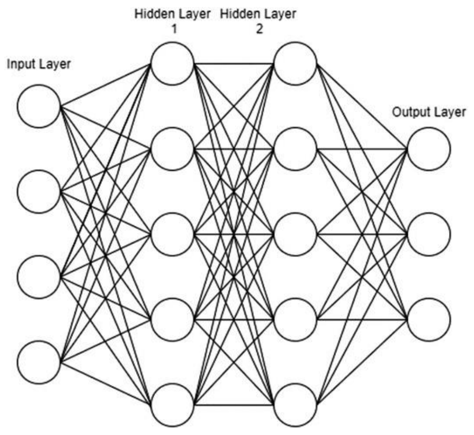


Figure 2.8 Architecture of a neural network with two hidden layers [17].

The forward pass is the process where input data is fed into the network and passed through each layer sequentially, from the input layer to the output layer. During this pass, the network applies weights, biases, and activation functions to compute the final output. This step is critical for making predictions based on the input data.
The backward pass, or backpropagation, occurs after the forward pass and involves calculating the gradient of the loss function with respect to each weight and bias in the network by moving backward from the output layer to the input layer. This process uses the chain rule to compute these gradients, which are then used to update the model's parameters to minimize the loss.

The loss, or cost, function quantifies the difference between the predicted outputs of the neural network and the actual target values, thereby measuring the network's performance. A lower loss indicates a better model. Mean Squared Error (MSE) is particularly widely used because it emphasizes larger errors more significantly. MSE can be represented by the following formula [17]:

$$\text{MSE} = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N} \tag{2.1}$$

where y_i – target value for the i^{th} sample,

$\hat{y}_i$ – prediction made for the i^{th} sample,

N – number of samples.

An optimizer is an algorithm or method used to change the attributes of the neural network, such as weights and learning rate, to reduce the losses. Optimizers guide the training process by updating the network's weights in response to the output of the loss function. Examples include Gradient Descent, Stochastic Gradient Descent (SGD), Adam, and RMSprop. Different optimizers may perform better in different types of neural networks and problem settings [17].

2.3.4 Anomaly detection ANNs

Anomaly detection in (ANNs) encompasses a range of techniques aimed at identifying unusual patterns that deviate from the norm within datasets. These techniques use the computational power and flexibility of different types of ANNs to model complex data distributions and detect anomalies. In this thesis, 3 types of ANNs are used due to their simplicity and low memory requirements: Multilayer Perceptron, Autoencoders and Long Short-Term Memory networks**.**

Multilayer Perceptron (MLP) Supervised Anomaly Detection involves training a type of neural network known as a Multilayer Perceptron to distinguish between normal and anomalous data points in a supervised learning context. In this approach, the MLP is trained on a labelled dataset where each instance is marked as either normal or an anomaly. Once trained, the MLP can predict the class (normal or anomaly) of new, unseen data points based on the patterns it has learned [17].

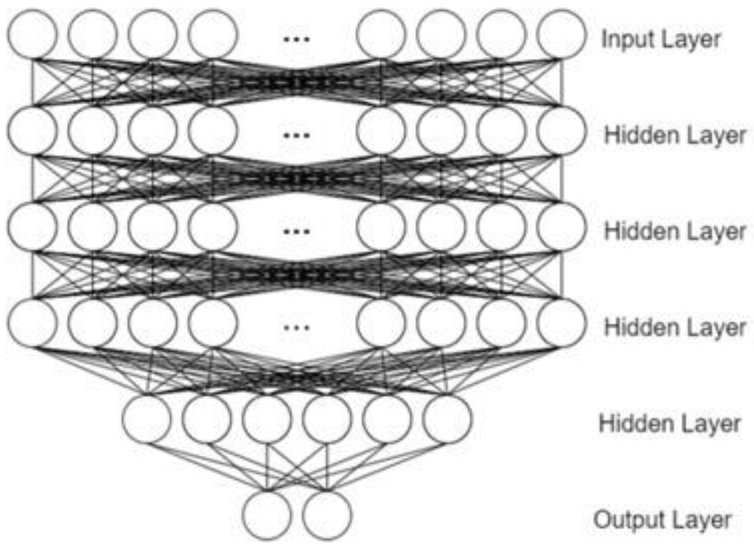


Figure 2.9 Example concept of a multilayer perceptron [17].

Autoencoders are a type of neural network used for unsupervised learning. They work by compressing input data into a lower-dimensional representation in the encoder phase, and then reconstructing the output to match the original input as closely as possible in the decoder phase (Figure 2.10). The process of encoding and decoding allows autoencoders to learn representations of the data, capturing its most important features [17].

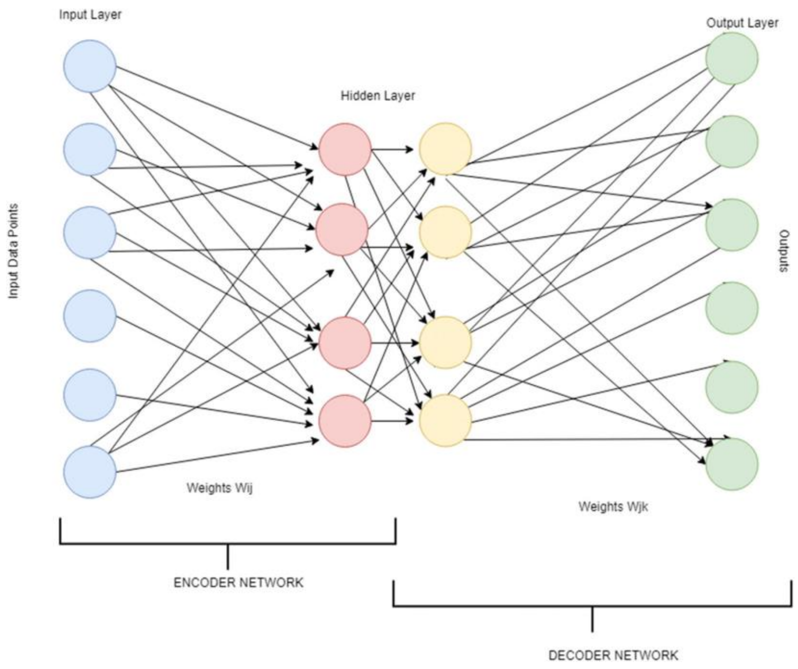


Figure 2.10 Expanded view of an autoencoder [17].

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) specialized in processing sequences of data. They excel in tasks that require learning from experience to classify, process, or predict time series where time intervals between events are unknown or variable. LSTMs are designed to overcome the vanishing gradient problem that can affect standard RNNs, allowing them to learn long-term dependencies. This is achieved through their structure of gates (input, output, and forget gates) that regulate the flow of information (Figure 2.11). These gates control whether to retain or discard information, making LSTMs capable of capturing long-term patterns in data sequences [17].

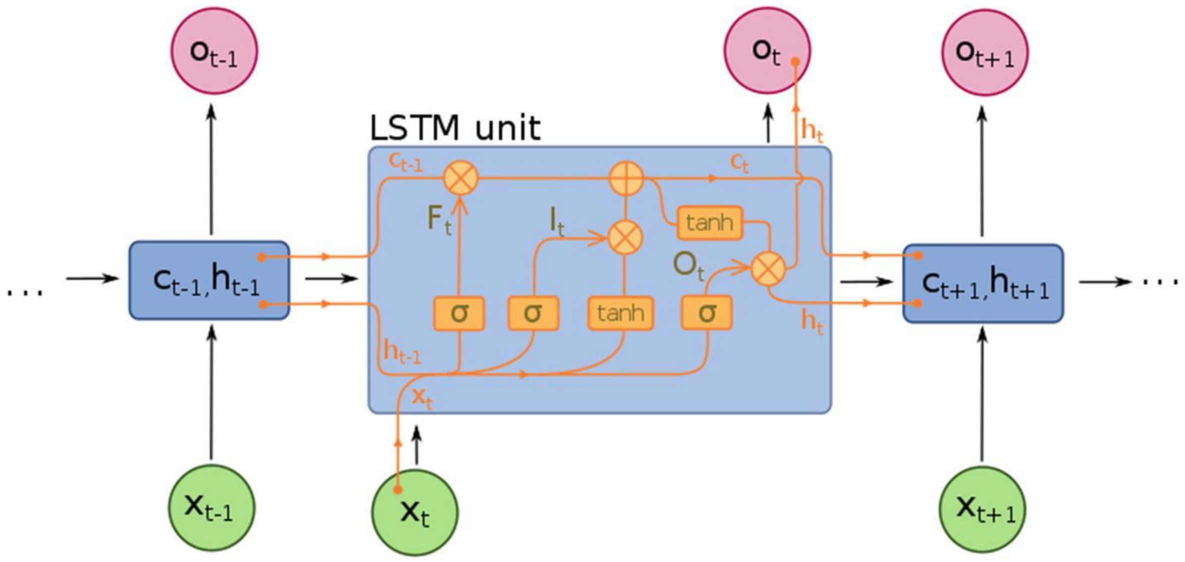


Figure 2.11 Example of an LSTM unit. *x* represents the input, *o* denotes the output, and *t* indicates the timestep. Between each unit, both long-term and short-term memory information is shared, while the internal weights remain consistent across all units [17].

2.4 Literature review summary

Machinery equipment monitoring systems are crucial part of the modern factory and are growing trend in the industry. Numerous providers offer machine monitoring services, yet these often require specialized personnel for implementation and are not available off-the-shelf. Each solution needs to be custom-tailored for specific tasks, which, while beneficial to the customer, results in higher costs. Additionally, most manufacturers require installation of extra equipment to the manufacturing line, as there are no portable devices readily available. This situation highlights a need for a ready-made device, capable of visually monitoring machinery without extensive setup or customization.

So, the devices Cost-effectiveness is a primary consideration in the development of the device. The aim is to select affordable components and technologies that do not compromise the system's effectiveness. This is achieved by integrating open-source software, which reduces licensing costs, and by selecting cost-efficient hardware components. Additionally, the design minimizes the need for high processing power, which typically drives up costs, ensuring that the system remains economical without sacrificing functionality.

Existing devices require hardware integration into the machine, which makes the integration process time consuming and expensive. Portability and Low Weight are crucial for the adaptability of the device across various industrial environments. The design focuses on utilizing compact and lightweight components that can be easily transported and installed. This flexibility enables the device to be seamlessly integrated with a wide range of industrial equipment, enhancing its applicability and convenience.

Most of the solutions assume prior expertise in the field of AI or machine vision. Thus, ease of use is another critical design objective, aimed at ensuring that the system is accessible to users of all technical skill levels. The system has a user interface that requires minimal training, making it straightforward for operators to use and an algorithm that.

To extract the required data for anomaly detection, the image processing pipeline can be divided into the following stages:

- Image Preprocessing - In this phase, images are transformed into a standardized format in terms of size and colour, and noise is removed to prepare them for analysis.

- Image Feature Extraction - During this phase, moving objects are detected and tracked using various motion detection algorithms such as KNN background subtraction, MOG2 background subtraction, and optical flow via the Farneback algorithm.

Although there is no publicly available information about the algorithms used in commercial solutions, anomaly detection typically involves the application of machine learning algorithms. To detect anomalies in the motion of the observed machine, a black box modelling approach is best, as it allows for variability in the observed machinery. In this approach, an ANN is trained on a dataset representative of the machine's normal operation. Three types of ANNs are commonly used for anomaly detection: MLP, Autoencoders, and LSTM.

2.5 Thesis goals and system requirements

Based on the literature review, the goal of this thesis is to create a prototype of a real-time machine motion monitoring device. This device must be designed to detect and issue warnings about anomalous motion behaviour in the observed machinery. This prototype must fulfil the role of an assistive tool for machine developers and operators, incorporating key design principles: cost-effectiveness, portability, and ease of use. To achieve these goals, the device must have the following functionalities:

- Record machine movement information,
- Display movement information to the operator,
- Construct a model of the movement pattern,
- Compare the model to the movement information,
- Detect anomalous movement and display it to the operator.

To achieve the described goals and device functionality, The devices hardware design must incorporate the following features:

- Capability to film the machine,
- A user interface,
- Support for ANN development and deployment,
- Cost efficiency,
- Compact form factor.

Several core objectives guide the software design process to ensure the final product meets both operational needs. These objectives focus on balancing cost-effectiveness, portability, user-friendliness, and performance. Performance in anomaly detection is essential for the effectiveness of the monitoring system. The device needs to be designed to deliver accuracy and reliability in detecting anomalies under diverse operational conditions. Performance metrics such as detection accuracy, processing time, and system robustness are key criteria used to evaluate the system’s efficacy.

3 OBSERVED MACHINE

The research design for developing this monitoring device requires a fundamentally experimental approach. This methodology is chosen for its ability to control various variables that impact the system's performance, including lighting conditions, camera angles, and the specific nature of the anomalies being detected. The capability to manipulate and isolate these factors allows for the examination of how different configurations and algorithms influence the accuracy and reliability of the process.

A disinfectant bottle cap installation machine was utilized as a case study. This machine operates by transferring bottles from an input conveyor to an indexing wheel, which advances them through several steps (Figure 3.1). At the second step, a cap is placed on the bottle, and at the fifth step, the cap is securely pressed onto the bottle. Finally, in the seventh step, the bottle exits the machine via an output conveyor. The caps themselves are fed into the machine using a vibrating linear and bowl feeder and are placed onto the bottles with pneumatic a suction cup mechanism.

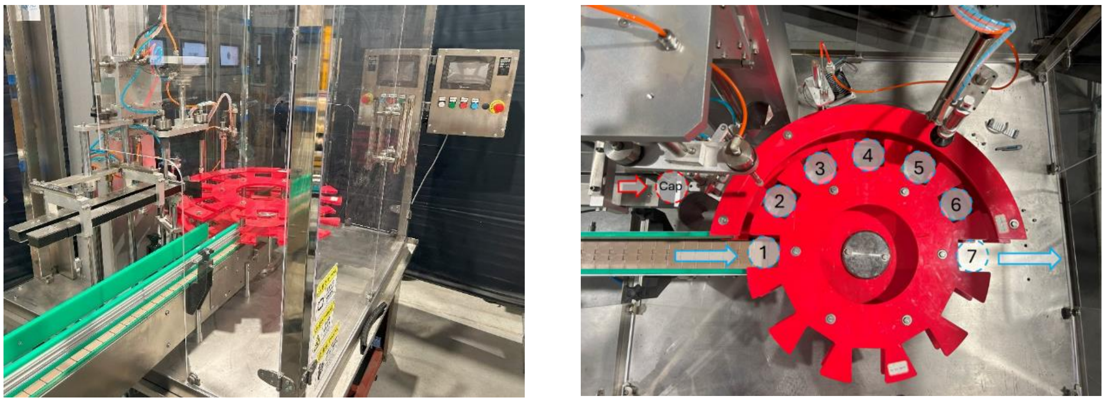


Figure 3.1 Disinfectant bottle cap assembly machine. Left hand side shows the overview of the machine, Right hand side shows top view of the indexing wheel movement direction.

To analyse the devices performance, 2 scenes were filmed. The first scene features a video clip of bottles moving on a conveyor line (Figure 3.2). The camera was positioned on the side of the conveyor belt, capturing bottles as they appeared from the left side of the screen, moved across the screen, and exited on the right side. A maximum of four bottles could appear on the screen simultaneously. This setup demonstrates optimal performance against a static, well-lit background.

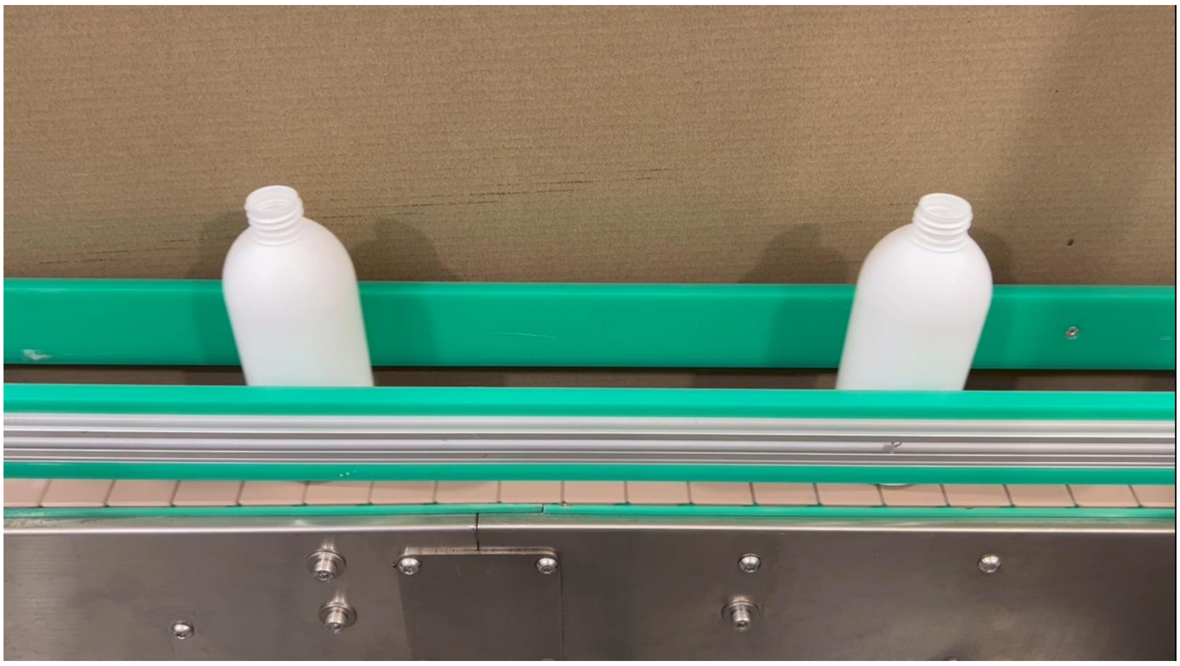


Figure 3.2 Example frame for the conveyor belt footage. Bottles are moving from left to right.

The second scene is more challenging and better represents real-world conditions. The camera overlooks the indexing wheel from the perspective of the output conveyor (Figure 3.3). In this frame, the indexing wheel advances every three seconds, and following each advancement, the cap presser moves downward. The presence of reflections from the plexiglass doors and the machine's bottom adds complexity to the motion detection due to varied movement directions, thereby increasing the model's complexity.

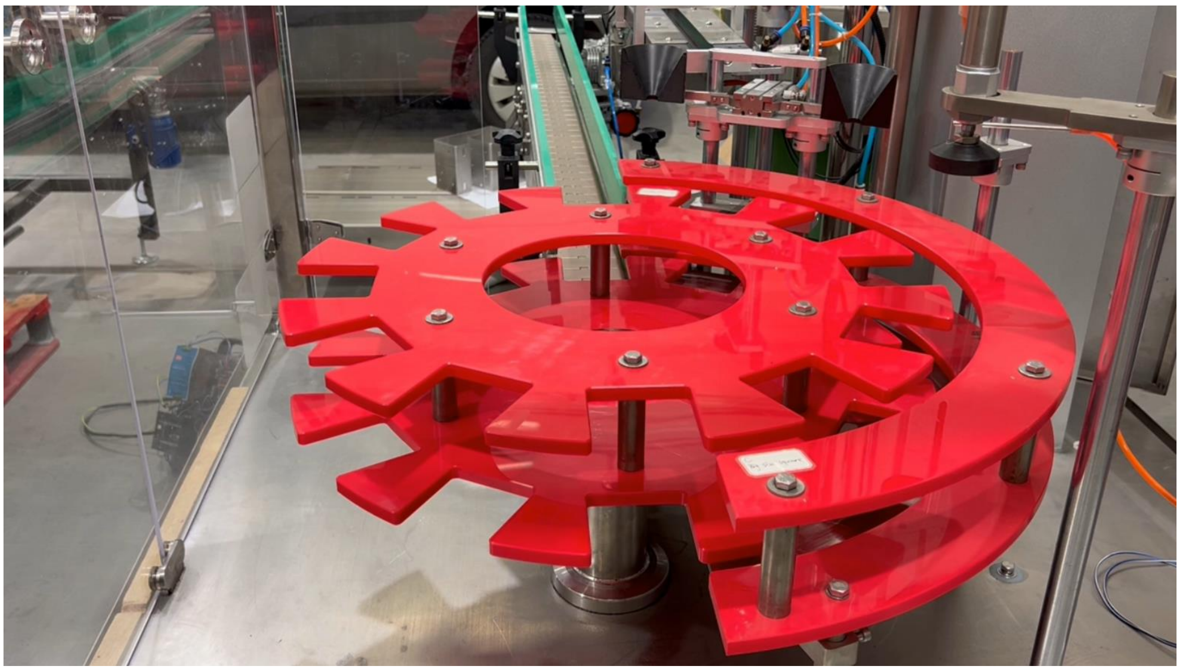


Figure 3.3 Example frame from the wheel turning footage.

4 HARDWARE COMPONENTS

4.1 Architecture

The architecture incorporates three essential hardware components (Figure 4.1): a processing unit (PU), a camera, and a user interface. The PU processes the video data captured by the camera. The user interface allows users to configure settings, view analyses, and receive alerts on detected anomalies, making the system accessible to operators of varying technical skill levels.

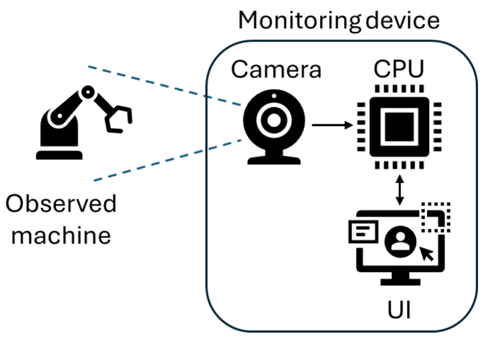


Figure 4.1 Device architecture.

4.2 Processing unit

The controller is a the most important component of the device, as it influences the selection and performance of all other hardware components. The choice of controller is determined by several key factors: price, computational capacity, interfaces, the volume of available documentation, size, and open-source compatibility with community support. Additionally, the controller must support AI functionalities, which is essential for implementing advanced machine learning algorithms. Open-source controllers offer the advantage of community-supported updates and greater flexibility in customization, which are vital for adapting the system to specific industrial needs. Given the vast array of PUs available, the focus is primarily on single-board computers that cater to these needs.

The Raspberry Pi 5 [18] is a compelling option for the controller due to its enhanced features and capabilities. This model provides significant upgrades from its predecessors, including improved computational power and expanded memory options, which are essential for processing and analysing visual data efficiently. Additionally, it offers a variety of interfaces such as USB ports and HDMI, supporting a wide range of peripherals needed for comprehensive monitoring solutions. The Raspberry Pi 5 also maintains its predecessor’s advantages of being cost-effective and having extensive documentation available, making it a highly accessible platform for developers. Its compact size allows for easy integration into industrial environments where space might be limited. These features make the Raspberry Pi 5 a robust choice for handling the demands of anomaly detection in industrial settings.

Figure 4.2 Raspberry Pi 5 board [19]

The NVIDIA Jetson Nano [20] is an excellent controller option due to its specialized features tailored for AI and machine learning tasks. It is equipped with a GPU which makes it highly suitable for processing complex visual data and performing real-time image analysis and anomaly detection. Despite its compact form factor, the Jetson Nano offers substantial computational capabilities, supporting a wide range of AI frameworks and libraries that are essential for developing advanced detection algorithms. Jetson Nano provides a variety of connectivity options including USB, HDMI, and GPIO, which facilitate the integration with other industrial components. It also boasts a strong community and extensive documentation, making it easier for developers to implement and troubleshoot their applications. Although it is slightly more expensive than some basic microcontrollers, its specialized features justify the cost for applications requiring high processing power and real-time performance, making it a robust choice for sophisticated industrial monitoring systems.

Figure 4.3 Jetson Nano board [21]

The Google Coral Dev Board is an excellent option particularly where edge computing and AI capabilities are paramount. This board is centred around the Edge TPU coprocessor, which excels in accelerating TensorFlow Lite models, enabling it to perform machine learning tasks at the edge with remarkable speed and efficiency. Key features include a small form factor, low power consumption, and substantial processing power, making it suitable for deployment in industrial environments where space and energy efficiency are concerns. The board also supports a variety of interfaces, including USB, HDMI, and GPIO, ensuring versatile connectivity with other devices and sensors. Google Coral Dev Board is supported by extensive documentation and a robust developer community, which simplifies the development process and troubleshooting. Its emphasis on machine learning and edge AI processing, combined with the support for open-source tools and software, makes it a powerful choice for implementing advanced, real-time anomaly detection in industrial settings.

Figure 4.4 Coral Dev Board [22]

While these boards appear similar, their distinct features make comparisons challenging. Each board possesses unique strengths as outlined in Table 4.1. The Raspberry Pi boasts a faster CPU and more RAM, but it has a slower GPU and lacks built-in storage. Despite these limitations, it is the most affordable option and benefits from the largest community support and extensive documentation. On the other hand, the Jetson Nano is equipped with a versatile NVIDIA GPU with CUDA support, making it highly capable for handling visual tasks and machine learning. However, it has a slower CPU and less RAM compared to the Raspberry Pi but compensates with built-in storage and robust support in machine learning applications. Lastly, the Google Coral Dev Board shares several specifications with the Jetson Nano but sets itself apart with a dedicated TPU for Edge AI computations. It is slightly more expensive and has limited community support compared to the others.

Table 4.1 Comparison between Raspberry Pi, Jetson Nano, and Coral Dev board

Name	Raspberry Pi 5	NVIDIA Jetson Nano	Google Coral Dev Board
CPU	2.4GHz Quad-core ARM A76	1.43 GHz Quad-core ARM A57	1.5 GHz Quad-core ARM A57
GPU	VideoCore VII	128-core Maxwell	Vivante GC7000Lite + Edge TPU coprocessor
RAM	8GB LPDDR4	4 GB LPDDR4	4 GB LPDDR4
Storage	- (microSD card slot)	16 GB eMMC	16 GB eMMC
Interfaces	microSD card slot Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports Dual 4Kp60 HDMI® display output 2 × 4-lane MIPI camera/display transceivers PCIe 2.0 40-pin GPIO header Dual-band 802.11ac Wi-Fi Bluetooth 5.0	microSD card slot Gigabit Ethernet 1 * USB 3.0 2 * USB 2.0 1*HDMI Type A 2x MIPI CSI-2 DPHY lanes M.2 Key E 40-pin GPIO header	microSD card slot 2x USB 3.0 uSDHC Gigabit Ethernet HDMI 2.0a MIPI DSI display MIPI-CSI2 camera input Wi-Fi 2x2 MIMO Bluetooth 4.2 40-pin GPIO header
Price	72,44 € [19]	180,20 € [23]	232,51 € [24]
Size	86 mm × 56 mm × 16 mm	69 mm x 45 mm x 45 mm	88 mm x 60 mm x 22 mm
Community support	Very large community and lots of documentation	Large community and lots of documentation. Excellent support for AI deployment	Smaller community and support

Based on this comparison, the Jetson Nano was determined to be a well-balanced choice among these three boards. It offers a powerful GPU with CUDA support, making it highly capable for machine learning and image processing tasks. Although it has a slightly higher cost than the Raspberry Pi, it is more affordable than many other specialized AI hardware options. The Jetson Nano also provides built-in storage, which is an advantage over the Raspberry Pi. While it does not have the smallest form factor compared to the other boards, its size is still compact enough for most applications. Moreover, the Jetson Nano benefits from substantial community support, particularly in the machine learning sector, making it an ideal choice for projects requiring advanced visual processing and AI capabilities.

4.3 Camera

The selection of the camera is a critical component of the hardware setup, second only to the computing platform itself. The choice of camera largely hinges on two pivotal factors: the data transfer rate and cost. Modern cameras typically possess the capability to capture high-resolution imagery at sufficient frame rates for detailed analysis. However, the data transfer rate becomes crucial when considering the limited time available for computing between frames. This rate must be high enough to ensure data is swiftly relayed to the controller, allowing timely initiation of image analysis. Additionally, the type of interface used for communication between the camera and the controller is determined by the controller's specifications, which can influence the overall effectiveness of the data transfer and integration into the detection system.

The Jetson Nano supports three primary methods for connecting a camera, each with its unique benefits and suitability for different application scenarios in industrial monitoring:

Universal Serial Bus (USB) cameras are widely available and offer a plug-and-play solution, making them easy to integrate and replace. USB connections are generally suitable for applications where moderate data rates are sufficient and cost-effectiveness is a priority. However, USB cameras may not always provide the reliability needed for high-speed data transfer.

Figure 4.5 Example of a USB camera [25].

Ethernet cameras are ideally suited for environments requiring long cable runs or extensive networking of multiple cameras. They excel in situations where cameras must be placed significant distances from their control units, such as large-scale industrial plants or outdoor monitoring areas. These cameras benefit from advanced network technologies like Power over Ethernet, which simplifies wiring and reduces installation costs by transmitting power and data over the same cable. Moreover, Ethernet cameras provide high data transfer rates and enhanced reliability, attributes crucial for applications demanding high-resolution and high-frame-rate imaging. Despite these benefits, Ethernet cameras tend to be more expensive compared to other types, reflecting their advanced capabilities and the technological sophistication they bring to sophisticated monitoring systems.

Figure 4.6 Example of an Ethernet-based camera [26].

The Camera Serial Interface (CSI) is optimal for applications demanding high data throughput and low latency, characteristics essential for effective real-time anomaly detection. CSI enables direct interfacing between the camera and the Jetson Nano’s processing core. This direct path facilitates faster image processing and reduces delay, critical factors in scenarios where timely processing can significantly impact system performance and reliability. Additionally, CSI-2 cameras are typically more cost-effective compared to other high-throughput interfaces, making them a preferred choice for embedded systems where budget constraints are considered alongside performance requirements.

Figure 4.7 Example of a CSI camera [27]

For this device, CSI was used due to its advantages in providing high data throughput and low latency, which are crucial for real-time processing applications, while keeping costs low. After determining that the CSI interface would best meet the system requirements for efficient image data handling, the Raspberry Pi Camera Module 2 was selected. The camera’s cost-effectiveness, widespread availability, and robust community support influenced this choice. The Raspberry Pi Camera Module 2 not only fits well within the device due to its compact size and lightweight design but also ensures rapid and reliable anomaly detection by facilitating a direct connection to the Jetson Nano’s processing core. Additionally, the camera’s favourable cost and strong user support network make it an economical choice for maintaining and scaling the system.

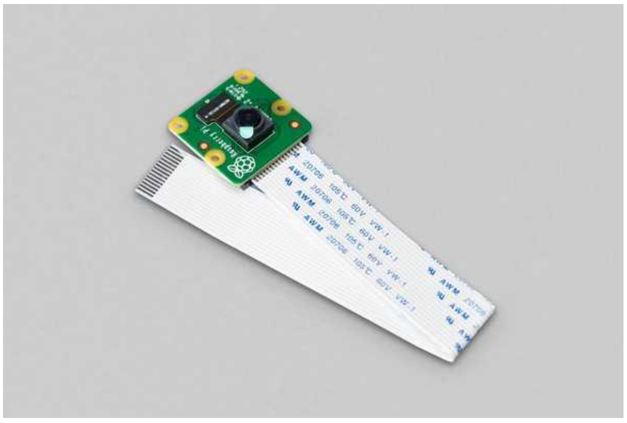


Figure 4.8 Raspberry Pi Camera Module 2 [28]

4.4 User interface display

The user interface (UI) of the device, although seemingly less critical, plays a pivotal role in user interaction and system operation. A well-designed UI is vital as it forms the primary touchpoint between the user and the system, ensuring efficient device management and ease of use.

Given the Jetson Nano's lack of built-in wireless interfaces and reliance on HDMI and USB connections, the UI needed to be in the form of a touchscreen, touchscreen interface was used, which allows for operation without extensive training. Among the available options, the 7-inch HDMI LCD touchscreen by Ingcool (Figure 4.9) was selected. This model was chosen for several reasons: its compatibility with the Jetson Nano, matching size with the device, support for essential functionalities, short shipping times, and affordability. The touchscreen’s low cost and quick integration make it an ideal, flexible solution for the current stage of the project, allowing for future modifications or upgrades as the system evolves or as user needs evolve.

Figure 4.9 LCD touchscreen [29]

4.5 Housing and assembly

The device's housing was designed in SolidWorks for 3D printing. The design principle focused on maximizing compactness while ensuring sufficient space for the hardware components, wiring, and airflow necessary for cooling. The housing consists of two main parts. The Jetson and touchscreen are mounted on the first part, the main frame, using M2.5 bolts. A Noctua 40x40x20 mm fan [30] was attached to the Jetson Nano's heatsink to enhance heat dissipation. The Raspberry Pi camera is affixed to the protective cover, which shields the Jetson Nano and the internal wiring. The two parts of the housing are friction-fitted together and secured with M3 bolts (Figure 4.10).

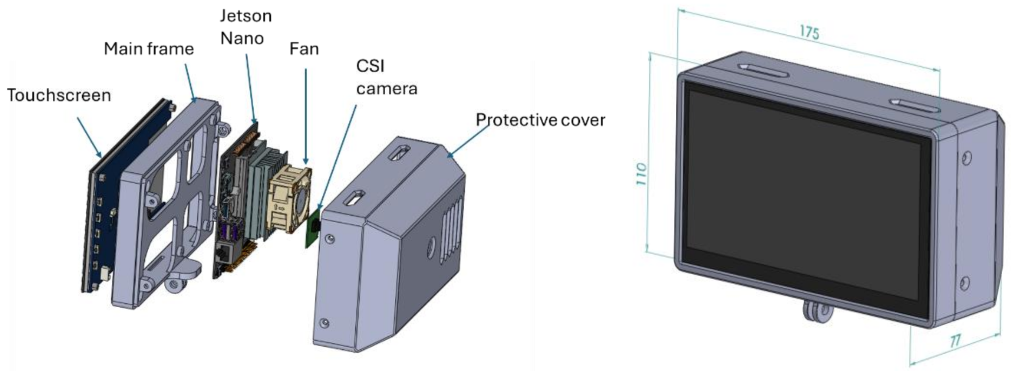


Figure 4.10 3D model of the created device, left side shows the exploded view, right side shows the general dimensions once assembled.

Power for the device is supplied through a USB-C connection to the Jetson Nano, which in turn provides power to the touchscreen, fan, and CSI camera (Figure 4.11). The CSI camera streams data into the Jetson Nano, where it is processed, and the results are displayed on the touchscreen via the HDMI interface. Inputs from the touchscreen are sent back to the Jeton Nano through a USB connection.

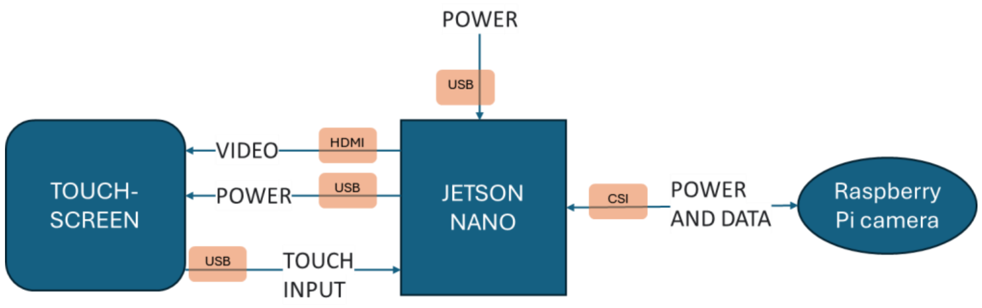


Figure 4.11 Overview of the electrical connections within the device.

5 THE SOFTWARE

5.1 Model design

5.1.1 Defining inputs and outputs

A black box model operates by processing inputs to generate outputs. It's crucial to clearly define these inputs and outputs. In the context of using a camera for machine monitoring, previously recorded frames can be utilized as inputs to predict the subsequent frame, making the previous frames the input and the current frame the output (Figure 5.1). By comparing pixel values between the predicted and the actual recorded frame, the accuracy of the model can be assessed. This comparison allows the training the model. The model can be used to compare the expected frame with the current frame to detect anomalies or deviations, indicating potential issues with the machine.

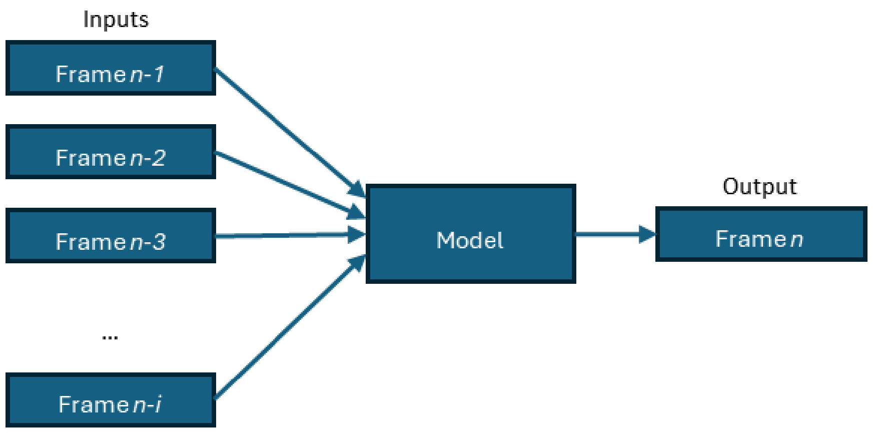


Figure 5.1 Model inputs and outputs, where *n* is the current frame number and *i* is the number of previous frames to be inputted into the model.

The approach of using past frames to predict future frames is theoretically sound but presents practical challenges in terms of computational efficiency and model complexity. Processing a continuous stream of images demands substantial computational resources, placing a significant burden on processing power and memory. This high requirement poses a particular challenge for a portable device, which will typically have limited hardware capabilities.

To address this issue, it's necessary to simplify the information processed by the model, thereby reducing the volume of data required. The critical aspect to focus on is the movement of the machine parts. These changes can be translated into part position information, significantly reducing the data before it's fed into the model. The model can then calculate the part positions from the next frame and compare them to the actual positions (Figure 5.2).

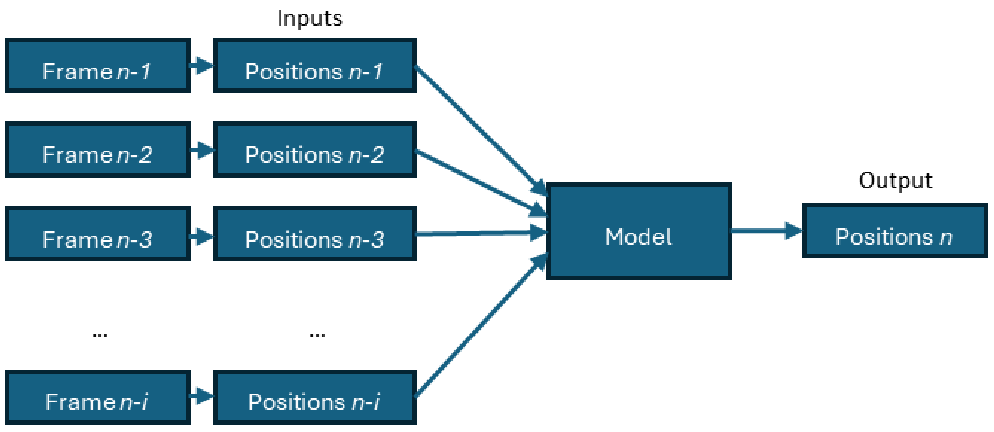


Figure 5.2 Model inputs and outputs, where n is the current frame number and i is the number of previous frames to be analysed. Each previous frame is converted to position data and then imputed into the model. The model output is the predicted position data frame n.

5.1.2 Workflow

A structured pipeline is necessary to analyse the observed machine on a standalone device. This involves several steps (Figure 5.3): reading the input images from a camera, preprocessing them for analysis, recognizing movement in the image, inputting the position information to a model of the machine and comparing the output of the model to the actual movement and finally, displaying the results.

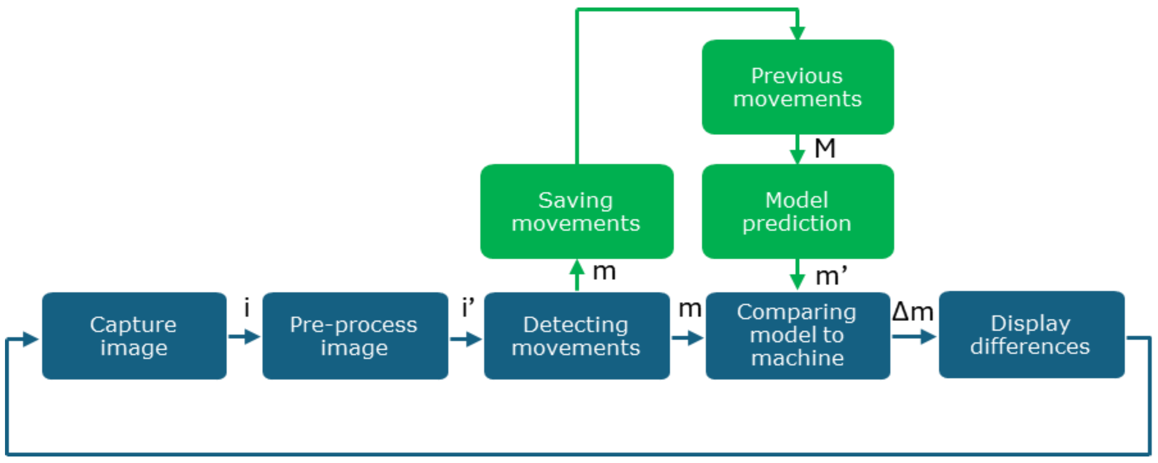


Figure 5.3 Machine monitoring pipeline, where i is the input image, i' is the pre-processed image, m is the detected movement information, m' is the predicted movement information for the current frame, M are the previous images movement information and Δm is the difference between the predicted and actual movement information.

To determine whether the machine is operating correctly, it's required to define what constitutes correct behaviour. Therefore, the identification phase is conducted before the monitoring process. This would be initiated by the user, after which the device is able to start monitoring (Figure 5.4).

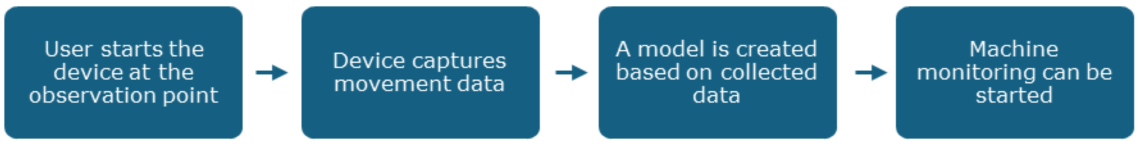

User starts the device at the observation point
Device captures movement data
A model is created based on collected data
Machine monitoring can be started

Figure 5.4 Model creation pipeline.

5.2 Evaluation

Validating the design involves testing its accuracy, efficiency, and effectiveness in real-world settings. The validation focuses on three key metrics: motion prediction accuracy, computation time, and anomaly score accuracy. Each of these metrics targets a specific aspect of the device's performance.

Motion tracking accuracy is visually assessed by the author. If the motion calculated by the system does not match the observed motion, it can be discarded.

Computation time measures the efficiency of the system, specifically how quickly it processes data and delivers outputs. This is measured by the playback frames per second (FPS)

Anomaly score is evaluated by calculating the modified MSE (Equation (5.1) of each bounding box, where the output of the created model is compared to the output from the motion tracking algorithm. Under normal behaviour, the model's outputs should closely align with the motion tracking outputs. A significant discrepancy between the two suggests an anomaly. If a significant difference occurs during normal behaviour, or if there is no difference when an anomaly is expected, then the device is not performing adequately.

$$\mathrm{MSE} = (y_x - q_x)^2 + \left(y_y - q_y\right)^2 + (y_w - q_w)^2 + (y_h - q_h)^2, \qquad (5.1)$$

where y_x – tracking bounding box X coordinate, normalized to image dimensions,

y_y – tracking bounding box Y coordinate, normalized to image dimensions,

y_w– tracking bounding box width, normalized to image dimensions,

y_h – tracking bounding box height, normalized to image dimensions.

q_x – model bounding box X coordinate, normalized to image dimensions
q_y – model bounding box Y coordinate, normalized to image dimensions
q_w– model bounding box width, normalized to image dimensions
q_h – model bounding box height, normalized to image dimensions

5.3 Software packages

OpenCV, or Open Computer Vision Library, is known for its versatile capabilities in computer vision. As an open-source library, it offers an infrastructure that supports real-time image processing and includes a broad range of functions from basic image manipulation to advanced algorithms for object and feature detection. This library is used for processing the visual data necessary, providing a vast range of pre-built functions that speed up the development and testing of complex image analysis techniques.

Complementing OpenCV, PyTorch is selected for its strengths in handling neural networks. It excels due to its flexible nature, making it ideal for academic research and projects that necessitate quick iterations and model adjustments. The library's GPU acceleration capabilities and the support of an active community offer tools and pre-trained models.

Alongside OpenCV and PyTorch, the device also incorporates NumPy and Tkinter to enhance both the backend computations and the user interface, respectively. NumPy is used for handling high-performance mathematical operations and manipulating large arrays of image data, which is essential for preprocessing steps before feeding data into the neural networks in PyTorch. On the frontend, Tkinter plays a role by providing a practical and easy-to-use graphical interface, allowing users to interact with the device. This includes starting and stopping processes, adjusting settings, and displaying real-time analysis results.

The Jetson Nano, which serves as the core computing unit of the system, supports, and comes with a ready-built version of Ubuntu 18.04. Because of these libraries used, the device is mainly programmed in the Python programming language.

5.4 Data collection

In the initial phase, video data is collected under controlled conditions where specific types of anomalies are artificially introduced into the machinery's operation. This approach allows the calibration of the system to recognize and differentiate various anomaly types based on visual cues. Following the controlled recordings, the system is further tested through a real-time camera feed directly connected to the algorithm's input. This step used for assessing the system's capability to process and analyse data in real time, simulating actual operational conditions. The real-time feed tests the system's response times and accuracy under typical industrial conditions, providing valuable feedback for further tuning and optimization.

The videos are recorded in MP4 format with a resolution of 1280 x 720 pixels. This resolution balances between detail and file size, showing clarity without excessively large data volumes that could hinder processing efficiency. The videos are captured at a frame rate of 30 frames per second, which is sufficient to capture motion and provide a realistic representation of how machines operate in real time.

5.5 Motion detection

Three distinct motion detection algorithms were tested for the device: MOG2, KNN, and Farneback optical flow. These algorithms were tested in two benchmark scenes to evaluate their effectiveness in detecting moving objects. Each algorithm was assessed based on its detection accuracy and performance in identifying and tracking moving objects in these scenes.

5.5.1 Preprocessing

The performance of an anomaly detection algorithm significantly relies on the input data. Generally, the more data available, the higher the potential accuracy of the model. However, processing larger datasets not only demands more computing time but also introduces potentially irrelevant data that may not contribute to the overall effectiveness of the model. Particularly with image data, the computational resources required can escalate rapidly.

To mitigate computational challenges, input data is simplified using preprocessing techniques such as erosion and dilation, which help reduce noise. After preprocessing, the largest 'n' objects that exceed a specified size threshold are identified and encapsulated within bounding boxes (Figure 5.5). This data, the size and position of these bounding boxes, serves as both the input to the model and the benchmark for expected output. Anomaly detection is performed by comparing the model's predictions against the actual measured data. Any discrepancies between these two datasets are analysed to identify and confirm the presence of anomalies.

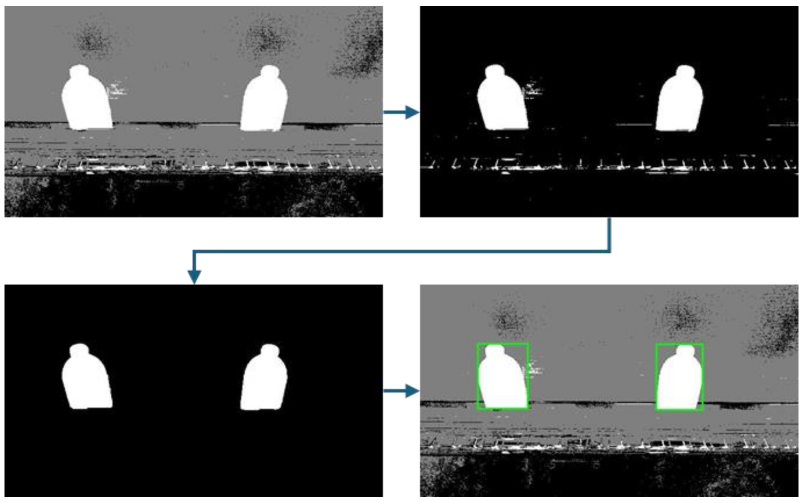


Figure 5.5 Example of finding the bounding boxes around moving objects in the bottle conveyor scene. Output from the motion detection algorithm shown in the top left; thresholding above shadow values in the top right; noise removal through erosion and dilation in the bottom left; and the largest white regions enclosed by bounding boxes in the bottom right.

5.6 Anomaly detection

5.6.1 Multilayer perceptron design

A typical Multi-Layer Perceptron (MLP) includes an input layer to receive data points, several hidden layers for processing, and an output layer with two nodes to classify data as either normal or anomalous. However, this approach is inadequate for the specific scenario, as MLP training requires a labelled dataset containing examples of both normal and anomalous conditions. Since there is only data representing normal operational conditions available, the network must be adapted to identify deviations from this 'normal' pattern as potential anomalies.

To address this challenge, the network is specifically configured to predict the next position of objects by analysing their historical positional data (Figure 5.6). Positional information, including the coordinates (x, y) and dimensions (width and height) for 'n' boxes during the last 'm' time steps, is fed into the MLP. The output is then set to predict the current position of these objects. This modelling approach allows the network to learn and recognize typical movement patterns during standard operations. Consequently, any deviations from these expected positions can be flagged as potential anomalies. This capability enables the system to identify irregular behaviours autonomously, without the need for a labelled dataset containing examples of anomalies.

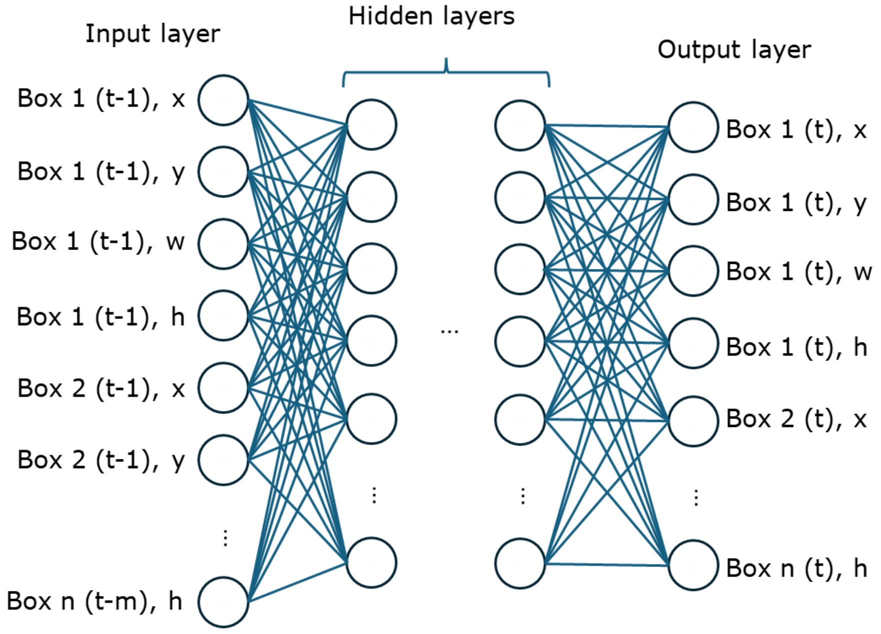


Figure 5.6 Configuration of the MLP, where *t* represents the current time step for positions, *n* is the maximum number of boxes analysed, *m* is the number of time steps analysed, *x* and *y* denote the horizontal and vertical positions of a box, respectively, and *w* and *h* represent the width and height of the box.

5.6.2 Autoencoder design

The structure of an autoencoder mirrors that of a MLP in that both are feed-forward networks. However, the autoencoder differentiates itself by typically featuring fewer neurons in its hidden layers than in the input or output layers. This reduction in neurons is aiming to condense the input data into a more compact representation that encapsulates the essential features of the system.

For this application, data concerning the current boxes is input into the autoencoder. The network is then tasked with reconstructing this data in its output (Figure 5.7). The premise is that if the autoencoder successfully learns the essential features of the system during training, it will be able to accurately reconstruct similar data. Consequently, any discrepancies between the model's output and the actual input data could indicate changes or anomalies within the system.

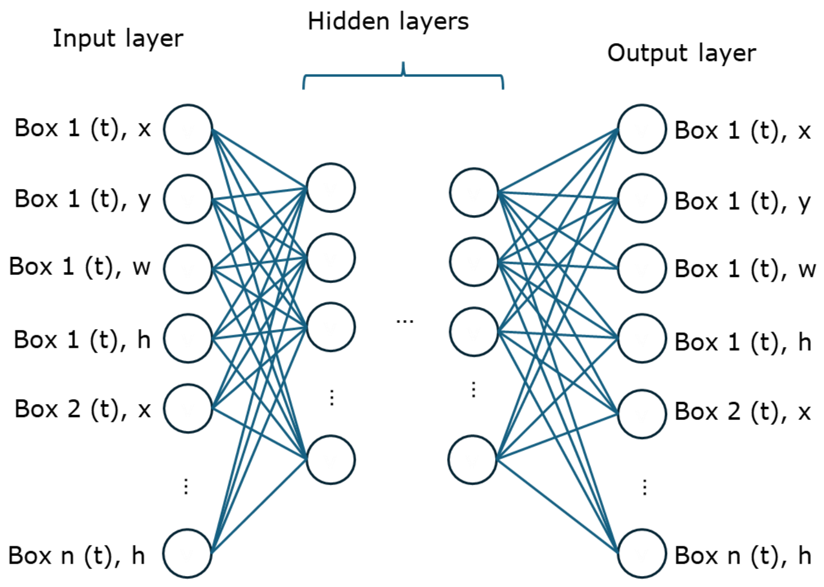


Figure 5.7 Configuration of the autoencoder, where *t* represents the current time step for positions, *n* is the maximum number of boxes analysed, *x* and *y* denote the horizontal and vertical positions of a box, respectively, and *w* and *h* represent the width and height of the box.

5.6.3 LSTM design

An LSTM (Long Short-Term Memory) network is an ideal choice for this task due to the time-dependent nature of the movements observed. Each movement influences subsequent movements, making the sequence of actions interdependent. LSTMs excel with time series data because they are specifically designed to recognize and remember patterns over extended time intervals. This capability allows them to be finely tuned for tasks where past events significantly affect future outcomes.

In this example, data from previous timesteps about the boxes is input into the LSTM. The LSTM is trained to predict the current timestep's box information based on this input coupled with the memory it has retained from past sequences. The greater the length of this memory, the more dynamics it can store, which enhances its predictive capability. However, this also increases the computational demands. The outputs of the LSTM nodes are then translated back into bounding box data through a fully connected layer at the output.

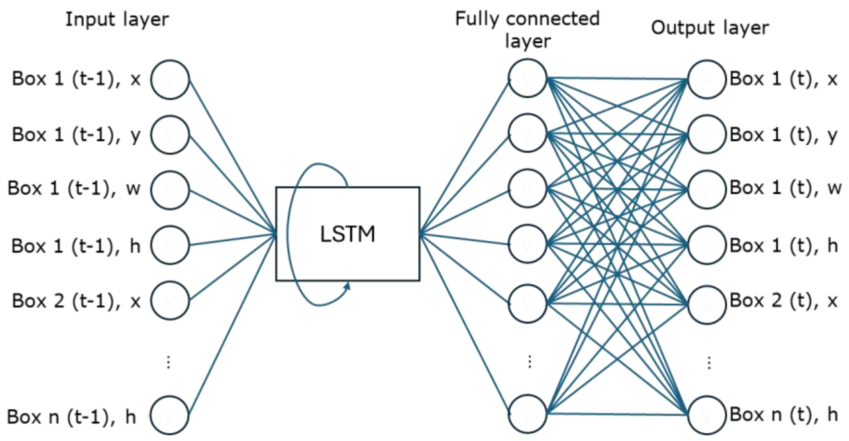


Figure 5.8 Configuration of the LSTM model, where *t* represents the current time step for positions, *n* is the maximum number of boxes analysed, *x* and *y* denote the horizontal and vertical positions of a box, respectively, and *w* and *h* represent the width and height of the box.

5.7 The interface

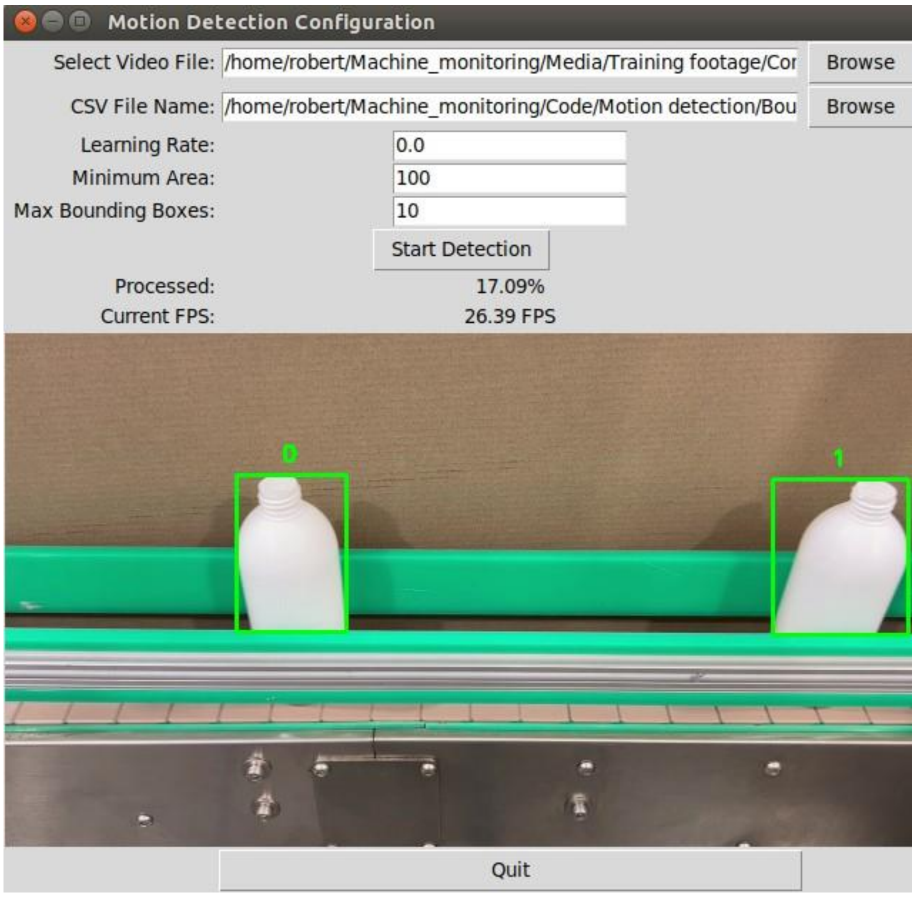
5.7.1 Data collection

During the data collection phase, the user plays a role in establishing a baseline of normal operation. They are required to upload or record a sample video that depicts the machinery operating correctly under conditions identical to those where ongoing monitoring will occur. It's vital that this recording is made in the same position and under the same lighting conditions as the observation location to ensure consistency in the data used for training.

Once the video is uploaded into the system, it undergoes preprocessing to enhance efficiency. The video is scaled down by a specified factor of 2 to reduce the time required for analysis. Following this, an OpenCV background subtractor is initialized with predefined settings that include the learning rate, minimum foreground object size, and the maximum number of objects allowed in the frame.

The background subtractor processes the video to isolate and identify each moving object's size and position. This information is recorded in a CSV file, along with the video's parameters. As the video progresses, the system draws bounding boxes around the detected objects and assigns each an ID marker. By highlighting the objects being tracked, users can visually verify that the system is accurately capturing and analysing the operational dynamics of the machinery.

Upon completion of the training phase, all gathered data—object sizes, positions, and corresponding video parameters—are saved to a .csv file. This compiled information forms the dataset on which the neural network will be trained. The training involves adapting the model to recognize and predict typical patterns of machine behaviour, setting a benchmark for detecting deviations that may indicate operational anomalies in the future.



Motion Detection Configuration

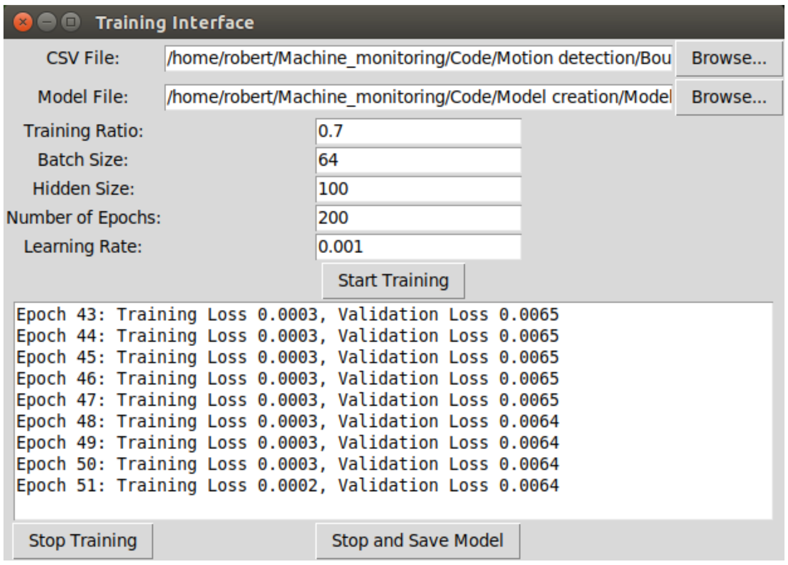
Select Video File:	/home/robert/Machine_monitoring/Media/Training footage/Cor	Browse
CSV File Name:	/home/robert/Machine_monitoring/Code/Motion detection/Bou	Browse
Learning Rate:	0.0	
Minimum Area:	100	
Max Bounding Boxes:	10	
Processed:	17.09%	
Current FPS:	26.39 FPS	

Figure 5.9 Screenshot of the UI During the Data Collection Process: The top part of the screen displays the input parameters, including the learning rate of the motion detection algorithm, the minimum bounding box area, and the maximum number of bounding boxes detected in a single frame. The bottom half of the screen presents the output from the motion detection algorithm, highlighting the moving object with a green bounding box and displaying its ID above it.

5.7.2 Training

In the training phase, the data gathered and recorded during the data gathering phase—specifically the information on each moving object's size, position, and associated video parameters—is loaded into memory. Once the data is loaded, the user is presented with the capability to customize several key parameters of the neural network through the user interface. These parameters include the training ratio (the split between training and validation data), batch size (the number of samples processed before the model is updated), hidden size (the number of units in the hidden layers of the neural network), number of epochs (the number of complete passes through the training dataset), and the learning rate (the step size at each iteration while moving toward a minimum of a loss function). Adjusting these settings allows the user to tailor the training process to fit specific needs and constraints, balancing between training speed and model accuracy.

With the parameters set, the user can initiate the training process via the user interface. As training progresses, real-time updates on the training and validation losses are displayed. This feedback provides insight into how well the model is learning from the data—lower losses indicate better learning. By monitoring these metrics, users can make decisions about possibly tweaking the neural network settings or halting the training early if the desired accuracy is achieved.



CSV File: /home/robert/Machine_monitoring/Code/Motion detection/Bou
Model File: /home/robert/Machine_monitoring/Code/Model creation/Model
Training Ratio: 0.7
Batch Size: 64
Hidden Size: 100
Number of Epochs: 200
Learning Rate: 0.001

```
Epoch 43: Training Loss 0.0003, Validation Loss 0.0065
Epoch 44: Training Loss 0.0003, Validation Loss 0.0065
Epoch 45: Training Loss 0.0003, Validation Loss 0.0065
Epoch 46: Training Loss 0.0003, Validation Loss 0.0065
Epoch 47: Training Loss 0.0003, Validation Loss 0.0065
Epoch 48: Training Loss 0.0003, Validation Loss 0.0064
Epoch 49: Training Loss 0.0003, Validation Loss 0.0064
Epoch 50: Training Loss 0.0003, Validation Loss 0.0064
Epoch 51: Training Loss 0.0002, Validation Loss 0.0064
```

Figure 5.10 Screenshot of the user interface during training.

5.7.3 Observation

In the observation phase begins with the user selecting the video input that will be monitored in real-time and loading the trained model, which has been optimized during the training phase. Just like in the data gathering phase, the system continues to analyse the video input to identify and track moving objects. Utilizing the same background subtraction techniques, the system calculates each object's position and size in real time. Simultaneously, the neural network model, now trained to understand the typical behaviour and positions of these objects under normal operating conditions, makes predictions about where these objects should ideally be located and how they should move based on the learned patterns.

The core of the anomaly detection lies in the computation of the anomaly score, which is determined by the mean squared error (MSE) between the model’s predictions and the actual observed values. This score quantifies the deviation of the observed behaviour from what is expected under normal conditions; a higher MSE indicates a greater deviation, suggesting an anomaly. If the anomaly score exceeds a predefined threshold, it signals that the machine is exhibiting unfamiliar or abnormal behaviour. This trigger can alert operators or trigger automated systems to take corrective actions, such as shutting down equipment for inspection or adjusting operational parameters to mitigate risk.

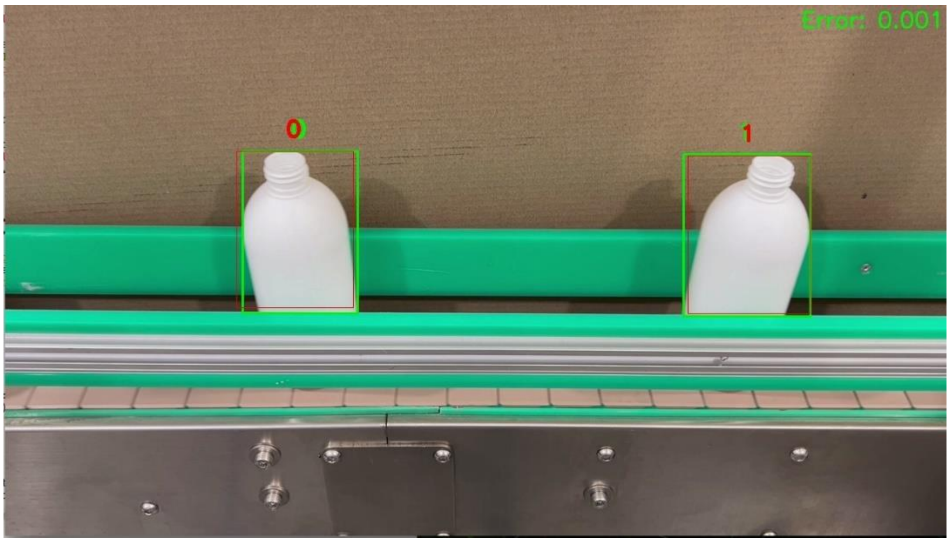


Figure 5.11 Screenshot of the observation window. The green bounding box is the observed object location, the red outline is the predicted location of the object. In the top right corner, the error between the prediction and the measured positions is displayed.

6 RESULTS AND DISCUSSIONS

6.1 Device

The mechanical components were assembled inside a 3D printed housing. These components include: A Jetson Nano for the processing unit, a Ingcool 7-inch LCD display for the user interface, A Noctua fan for cooling and a Raspberry Pi camera 2 for the camera. The housing included a standard mounting point, so that the camera could be attached to a tripod (Figure 6.1).

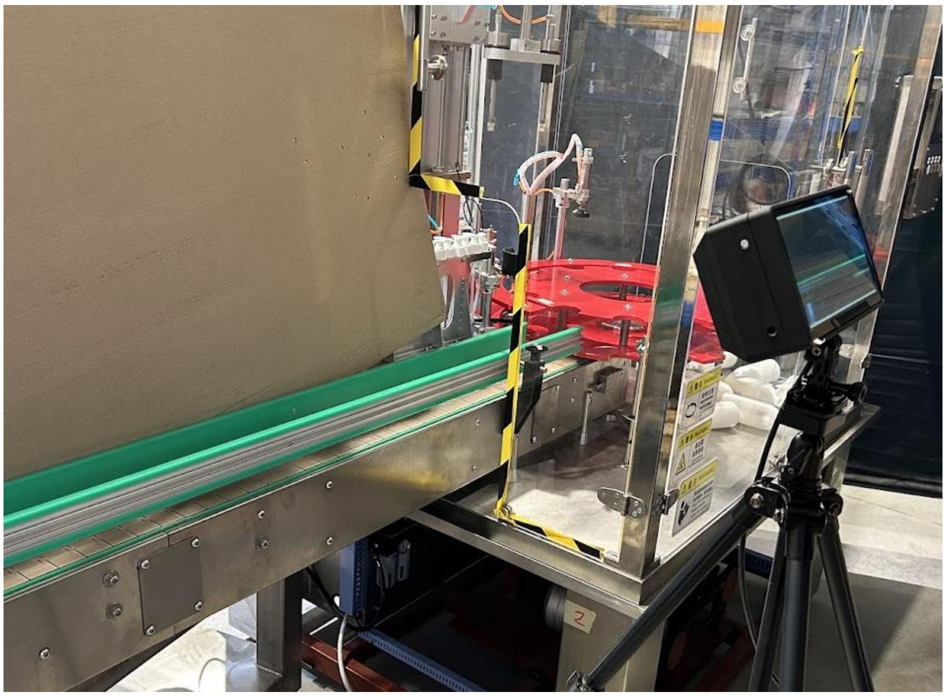


Figure 6.1 Device mounted on a tripod, observing the machine.

6.2 Performance

6.2.1 Motion detection

Three different methods for motion detection were tested: KNN background subtraction, MOG2 background subtraction, and optical flow using the Farneback algorithm. These methods were evaluated in two different scenes, with videos shot at a resolution of 1280x720 and a frame rate of 30 fps.

The KNN background subtraction method demonstrated the poorest performance in scenarios with a static background. Despite the stable conditions, it struggled to detect moving objects as complete entities; only the leading and trailing edges of the objects were consistently identified (Figure 6.2). In scenarios where the background was subject to changes, the KNN background subtractor exhibited similar performance issues. The algorithm continued to have difficulty recognizing the full scope of movement in the indexing wheel footage, typically detecting only the advancing or trailing edge of moving objects. When the video was scaled down by 50%, the method achieved an average frame rate of 35 FPS in both scenes.

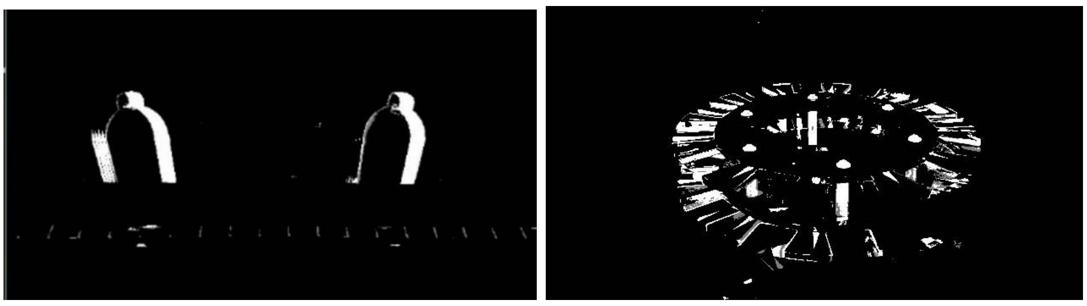


Figure 6.2. Examples of the KNN background separation algorithm. Moving areas are shown in white. Left side shows frame from conveyor belt scene, right side shows frame from index wheel scene.

The MOG2 algorithm excelled in the conveyor belt scene. It distinguished the background and successfully separated the bottles from the conveyor belt. The algorithm was good at detecting shadows, differentiating them from moving objects (Figure 6.3). In the case of the indexing wheel, the learning rate needed to be increased due to minor changes in background positions. Nevertheless, the model worked smoothly, finding the moving parts of the machine very effectively. When the video was scaled down by 50%, the algorithm maintained an average frame rate of 34 FPS in both scenes.

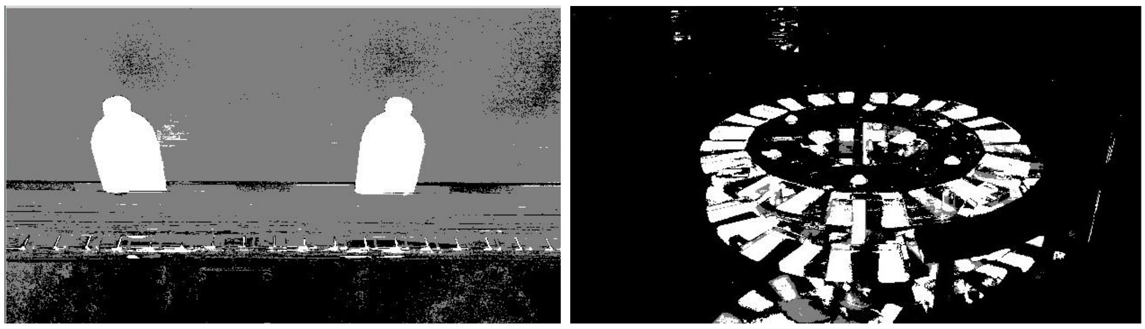


Figure 6.3 Examples of the MOG2 background separation algorithm. Moving areas are shown in white. Left side shows frame from conveyor belt scene, right side shows frame from index wheel scene.

Calculating optical flow using the Farneback method [31] proved to be the slowest but also the most promising. When scaled down by 50%, the framerate was close to 1 frame per second. Further reducing the resolution helped slightly; scaling the video down to 10% of its original size resulted in a resolution of 128x76, achieving 20 frames per second. However, at that resolution, there was little to no useful information. Despite these limitations, the method shows promise with more computing power, as it illustrates movement direction within the frame. With this added benefit, the resulting model could have the potential to become more accurate.

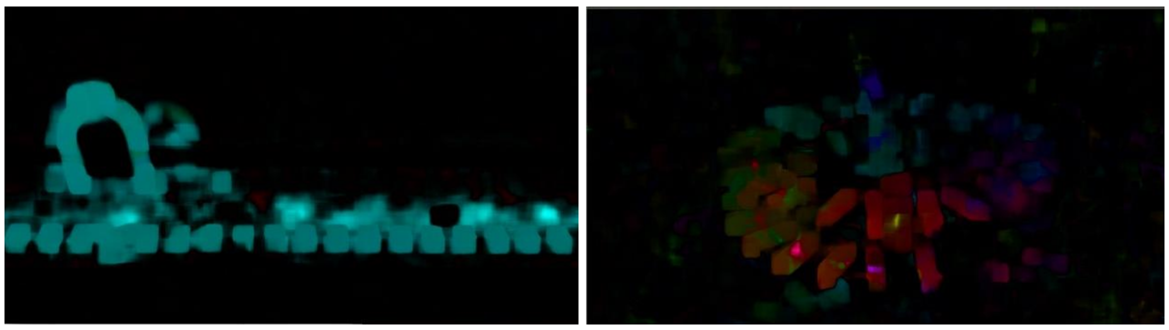


Figure 6.4 Examples of the optical flow analysis at a resolution of 640x320. Movement direction is shown the colour hue, and the magnitude is shown by the colour intensity. Left side shows frame from conveyor belt scene, right side shows frame from index wheel scene.

Among the three motion detection algorithms evaluated—KNN background subtraction, MOG2 background subtraction, and optical flow using the Farneback method—the MOG2 algorithm emerged as the most effective across both test scenes. Although the KNN algorithm slightly outperformed others in terms of FPS, MOG2 demonstrated greater adaptability in varied scenarios, especially in environments with controlled backgrounds. Moreover, MOG2 was particularly adept at distinguishing between actual movements and shadows, even in dynamic backgrounds. While optical flow provided detailed motion with directional data, its low frame rate limits its suitability for real-time applications. Nevertheless, with advancements in computing power, optical flow has potential for future applications.

The bounding box finding algorithm performance deteriorated in scenarios where the background is complex, and the moving objects travel in different directions with high overlap. Under these conditions, the bounding boxes fail to provide reliable data for model building. Specifically, in the case of the indexing wheel, the bounding boxes created through this process frequently shift position erratically, failing to offer a clear depiction of the machine's mechanics (Figure 6.5). Due to these issues, the indexing wheel scene was not utilized for anomaly detection testing. In the future, a top-down view of the rotating wheel should be tested.

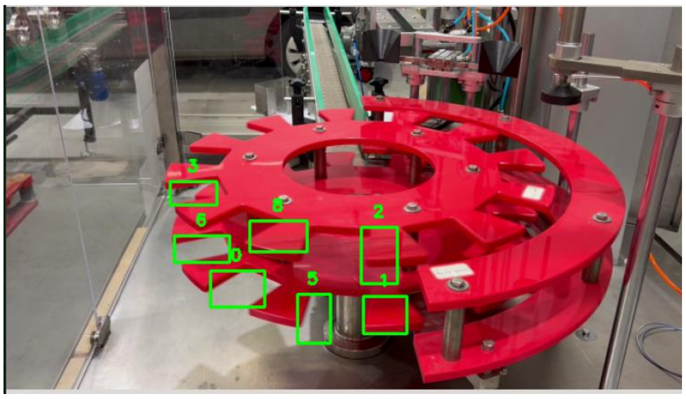


Figure 6.5 Example of bounding boxes inside indexing wheel scene. Even though the whole wheel is spinning, only the lower left corner shows movement with bounding boxes.

6.2.2 Anomaly detection

For the anomaly detection algorithm, three types of neural networks were evaluated: the multilayer perceptron (MLP), autoencoder, and long short-term memory (LSTM). Each network was trained on a video illustrating normal working conditions, allowing them to establish a baseline for expected behaviour. After training, they were tested on a separate validation video file containing both normal behaviour and anomalous behaviour, such as bottles falling. The neural networks were trained to minimize the error between the predicted bounding box location and size and the actual bounding box position and size.

The MLP was tested under different configurations, with variations in the number of previous time steps as inputs (ranging from 1 to 30), the number of hidden layers (from 1 to 5), and the number of nodes per hidden layer (from 100 to 2000) with a maximum number of boxes set to 6. Despite these variations, the model struggled to accurately represent real-world scenarios, even though it improved the loss function within the learning process (Figure 6.6). The predicted bounding boxes failed to align with the actual positions in the real world (Figure 6.7). This misalignment was more pronounced in configurations with more hidden layers, which also resulted in longer and less effective training sessions. As a result, the model was deemed unsuitable for reliable anomaly detection.

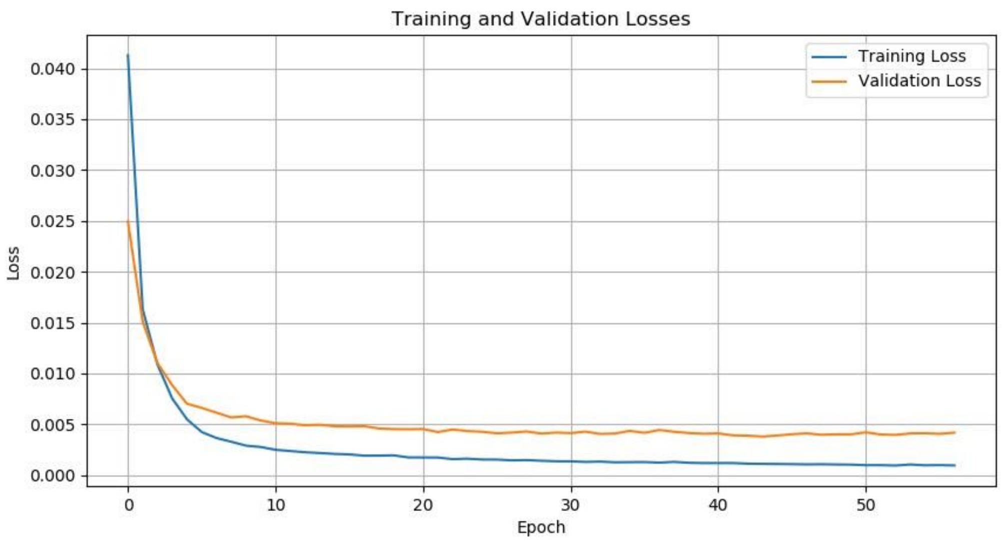


Figure 6.6 Training progress of the MLP on training data.

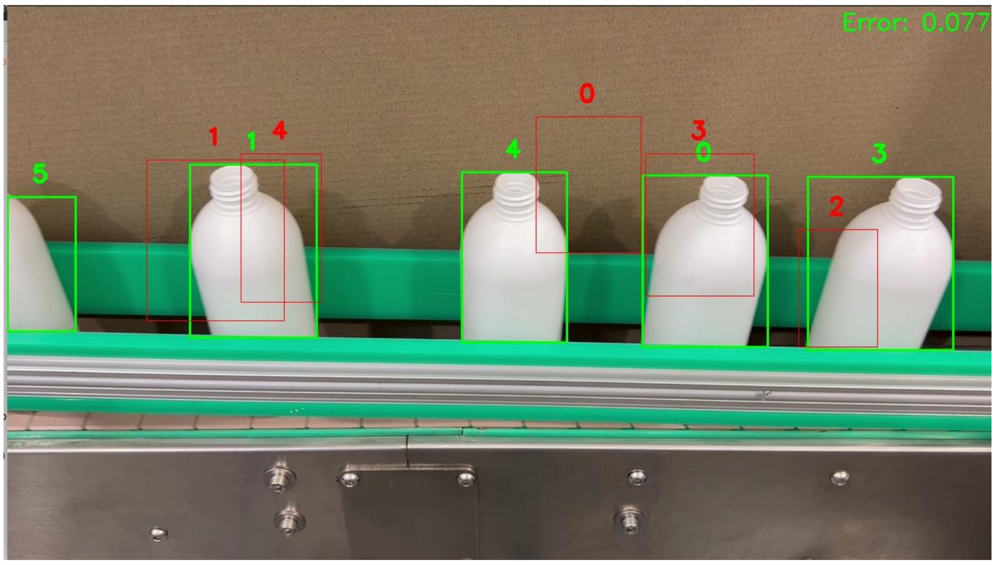


Figure 6.7 MLP model prediction results, configured with 30 time steps and three hidden layers (2000, 1000, 500 neurons each). The green outline indicates the actual tracked positions of the objects, while the red outline shows the predicted positions by the MLP model.

The autoencoder network underwent testing with variations that included the number of hidden layers (from 1 to 4) and the number of nodes per hidden layer (ranging from 25 to 10), with the number of boxes analysed capped at 6. The most effective version of the autoencoder featured four layers, with the number of neurons in each layer matching those in the input and output layers (Figure 6.8). Any reduction in the number of neurons led to a decrease in the model's accuracy. This setup enabled the autoencoder to model the positions of the boxes more accurately than the MLP. The predicted bounding boxes aligned more closely with the actual measured values (Figure 6.9).

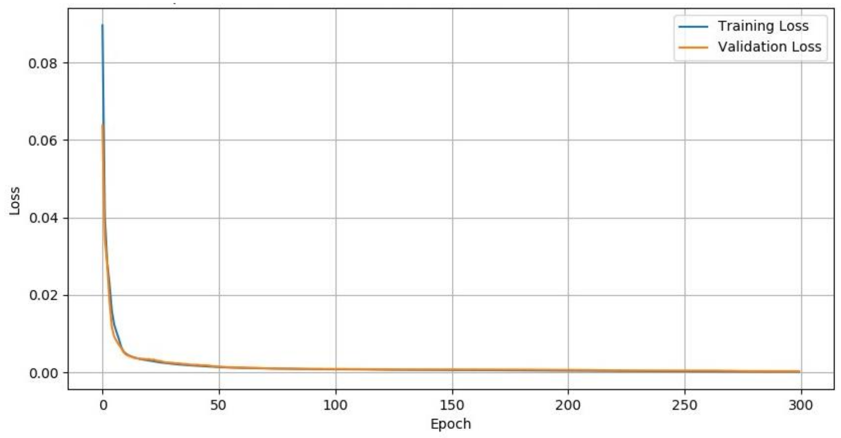


Figure 6.8 Training progress of an autoencoder for 6 boxes, with 4 hidden layers with 24 neurons.

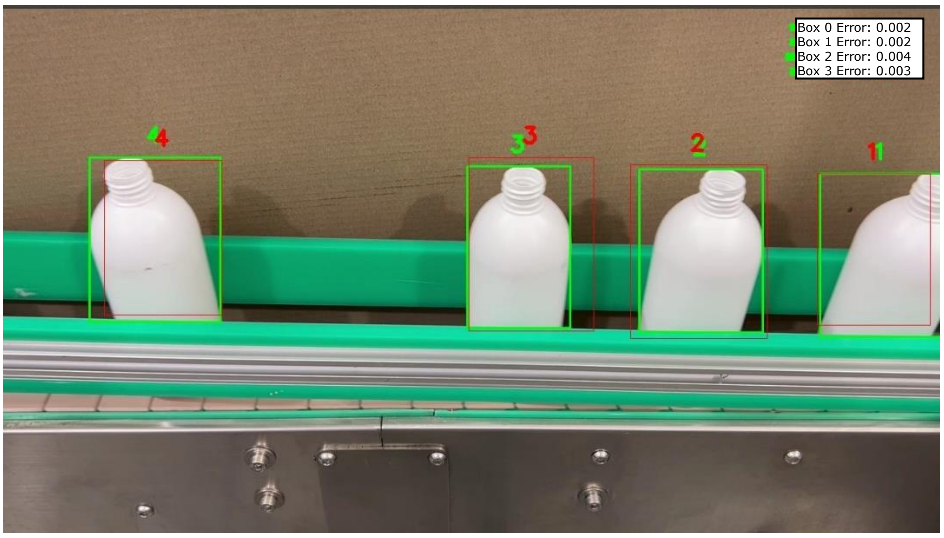


Figure 6.9 Autoencoder output. Top right corner shows the errors with a proportional rectangle to illustrate the size difference. The green outline indicates the actual tracked positions of the objects, while the red outline shows the predicted positions by the MLP model.

The autoencoder also performs very well in case of an anomaly, where a tipped bottle showed very high error, compared to other bottles (Figure 6.10). However, the model is prone to inaccuracy. Situations such as bottles exiting the scene can trigger these inaccuracies, compromising the reliability of the model (Figure 6.11).

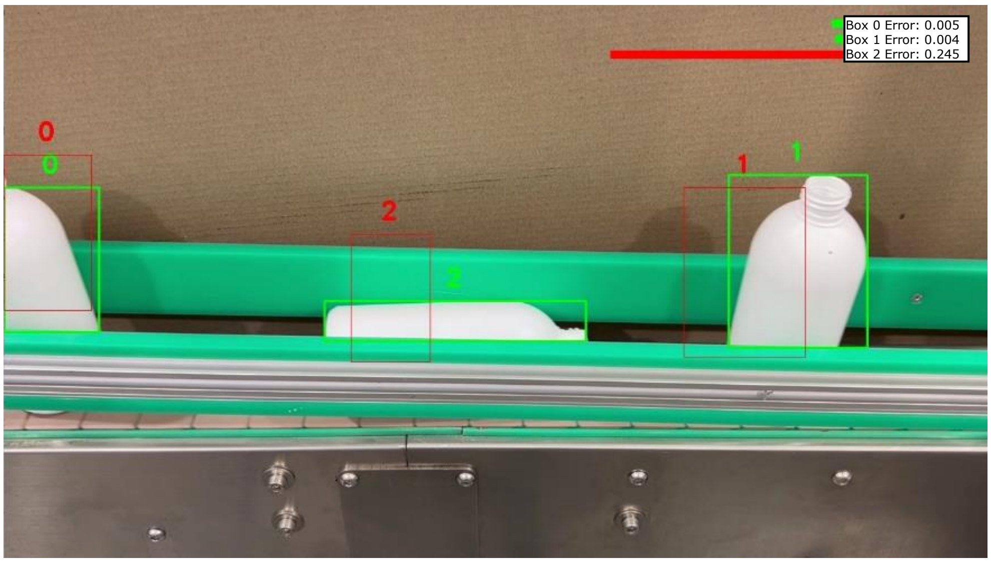


Figure 6.10 Output of the autoencoder when presented with a tipped bottle. Top right corner shows the errors with a proportional rectangle to illustrate the size difference. For the tipped bottle (2) the error is very high, indicating an anomaly.

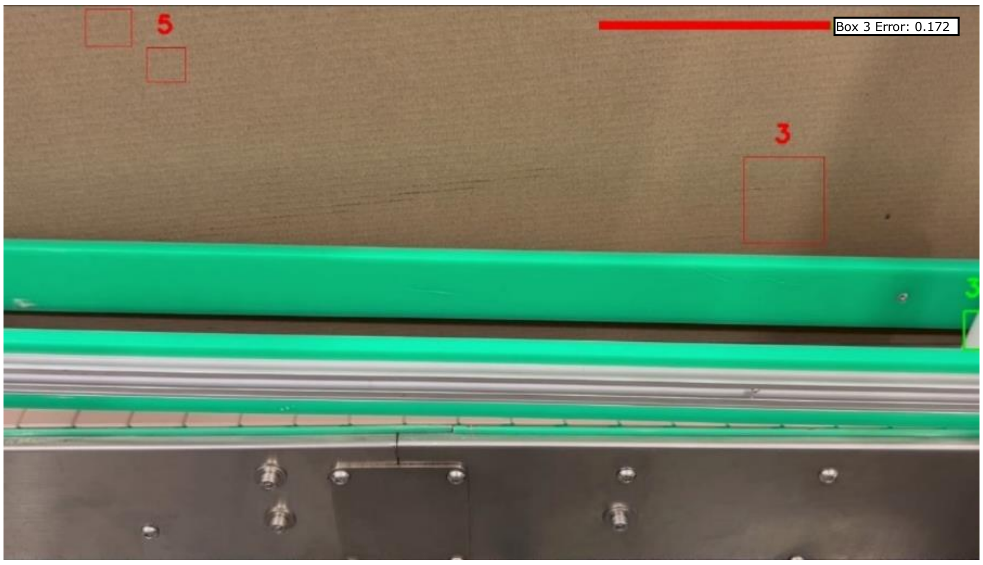


Figure 6.11. Output of the autoencoder during a transition phase where a bottle exits on the right side. The model struggles with this scenario, incorrectly predicting two movements in the top left corner and inaccurately calculating the position of bottle 3, which is flagged as an anomaly.

The LSTM model was tested with variations in the number of memory nodes (ranging from 25 to 500) and the number of internal LSTM layers (1 to 2), while the maximum number of boxes was set to 6. The most effective setup consisted of a single LSTM layer with 30 nodes, which proved capable of tracking the objects (Figure 6.12). These adjustments allowed the model to predict the size and location of the boxes (Figure 6.13). However, like the autoencoder, the model faced challenges with edge cases, such as when boxes would suddenly appear or disappear (Figure 6.14). Despite these challenges, the model could identify anomalies, such as a bottle that had tipped over (Figure 6.15).

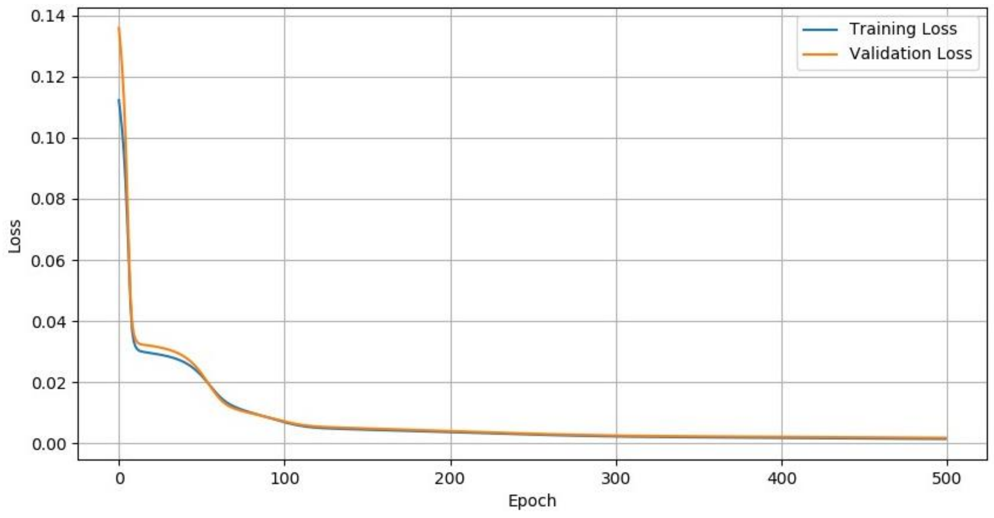


Figure 6.12 Training progress of the LSTM model with 30 nodes.

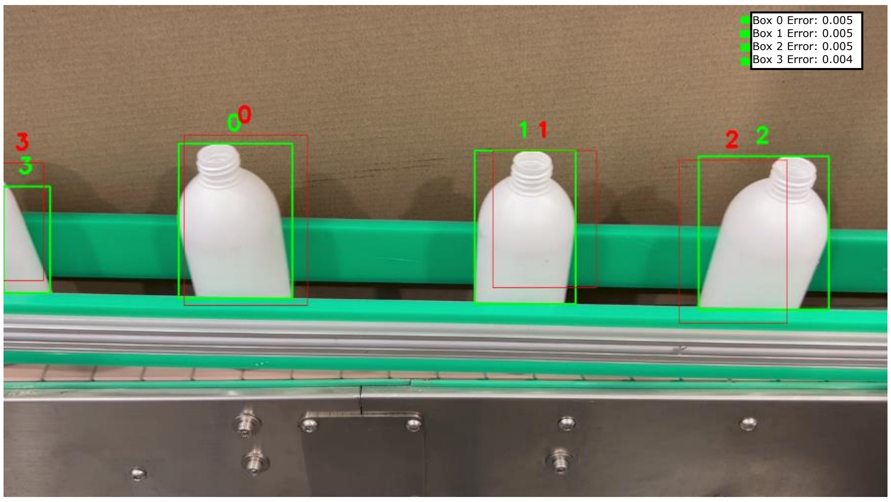


Figure 6.13 LSTM model output during normal conditions. The green outline indicates the actual tracked positions of the objects, while the red outline shows the predicted positions by the MLP model.

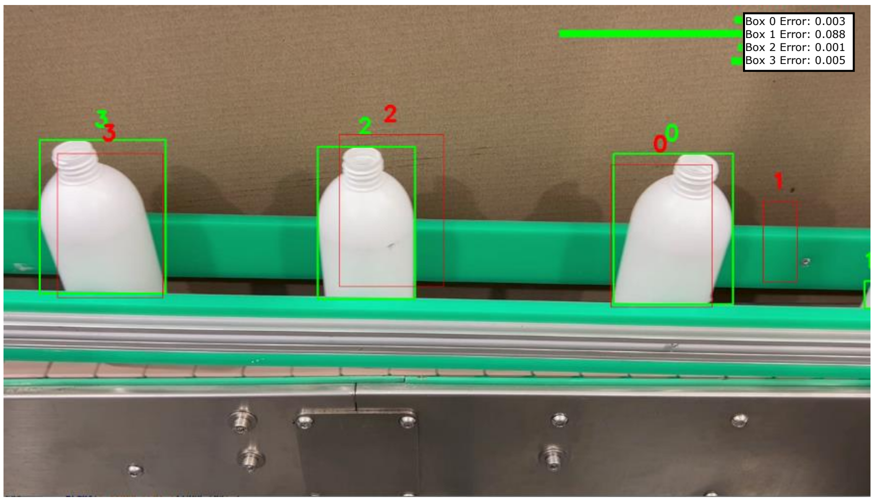


Figure 6.14 LSTM model struggling to accurately track a bottle as it exists on the right side of the screen.

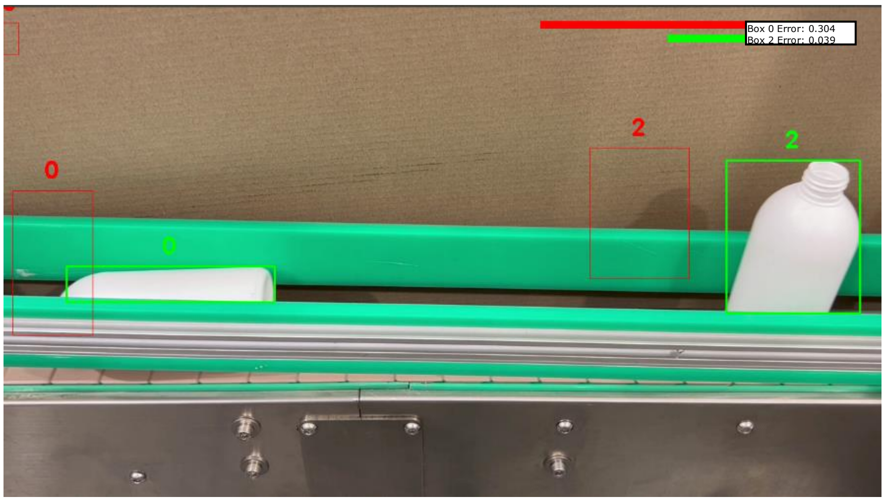


Figure 6.15 LSTM model detecting an anomaly of a tipped bottle.

The MLP, while traditional, proved not effective. The Autoencoder improved upon the MLP by learning to compress and reconstruct input data, which allowed it to better detect discrepancies indicative of anomalies. However, it too struggled with edge cases where objects transitioned into or out of the scene. The LSTM model while also preforming better than the MLP, still faced challenges with sudden appearances and disappearances of objects.

Ultimately, while each model has its strengths, the autoencoder stands out for its robustness and adaptability in complex dynamic environments. Future work should focus on enhancing these models' resistance to false positives and improving their ability to generalize from limited examples. Advancements in computational power and further tuning of model parameters could also enhance their efficacy and reliability in real-time anomaly detection applications.

6.3 Future plans

6.3.1 Hardware

The hardware configuration of the device offers several avenues for enhancement to accommodate more complex models and improve processing speeds. The impending discontinuation of the Jetson Nano board in 2026 necessitates the migration to a newer model, ensuring future compatibility and performance improvements. An upgrade to the processing unit's (PU) RAM is advisable, addressing a limitation encountered during the development phase. Additionally, incorporating an SSD would increase file transfer rates and support the storage of larger files.

Further improvements could include replacing the current Raspberry Pi camera with a model featuring adjustable focus to enhance image clarity and detail. An adaptor could be designed to allow the device to accommodate standard camera lenses, broadening its applicability and flexibility in various industrial environments. To protect the device in more demanding conditions, the housing could be engineered to be watertight, safeguarding the electronic components from moisture and dust.

To optimize space and possibly enhance the device's overall design, the processing unit could be replaced with a custom-designed PCB. This would not only save space but also potentially streamline the assembly and maintenance of the device, making it more robust and user-friendly in industrial settings.

6.3.2 Software

Improvements to the software aspect of the anomaly detection device could significantly enhance its efficiency and accuracy. Migrating the existing software from its current programming language to C++ could substantially boost execution speed. C++ is known for its performance advantages, especially in systems requiring real-time processing.

In terms of algorithmic enhancement, exploring and integrating newer optical flow algorithms would be beneficial. By implementing advanced optical flow techniques, the system can achieve more accurate and reliable motion detection. Furthermore, developing a neural network designed to process raw image data directly could streamline the entire analysis pipeline. This neural network would eliminate the need for a separate motion detection step, processing the input images in one continuous operation. By doing this, the system can reduce computational overhead and simplify the processing chain, potentially leading to faster and more efficient anomaly detection. This approach would leverage the strengths of deep learning to handle complex patterns and anomalies that traditional image processing methods may not detect effectively.

SUMMARY

This thesis explored the development of a motion-based anomaly detection device designed to enhance the monitoring of industrial machinery using artificial intelligence (AI) and machine vision technologies. During the literature review, different existing solution were analysed, and it was determined that this device could prove useful. The thesis proposed an approach to machinery condition monitoring that leverages the capabilities of cameras combined with AI to provide real-time, efficient, and scalable observations.

The core of the proposed device consists of three main components: a camera, a processing unit, and a touchscreen interface. The camera captures live footage of machinery in operation, which is then processed by the unit to detect and analyse anomalies in the machine's motion. This processing utilizes image processing algorithms from the OpenCV library and modelling techniques via neural networks implemented with PyTorch. The primary aim is to identify deviations in machine operation, such as unexpected part movements, which can indicate equipment failure.

A prototype has been successfully developed and tested. This thesis evaluated different motion detection algorithms—KNN, MOG2, and optical flow—to establish a motion data for anomaly detection. Additionally, 3 machine learning models were tested on this motion data, including the Multi-Layer Perceptron (MLP), Autoencoder, and Long Short-Term Memory (LSTM) networks, to assess their anomaly detection capabilities in industrial settings.

Among the motion detection methods, MOG2 stood out for its adaptability and accuracy in distinguishing between moving objects and static backgrounds, especially under controlled conditions. Optical flow, despite its lower frame rate, showed potential for detailed movement analysis and could be more applicable with future technological advancements. An autoencoder proved more effective by learning to reconstruct data and identify deviations without explicit anomaly labels. However, it faced challenges with transitional object movements.

Future efforts should concentrate on enhancing the robustness of these models and minimizing false positives. One promising strategy could involve developing hybrid models that merge the strengths of neural network-based motion detection algorithms and anomaly detection neural networks. Additionally, investing in computational resources and refining model parameters are steps to improve the performance.

KOKKUVÕTE

See lõputöö uuris liikumispõhise anomaaliate tuvastamise seadme väljatöötamist, mis on mõeldud tehisintellekti ja masinnägemise tehnoloogiate abil tööstusmasinate jälgimise tõhustamiseks. Kirjanduse ülevaate käigus analüüsiti erinevaid saadaval olevaid tooteid ning lahendusi ja leiti, et selline seade võib osutuda kasulikuks. Lõputöö pakkus välja lähenemisviisi masinate seisukorra jälgimiseks, mis kasutab kaamera ja tehisintellektiga kombineeritud seadet, et pakkuda reaalajas masina seiret.

Kavandatava seade koosneb kolmest põhikomponendist: kaamerast, protsessorist ja puutetundliku ekraanist. Kaamera jäädvustab reaalajas kaadreid töötavatest masina osadest, mida seade seejärel töötleb, et tuvastada ja analüüsida masina liikumise kõrvalekaldeid. See töötlemine kasutab pilditöötlusalgoritme OpenCV teegist ja modelleerimistehnikaid PyTorchiga rakendatud närvivõrkude kaudu. Eesmärk on tuvastada kõrvalekalded masina töös, näiteks masina osade ootamatud liikumised, mis võivad viidata masina rikkele.

Prototüüp on edukalt välja töötatud ja testitud. Selles lõputöös hinnati erinevaid liikumistuvastuse algoritme, KNN, MOG2 ja optilist voogu, et luua anomaalia tuvastamiseks liikumisandmed. Lisaks testiti nende liikumisandmetega 3 masinõppemudelit, sealhulgas mitmekihilise pertseptroni, automaatkodeerija ja pika-lühiajalise mälu võrke, et hinnata nende anomaaliate tuvastamise võimalusi tööstuslikes olukordades.

Liikumistuvastusmeetoditest paistis MOG2 silma kohanemisvõime ja täpsusega liikuvate objektide ja staatilise tausta eristamisel, eriti kontrollitud tingimustes. Vaatamata madalamale kaadrisagedusele näitas optiline voog üksikasjaliku liikumise analüüsi potentsiaali ja võib olla tulevaste tehnoloogiliste edusammudega paremini rakendatav. Automaatkodeerija osutus tõhusamaimaks masinõppemudeliks, õppides rekonstrueerima andmeid ja tuvastama kõrvalekaldeid ilma anomaalia siltideta. Siiski seisis see silmitsi väljakutsetega üleminekuliste objektide liikumisega.

Tulevased jõupingutused peaksid keskenduma nende mudelite optimeerimisega ja valepositiivide minimeerimisele. Üks strateegia võiks hõlmata hübriidmudelite väljatöötamist, mis ühendavad liikumistuvastuse ja anomaaliatuvastuse masinõppemudelid. Lisaks parandaks tulemuse arvutusressurssidesse investeerimine ja mudeli parameetrite optimeerimine.

LIST OF REFERENCES

[1] Statistics Estonia, 'RAA0042: VALUE ADDED (ESA 2010) by Economic activity (EMTAK 2008) /component, Year, Quarter and Indicator'. Feb. 11, 2024. [Online]. Available: https://andmed.stat.ee/et/stat/majandus__rahvamajanduse-arvepidamine__sisemajanduse-koguprodukt-(skp)__sisemajanduse-koguprodukt-tootmise-meetodil/RAA0042/table/tableViewLayout2

[2] The European Parliament and The Council of The European Union, 'Directive 2006/42/EC on machinery, and amending Directive 95/16/EC (recast)', Jun. 2006, [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32006L0042

[3] Siemens AG, 'The True Cost of Downtime 2022'. 2023. Accessed: Feb. 11, 2024. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:3d606495-dbe0-43e4-80b1-d04e27ada920/dics-b10153-00-7600truecostofdowntime2022-144.pdf

[4] 'TX2 Carrier Board for Embedded Vision | Teledyne FLIR'. Accessed: Apr. 02, 2024. [Online]. Available: https://www.flir.eu/products/quartet-embedded-solution-for-tx2?vertical=machine+vision&segment=iis

[5] 'FourJaw's Hardware | Machine Monitoring | FourJaw'. Accessed: Apr. 03, 2024. [Online]. Available: https://fourjaw.com/intro-to-fourjaw/hardware

[6] 'Festo AX – Make decisions based on facts | Festo EE'. Accessed: Apr. 03, 2024. [Online]. Available: https://www.festo.com/ee/en/e/solutions/digitalisation-of-production/software-for-industrial-production/festo-automation-experience-id_408724/

[7] 'AI in Manufacturing - AI Optimizes Manufacturing & Plant Operations'. Accessed: Apr. 02, 2024. [Online]. Available: https://www.clarifai.com/solutions/ai-in-manufacturing

[8] E. R. Davies, *Computer and Machine Vision: Theory, Algorithms, Practicalities*, 4th ed. San Diego: Elsevier Science, 2012. doi: 10.1016/C2010-0-66926-4.

[9] Z. Zivkovic, *Improved Adaptive Gaussian Mixture Model for Background Subtraction*, vol. 2. 2004, p. 31 Vol.2. doi: 10.1109/ICPR.2004.1333992.

[10] Z. Zivkovic and F. van der Heijden, 'Efficient adaptive density estimation per image pixel for the task of background subtraction', *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, May 2006, doi: 10.1016/j.patrec.2005.11.005.

[11] 'OpenCV: How to Use Background Subtraction Methods'. Accessed: Apr. 06, 2024. [Online]. Available: https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html

[12] '(16) (PDF) Multi-objective optimization for parameter selection and characterization of optical flow methods'. Accessed: Apr. 23, 2024. [Online]. Available: https://www.researchgate.net/publication/295100063_Multi-objective_optimization_for_parameter_selection_and_characterization_of_optical_flow_methods

[13] 'Morphology - Erosion'. Accessed: Apr. 06, 2024. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm

[14] 'Morphology - Dilation'. Accessed: Apr. 06, 2024. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm

[15] S. Suzuki and K. be, 'Topological structural analysis of digitized binary images by border following', *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, Apr. 1985, doi: 10.1016/0734-189X(85)90016-7.

[16] A. Duun-Henriksen *et al.*, 'Model Identification Using Stochastic Differential Equation Grey-Box Models in Diabetes', *Journal of diabetes science and technology*, vol. 7, pp. 431–440, Mar. 2013, doi: 10.1177/193229681300700220.

[17] S. K. Adari and S. Alla, *Beginning Anomaly Detection Using Python-Based Deep Learning: Implement Anomaly Detection Applications with Keras and Pytorch*, 2nd ed. Berkeley, CA: Apress L. P, 2024. doi: 10.1007/979-8-8688-0008-5.

[18] R. P. Ltd, 'Buy a Raspberry Pi 5', Raspberry Pi. Accessed: Apr. 15, 2024. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/

[19] 'RPI5-8GB-SINGLE - SBC, Raspberry Pi5 8GB, BCM2712, Arm Cortex-A76, 8GB RAM, MicroSD, Wifi, HDMI, Power button'. Accessed: Apr. 15, 2024. [Online]. Available: https://ee.farnell.com/raspberry-pi/rpi5-8gb-single/raspberry-pi-5-model-b-8gb-2-4ghz/dp/4256000

[20] 'reComputer J1010 -Edge AI Device with Jetson Nano module, M.2 Key E Slot, Type-C connectors, Aluminium case, pre-installed JetPack System'. Accessed: Apr. 15, 2024. [Online]. Available: https://www.seeedstudio.com/Jetson-10-1-A0-p-5336.html

[21] 'Jetson Nano Developer Kit', NVIDIA Developer. Accessed: Apr. 15, 2024. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[22] 'Coral Dev Board 1GB - minicomputer with NXP i.MX 8M', Kamami.pl. Accessed: Apr. 15, 2024. [Online]. Available: https://kamami.pl/en/coral/575394-coral-dev-board-minicomputer-with-nxp-i-mx-8m.html

[23] '110061362 - Developer Kit, NVIDIA Jetson Nano, ARM Cortex-A57 CPU, Maxwell, 4GB RAM, 16 GB eMMC, HDMI'. Accessed: Apr. 15, 2024. [Online]. Available: https://ee.farnell.com/seeed-studio/110061362/nvidia-jetson-nano-ref-carrier/dp/4126473

[24] ‘Google Coral Dev Board (1GB/4GB)’, buyzero.de. Accessed: Apr. 15, 2024. [Online]. Available: https://buyzero.de/products/google-coral-dev-board-4gb

[25] ‘Amazon.com: Camera USB 5-50mm Varifocal Lens Webcam High Speed VGA 100fps Full HD 1080P USB Camera with Aluminum Mini Case Webcam Support OpenCV for Android Windows Linux PC Mac : Electronics’. Accessed: Apr. 20, 2024. [Online]. Available: https://www.amazon.com/Camera-USB-Varifocal-Aluminum-Android/dp/B07R4ZKYFQ

[26] F. Systems, ‘Gigabit Ethernet-based machine vision camera’, The Engineer. Accessed: Apr. 20, 2024. [Online]. Available: https://www.theengineer.co.uk/content/product/gigabit-ethernet-based-machine-vision-camera/

[27] ‘CSI Camera Set 5MP AR0521 Color’, Toradex. Accessed: Apr. 20, 2024. [Online]. Available: https://www.toradex.com/accessories/csi-camera-set-5mp-ar0521-color

[28] R. P. Ltd, ‘Buy a Raspberry Pi Camera Module 2’, Raspberry Pi. Accessed: Apr. 20, 2024. [Online]. Available: https://www.raspberrypi.com/products/camera-module-v2/

[29] ‘Amazon.com: Ingcool 7 inch HDMI LCD 1024x600 Resolution Capacitive Touch Screen IPS Display Module Compatible with Raspberry Pi 4 3 2 1 B B+ A+,Jetson Nano,Support Software Resolution (up to 1920×1080) : Electronics’. Accessed: Apr. 20, 2024. [Online]. Available: https://www.amazon.com/Ingcool-Resolution-Capacitive-Compatible-Raspberry/dp/B08H8HZRLQ/ref=sr_1_3?crid=3C3QHM2ZIG382&dib=eyJ2IjoiMSJ9.446Q9IeEUg0BgkiZdUlgElL4ffF2HlI4VGKLEUy8hXFjBprHhXWifC8W7wr19-gUL-_O3MBroSegUCeNUa49aVzDOcWEunmOy7rajPPfPPs0C1MJj5j_j6DE68DQtijc3lZfp887epqcHBWk_uLYzuCD6Gk4lszMcmsXZTrlHG3gf0M6daPienHKtxivpFJo_gBV36suBEBEvXfTtnKTpetOUIK4ZxCiWpq_UqrhJM8.626kfgEwaXuTCqT8WYKfV7GWAMpNAKpC6dr5d3CdU5Q&dib_tag=se&keywords=jetson%2Bnano%2Btouchscreen&qid=1713195690&sprefix=jetson%2Bnano%2Btouchscree%2Caps%2C230&sr=8-3&th=1

[30] ‘NF-A4x20 5V PWM’. Accessed: May 11, 2024. [Online]. Available: https://noctua.at/en/products/fan/nf-a4x20-pwm-178

[31] ‘OpenCV: cv::cuda::FarnebackOpticalFlow Class Reference’. Accessed: Apr. 23, 2024. [Online]. Available: https://docs.opencv.org/3.4/d9/d30/classcv_1_1cuda_1_1FarnebackOpticalFlow.html