Gesture Recognition for Enhanced Human-Robot Collaboration in Intralogistics Using CNN-LSTM Frameworks

Project Thesis in Electromobility-ACES

Friedrich-Alexander-Universität Erlangen-Nürnberg
Institute for Factory Automation and Production Systems
Prof. Dr.-Ing. Jörg Franke

Author: Vishakh Cheruparambath, 23240372

Supervisor(s): Prof. Dr.-Ing. Jörg Franke
Jakob Hartmann, M.Sc.

Submission date: 18.04.2025

Processing time: 5 months

Declaration

I hereby declare that I have written this thesis without outside help and without using any sources other than those indicated. This thesis has not been submitted in the same or a similar version to any other examination office and been accepted as part of an examination performance. All statements that have been adopted literally or paraphrased are marked as such.

Erlangen, 18.04.2025

Vishakh Cheruparambath

Contents

List of Figures

List of Tables

Abstract

Human-robot interaction is becoming increasingly important in dynamic situations. This includes intralogistics, where communication techniques are intuitive and critical for effective collaboration. Gesture-based interfaces provide a natural and seamless platform for such engagement. However, previous solutions frequently address static and dynamic gestures independently, relying on isolated modules, or lack real-time responsiveness, limiting their usefulness in fast-paced contexts. To address these problems, this study proposes a unified gesture recognition pipeline that integrates CNN-LSTM model to classify static and dynamic actions in real time utilizing a ZED 2i stereo camera. The systems dynamic model-switching mechanism, driven by natural motions such as waving or pointing, enables context-aware activation. The pipeline also includes gesture classification, command execution, and depth point mapping applying stereo depth data particularly for pointing gestures, which enable spatially aware robot actions. Unlike previous techniques, our system integrates all gesture-related features into a single, high-performance inference framework. The evaluation demonstrates that it performs accurately and responsively in real-world intralogistics settings, improving the intuitiveness and performance of human-robot collaboration.

Keywords: *gesture recognition, human-robot interaction, CNN-LSTM, intralogistics, stereo vision, pointing gesture, ZED 2i*

1 Introduction

Autonomous mobile robots (AMRs) are gaining widespread popularity in the field of Intralogistics. Since then material transport in the field of intralogistics often requires high physical labor by hand pallet trucks, tow tractors, and order picking trolleys [1]. Manual material handling is common in intralogistics because internal processes can be dynamical and complex in nature. This creates human adjustment and decision-making. However, relying on manual material handling makes inefficiencies, safety concerning issues, and high labor cost. To overcome these challenges, it is necessary to include smart, self-running systems in these facilities. A fleet of AMRs can autonomously manage the physical flow of material in the warehouses settings [2].

The chapter explains the motivation for studying the adaptive and user friendly human-robot interaction (HRI) and the objective of this thesis. It also gives an overview of the existing methods, showing their strengths and limitations. This work focuses on using CNN-LSTM model for gesture recognition tasks in intralogistics application.

1.1 Situation

AMRs provides a promising solution by enabling collaborative, distributed operations with reduced human effort and flexibility. As intralogistics shifts from hierarchical control strategies towards distributed, self-optimizing systems, it will be better able to accommodate dynamic environments, achieve higher throughput, and enable effortless human-robot collaboration. This leads to more efficient, scalable, and resilient material flow systems [3]. But independent robots need to translate between the analog world and the robot's digital internal computation. Since, the actual world is analog, noisy and changing often a robot needs to translate it into a digital format suitable for its internal computations so that it can communicate in dynamic environments effectively. Thus, the robot's capacity for perceiving, knowing, and doing is constrained inherently by its sensing, representational schemes, and real-time processing capabilities. Addressing these constraints involves not just technological advancements but also a more intrinsic integration of human-centered design thinking with interdisciplinary research in cognition, interaction, and perception. [4]

Modern warehouses are dynamic environments where a lot of activities take place. Activities such as moving goods, storing, picking, and packaging, all within big buildings used as distribution centers or manufacturing units. These areas use a mix of manual work and automated systems, with things like robots that move around, conveyor belts, forklifts, and vehicles that have sensors. With the increasing number of different products and changing demand, there's a need for smart systems that can understand their surroundings, move around on their own, and work together in real-time. Technologies such as stereo camera, LiDAR, SLAM, RFID, and machine vision helps the systems know where they are and handle objects, but people still need to set things up and keep an eye on everything, often using things like hand gestures to interact with these machines. [3]

1.2 Complication

A natural human-robot collaboration in intralogistics needs efficient gesture recognition for seamless communication between AMRs and users [4]. The KOLAMeRo project [5] is focused on natural and effective collaboration between people and AMRs. The systems must be capable of continuous learning from their surroundings and adaptive to new tasks with minimum expert involvement. Robots have already become integral parts in many industrial applications, offering benefits like less fatigue, increased speed, greater repeatability, and better accuracy. On the other hand, although they are limited by elements such as consistency, accuracy, and tolerance [6], manual procedures are quite dependent on the skills of human workers. Therefore, the integration of robotic systems with skilled human operators through efficient human-robot collaboration could overcome these limitations, leading to improved efficiency and productivity.

Conventional interaction methods such as keyboards, buttons, or touchscreens are often unnatural and hinder smooth collaboration between humans and robots. While speech recognition technologies are gaining widespread acceptance, they remain restricted by issues such as accent variability and environmental noise susceptibility. On the other hand, vision-based gesture recognition (i.e static or dynamic) is a contact-less and natural mode of communication that is very much in line with inherent human actions. But they requires high initial cost and trained workers.

1.3 Solution

The thesis aims to develop an adaptive and intuitive gesture recognition system to enhance human-robot collaboration in intralogistics. To achieve this objective, the system leverages deep learning methods in the form of stacking Convolutional Neural Networks (CNNs) for spatial feature extraction and Long Short-Term Memory (LSTM) networks for modeling temporal sequences. This model improves gesture detection accuracy and reliability by leveraging hand and sequence based recognition models. The long-term objective of gesture recognition is to facilitate human-robot collaboration through decreased reliance on conventional input devices and the ability to interact with humans using a more natural approach.

This thesis seeks to demonstrate the working model on a Linde Material Handling P50 ION tow tractor using a ZED 2i stereo camera from Stereolabs Inc. The core objective is to examine the applicability of gesture control under real intralogistics environment. By enabling contact-less interaction through gestures, this approach can enhance worker safety for example using an emergency stop command via closed fist gesture, or commanding AMRs to move to a particular ground point location using a pointing gesture. Hence, this can reduce the cognitive and physical load on human operators. The idea for gesture recognition system was inspired from the advancements in computer vision based researches in recent times.

For collecting training datasets, the depth added hand gestures in dynamic environments necessitated careful preprocessing, and sequence collections in a constrained environment. The resolution of these problems was essential for facilitating successful human-robot interactions, guaranteeing stable real-time inference, and optimizing model performance in real-world scenarios. The final objective is to provide effective human-robot interaction via real-time inference, ensuring smooth integration into a Robot Operating System (ROS 2) package for de-

ployment in autonomous systems inside intralogistics environments. The gesture recognition model can be further extended to multi modal interactions like speech recognition etc. This requires expertise in the field of Large language Models(LLMs) and Natural Language Processing(NLPs).

1.4 Structure of the Thesis

The thesis is organized into six main chapters. Chapter 1 provides the introduction to the topic, stating the situation, complication, solution, and the concise explanation of the thesis structure. Chapter 2 provides the background essential for this research, such as basic concepts on human-robot interaction, brain-inspired models, and deep learning techniques. The chapter describes neural networks, highlighting multilayer perceptrons, and convolutional, and recurrent neural networks, along with long short-term memory networks. Chapter 3 overviews related literature, demonstrating current sensor technologies, gesture datasets, and gesture recognition methods. It identifies the current research gap and proposes the approach employed in this project thesis.

Chapter 4 describes the approach, starting with an overview of the robotic system architecture, including hardware and software components. This section describes the method followed in training static and dynamic gesture classification models, details the suggested classification method, and explores the system's integration into the ROS 2 environment. Chapter 5 gives a report of the model evaluation results acquired from experimental analysis. Finally, chapter 6 concludes by summarizing the primary findings and contributions of the research and outlines potential directions for future research.

2 Fundamentals

The fundamental ideas of gesture-based human-robot collaboration are presented in this chapter. It starts with human-robot interaction and how it functions in see-think-act cycle, then moves on to human brain insights that motivate learning models. The chapter then covers key machine learning and deep learning concepts, which enable perception and recognition.

2.1 Human-Robot Interaction

Human-robot interaction (HRI) is an interdisciplinary subject that encompasses the scope of robotics, artificial intelligence, and human-computer interaction (HCI), where the ultimate focus is in creating robots that can collaborate with humans. Whereas most computing devices are disembodied, robots have a physical body that can detect and react to human events [4]. The major difference between HRI and HCI lies in the embodiment. Unlike HCI, which entails interaction with digital systems, HRI emphasises interacting with actual robots that operate in the real world and demand both technical and social knowledge. Embodiment affects functional capacity, but it also influences human perception, expectations, and relationships [4]. Siegwart, Nourbakhsh, and Scaramuzza (2011) in [7] state that autonomous robots follow a see-think-act cycle to perceive, decide, and act based on internal models and motion control. Figure 1, demonstrates a see-think-act architecture from perception to motion control. It shows how the robot perceives its environment and deduces information, and maps the environment in order to localize itself. It then plans a path based on its position and mission goals, and then actually carries out the motion through its actuators to influence the real world. [7]

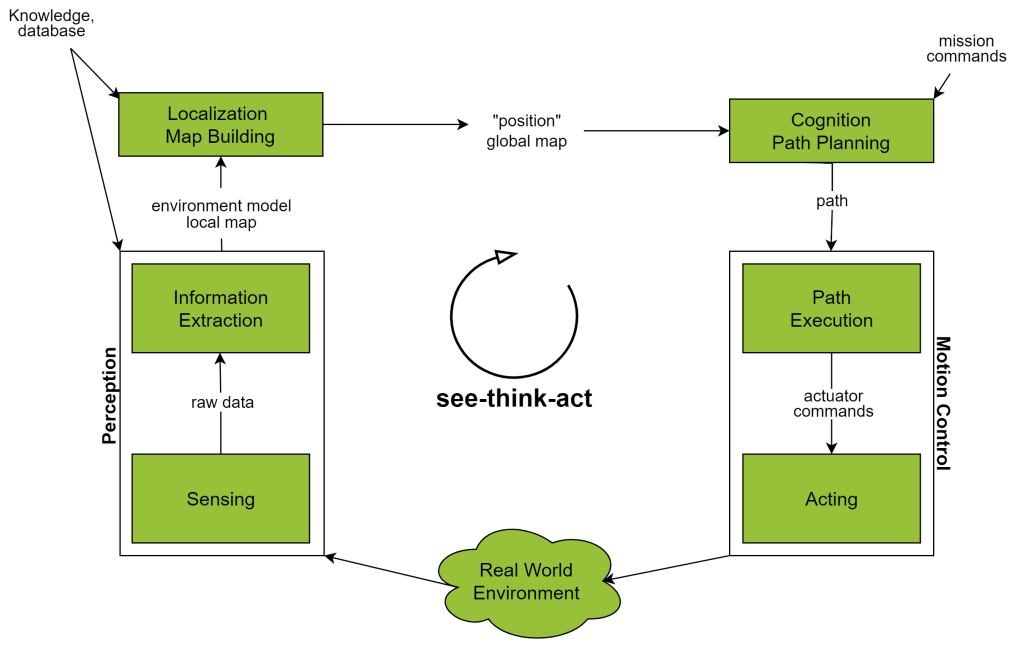


Figure 1: Overview of functional workflow of an autonomous mobile robot, illustrating how perception, cognition, and motion control work together in a continuous loop to interpret the environment, make navigation decisions, and interact with the real world. [7]

HRI bridges technological and design challenges, as robots are expected to operate in socially intricate and dynamic environments while offering interactions that feel natural and intuitive to humans. Robot stance, form, and sensing modalities all play significant roles in user experiences. HRI research provides insights into how humans think, feel, and interact socially with artificial beings in their environment. The aim of HRI research is to design robots that are able to seamlessly inhabit everyday human contexts. [4]

Based on proximity, human-robot interaction can be classified into two:

- **Remote Interaction**: Humans and robots operate in separate locations, often with spatial or temporal separation, and interact via networked or indirect communication methods.
- **Proximate Interaction**: Humans and robots share the same physical space, enabling direct and real-time interaction within a collaborative environment.

These interaction types shape communication strategies, system design, and autonomy levels, ensuring smooth integration into real-world applications [8].

2.2 Human Brain

The human brain is a sophisticated and extremely effective mechanism for processing sensory information, making decisions, and initiating actions. The brain is addressed in this chapter because it provides biological inspiration for the creation of artificial neural networks. Designing intelligent systems benefits considerably from an understanding of how the brain functions, learns, and responds. As seen in Figure 2, functionally it is a three-stage system comprising receptors (input), a neural (nerve) network (processing), and effectors (output). The receptors convert external stimuli into electrical impulses; the neural net processes the information and makes decisions; and the effectors translate neural outputs into physical actions.

Two forms of signal flow are known: forward transmission (left to right) and feedback (right to left), and through these the brain is able to make best and fast decisions [9]. For instance, sensory neurons send a (forward) signal to the brain when touched on a hot surface, and the brain uses that input to quickly send a (feedback) signal to the muscles, causing hand to pull away immediately.

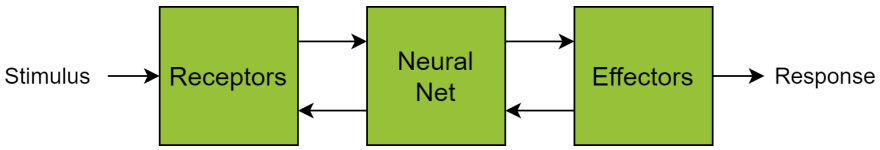


Figure 2: A simplified three-stage model of the human nervous system. Stimuli are detected by receptors and transmitted to the brain for processing, which then generates responses via effectors. Feedback system allow the system to adapt and refine its actions.[9]

Neurons are the key elements of the brain, communicating with each other through synapses, which are both structural and functional links. Although neuronal communication is slower than in electronic circuits, the brain overcomes this drawback by massive parallel processing, with approximately 10 billion neurons and 60 trillion synapses. This highly interconnected network,

together with the plasticity of the brain, enables it to set up and alter synaptic connections, thus allowing learning and flexibility. [9]

Neurons vary widely in form and function. A well-known type is the pyramidal cell, distinguished by its extensive branching, which allows it to connect with thousands of other neurons. These cells communicate through small electrical impulses known as action potentials, which travel down lengthy structures called axons. Despite the natural electrical resistance of axons, the brain cleverly orchestrates the rapid and synchronized transmission of these signals, allowing fast communication between different parts of the brain. [9]

2.3 Machine Learning

Machine Learning (ML) is a sub-field of artificial intelligence that focuses on creating systems that can learn from data and improve performance without requiring explicit programming. ML algorithms are intended to detect patterns in observed data and generalize them to create accurate predictions about previously unseen cases. Machine learning seeks to learn a mapping from input data $\mathbf{x}$ to output labels y by minimizing a loss function that measures prediction inaccuracy. A typical machine learning system includes data (for training and evaluation), models (for mapping inputs to outputs), loss functions (to assess performance), optimization techniques (e.g., gradient descent), and the capacity to generalize to new data. [10]

ML can be classified into three types. They are supervised, unsupervised, and reinforcement learning. Supervised learning involves training the model on labeled data, where there is a target output for each input; classification and regression are examples of these kinds of tasks. Unsupervised learning, however, is the task of finding structure in data without labeled outputs. The typical uses include clustering, dimensionality reduction, and density estimation. Reinforcement learning enables an agent to learn optimal decision-making through interaction with its environment and by receiving feedback in the form of rewards. In addition, by transforming raw data into more informative representations for learning, preprocessing and feature extraction also play important roles. The final aim of machine learning is to develop models that are able to make accurate and dependable decisions in real-life environments by generalizing from the training data. [10]

2.4 Deep Learning

Deep learning (DL) is a branch of machine learning that utilizes multi-layered neural networks to handle raw data without any pre-processing. The working of this approach lies in the mechanisms of human brain in handling data, whereby, as in the brain's neural network, deep architectures learn and identify features autonomously without being specifically programmed. Deep architectures construct hierarchical representations beginning with elementary features and advancing towards more abstract ones. [11]

This technique has shown better performances in applications like image classification, audio processing, speech recognition, and natural language processing. Major contributions to this field, was only possible due to advances in computational power, availability of large datasets, and improved training algorithms, making it to solve complex tasks more efficiently than conventional approaches. [11]

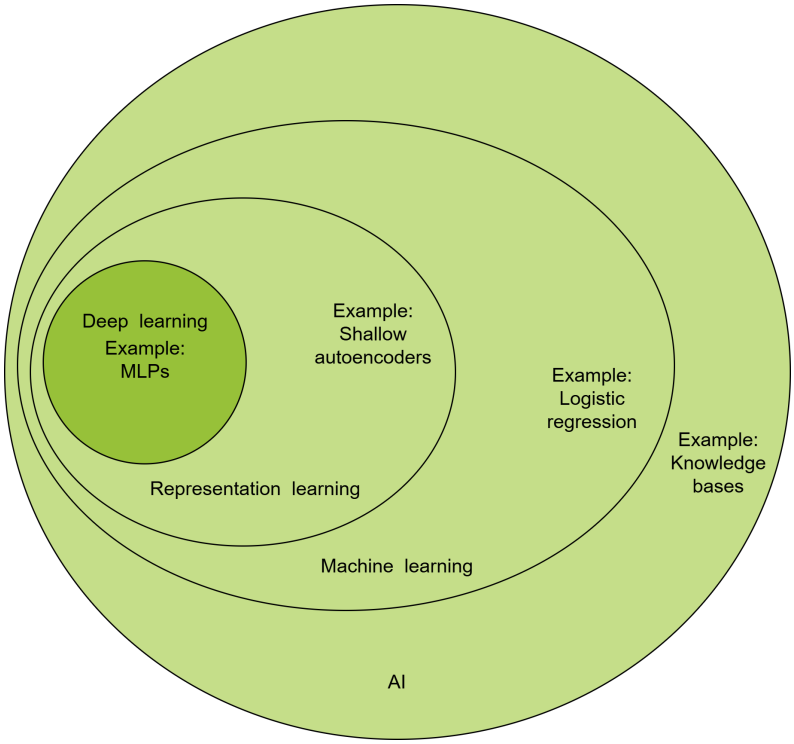


Figure 3: A Venn diagram that shows how deep learning, representation learning, machine learning, and artificial intelligence are related hierarchically. [11]

From figure 3, within the field of artificial intelligence, deep learning is positioned as a subset of representation learning, which is a subset of machine learning. Representation learning encompasses techniques that automatically discover useful features or representations from raw data and achieve better performance on classification or prediction tasks. A typical example of this is shallow auto encoders, which learn compressed representations of input data via an unsupervised process. Deep learning methods, extends by multilayer perceptrons (MLPs), generalize this concept by extracting hierarchical features at various layers. An example of an AI technology for each section of the Venn-diagram is included.

2.4.1 Neural Networks

Artificial Neural Networks (ANNs) form the foundational architecture underlying deep learning. ANNs, in short as Neural Networks form the basics computational model that gets inspired by the functionality of biological nervous system of human brain. Artificial neurons receives multiple inputs, processing it by a weighted connections, and then output through an activation function. The basic structure of a neural network consist of input layer, hidden layers, and output layer. Thereby forming a connected network that can learn complex patterns and data relations. [12]

Although in theory, a neural network can have any number of neurons that are connected randomly. But in practical applications it tends to arrange them in consecutive layers for more computational efficiency. A common setup has at least an input layer and an output layer, supplemented with one or more hidden layers inserted between them. Structuring connections in this way enables neural networks to offer a range of functions, from simple decision-making

functions. The functionality of a single artificial neurons in a network can be mathematically represented by the below equation 2.1. Here, x_1, x_2, x_3 forms the input features and w_1, w_2, w_3 are their corresponding weights; the bias term for shifting the activation threshold is also represented as b. A bias is a additional scalar applied to a neuron to modify the output value, thereby assisting the neural network in achieving a better alignment with the data. [12]

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b \tag{2.1}$$

In an artificial neural network, a neuron calculates the weighted sum of its input and applies a bias term. This calculated initial linear output is then passed through a non-linear activation function, which then required for facilitating the modeling of complex data patterns. Similar to synapses of human brain, the activation function determines whether the input is above a specified threshold. [12]

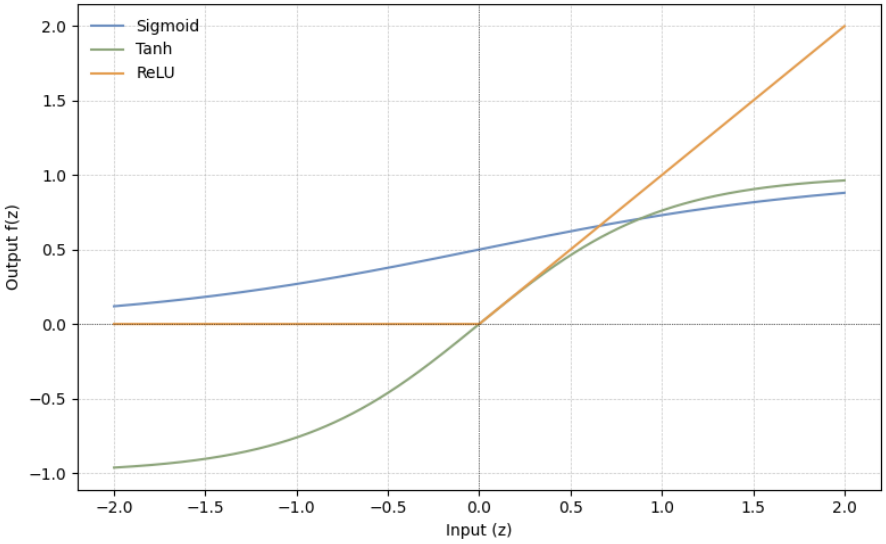


Figure 4: Common activation functions used in deep learning. [12]

As shown in Figure 4, common activation functions includes the sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU). Each constrain the neuron's output within a specific range and enabling the network to capture non-linear relationships. During training, the model adjusts its weights and biases to minimize prediction error, allowing it to learn and improve performance over time. [12]

2.4.2 Multilayer Perceptron

Multilayer Perceptron or MLPs is a form of feed-forward ANN with an input layer, one or more hidden layers, and an output layer as illustrated in Figure 5. The hidden layers, which cannot be observed from outside the network, allow the MLP to learn and represent complicated, non-linear relationships within the data. MLPs surpass single hidden layer networks in their ability to capture complex patterns and enhance learning capabilities. However, adding additional hidden layers does not necessarily ensure better performance because it causes the problem of overfitting and makes training more difficult. [12]

The universal approximation theorem states that an MLP with one hidden layer and a nonlinear activation function may approximate any measurable function if it has enough hidden units. In practice, MLPs tend to be built with more than one hidden layer in an effort to find the best trade-off between their learning capability and generalization performance. [12]

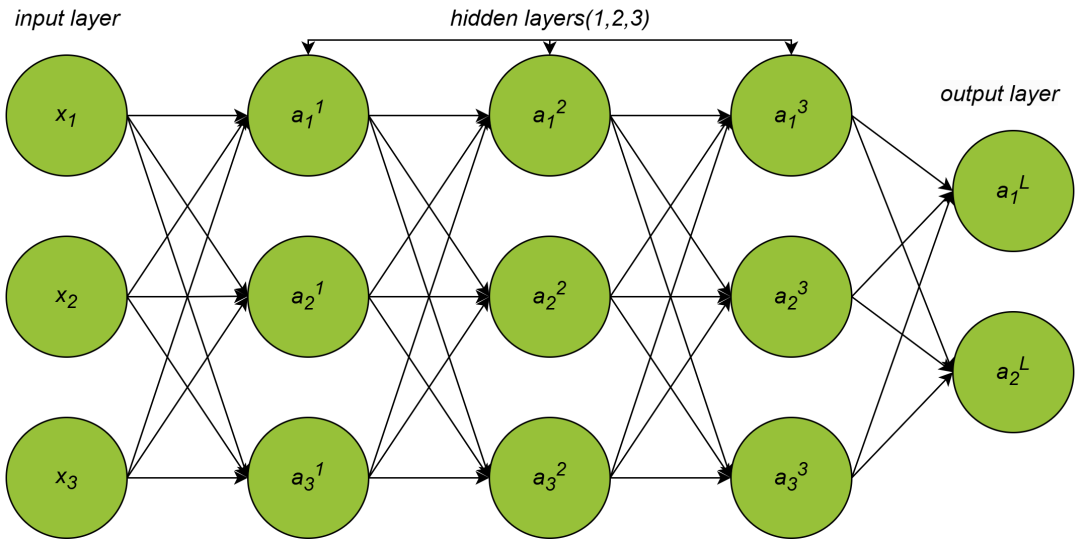


Figure 5: A multilayer perceptron (MLP) with three hidden layers in addition to one output layer. The neurons are labeled as a_i^l, where the index variable l indicates the layer index and i denotes the output index. [12]

2.5 Convolutional Neural Networks

Convolutional Neural Networks or CNNs, developed by LeCun in 1989 [13], are a special type of neural network that is specially designed to process data with a grid-like topology. Grid-like topology includes data in the form of a 1D grid of a single feature over time or 2D grid of pixel intensities for image data. CNNs have achieved great success in a wide range of applications like computer vision, natural language processing, and audio signal analysis. Unlike traditional neural networks, CNNs utilize convolutional operation instead of regular matrix multiplication in atleast one of their layers. They are computationally efficient in feature extraction, due to the exploitation in spatial and temporal local correlations. [11]

Convolution operation forms the basic mathematical expression that is deeply influenced in CNNs to extract useful features from input data. The operations tries to merge two functions or arrays, to form a third function that shows how one function interacts or modifies with other function. Convolution enables local, parameter-efficient, and translation-equivariant computations that is perfect for the structured input data like images, time-series data, and volumetric data. [11]

Continuous Convolution

In the continuous domain, convolution is expressed as an integral of a weighted average of the input function and another function which is known as kernel or filter. Continuous Convolution of two function, can be expressed mathematically by the following equation 2.2.

It consists of three main elements. The input function which may represent a signal is shown as $x(a)$; the kernel or weighting function by $w(t-a)$; and the resulting output $s(t)$. The convolution process combines past input values with weights defined by the kernel to produce a transformed or enhanced version of the original signal. Kernels are usually designed to give prominence to recent inputs or detect specific features like edges or periodicity. In practical applications, the kernel may be restricted; for example for e.g., $w(a) = 0$ for $a < 0$ to ensure causality, or normalized to serve as a probability density function (PDF). [11]

Table 1 refers to the notations used for explaining continuous, discrete, and 2-D convolution.

$$s(t) = \int_{-\infty}^{\infty} x(a)\, w(t-a)\, da \tag{2.2}$$

Table 1: Notations used for Convolution Networks.

Symbol	Description
$x(a)$	Input function (continuous), e.g., signal
$w(t-a)$	Kernel or filter function (continuous) applied to past values
$s(t)$	Output function after convolution (continuous domain)
$x[a]$	Input sequence (discrete)
$w[a]$	Kernel/filter sequence (discrete)
$I(m,n)$	Input image (2-D convolution)
$K(m,n)$	2D convolutional kernel (image processing)
$S(i,j)$	Output feature map from 2D convolution

Discrete Convolution

Data is being sampled at fixed intervals during most machine learning applications. This means we work with discrete signals rather than continuous signals. Accordingly, convolution is carried out using summation instead of integration. The discrete convolution between an input sequence $x[a]$ and a kernel fucntion $w[a]$ is given by equation 2.3.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a) \tag{2.3}$$

In this context, $x(a)$ represents the input values, $w(t-a)$ is the kernel (or filter) that defines how each input is weighted, and $s(t)$ denotes the resulting output. In the context of CNNs, both the input and kernel are finite tensors like an image or a sequence. The convolution produces

what is often called a feature map. Since the input and kernel are finite in real applications, the summation is only carried out over the relevant non-zero regions, rather than the entire range of integers. [11]

Two-Dimensional Convolution

For 2-dimensional data like gray scale images, the convolution operation spans both spatial dimensions. Let $I(m, n)$ represent the input image, $K(m, n)$ represents convolution kernel, and $S(i, j)$ the resulting feature map respectively. The mathematically expressed of 2-D discrete convolution can be shown as equation 2.4.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)\, K(i-m, j-n) \tag{2.4}$$

In this formulation, the kernel is flipped before being applied to the input, reflecting the conventional definition of convolution from signal processing. Alternatively, an equivalent form based on the commutative property of convolution is more frequently used in machine learning libraries by the expression 2.5.

$$S(i,j) = \sum_m \sum_n I(i-m, j-n)\, K(m,n) \tag{2.5}$$

This operation effectively slides the kernel across the input, computing a weighted sum at each location. As a result, the convolution highlights localized features in the input, such as edges, corners, or textures, enabling the network to capture spatial hierarchies in the given data. [11]

To summarize, the convolutional layers are designed to take features from input data by implementing learnable filters that can slide along spatial dimension. A dot product is computed using a filter, by its weights and local region of input, where sliding in terms of a stride S, with optional zero-padding P to retain spatial dimensional information.

The output size of a convolution operation can be calculated as per equation 2.6. Here W is the width of input, P the padding, F the filter size, and S the stride. The number of filters used determines the depth of output. After convolution, an activation function is typically applied to introduce non-linearity into the model.

$$\text{Output Size} = \frac{W - F + 2P}{S} + 1 \tag{2.6}$$

Pooling exploits the fact that neighboring pixels in images look like one another. Hence, effective down-sampling, i.e., retaining only the maximum or mean of a local patch, is found to be good for modeling. Pooling layers come after convolutional layers to decrease spatial size but preserve important features. Max pooling and average-pooling are the most used methods in pooling, which operate in small windows (e.g., 2×2) with a specified stride. Pooling reduces cost of computation by decreasing the number of parameters and helps translate invariant feature extraction.

The basic components to construct a CNNs are convolutional layer, the pooling layer together with the dense layer. In contrast to dense layers, which have different weights for each pair

of input-output, convolutional layers use parameter sharing, where the same set of weights, a filter, is applied to various spatial locations. This significantly decreases the number of parameters and enhances model generalization, particularly for high-dimensional data such as images.

Batch normalisation layers can be incorporated into CNNs to expedite training by addressing internal covariate shift. As previously stated, the input to a batch-norm layer is normalised by a mean and a variance that are independent of the other layers. As a result, batch normalization simplifies the interactions between layers in the gradient update and allows for higher learning rates, hence accelerating training. [12]

2.6 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are presented as an extension to the conventional feed-forward networks to process time-dependent or sequential data. As demonstrated in Figure 6, rather than presenting all the inputs at once, RNNs receive a series of inputs sequentially over time. The concept of recurrent network is to visualize it as unfolding through time. The same weight are applied repeatedly across each of the time step.

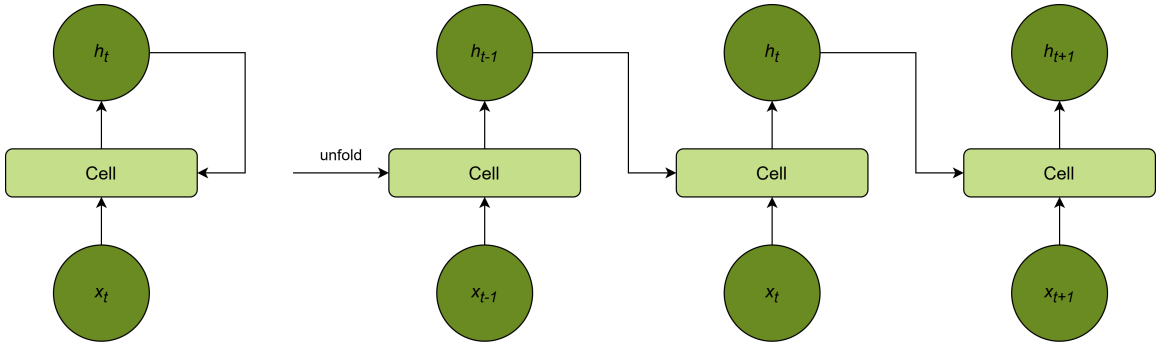


Figure 6: A simple RNN Architecture. At each time step, the cell takes the current input x_t and previous hidden state h_{t-1} to compute a new state h_t, which carries forward accumulated sequence information. This recurrent structure enables modeling of temporal dependencies in sequential data. [12]

Unlike feed forward (FF) networks that process each input independently, RNNs maintain a hidden state h_t that captures historical context across time. At each time step t, the network updates this hidden state based on the current input x_t and the previous hidden state h_{t-1}, formulated using equation 2.7.

$$h_t = \phi(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \tag{2.7}$$

The output at each time step t is computed from the hidden state as 2.8:

$$y_t = W_{yh}h_t + b_y \tag{2.8}$$

The terms W_{ih}, W_{hh} are input to hidden and hidden to hidden learnable weight matrices and W_{ho} for hidden to output weight matrix respectively. b_h and b_y are bias terms, and ϕ is a non-linear activation function such as tanh or ReLU. This structure allows the RNN to leverage temporal dependencies in sequential data.

RNN Training

For training RNNs, a method called Backpropagation Through Time, also known as BPTT is used. It calculates gradients over the whole sequence by unfolding the network in time and using the chain rule. It is the same as regular backpropagation but considers how hidden states depend on each other in time. One big problem in training RNNs is the vanishing or exploding gradient problem; a serious obstacle for very long sequences. This is due to the fact that repeated multiplication of gradients during backpropagation through time can yield very small (vanishing) or very big (exploding) values depending on the magnitude of the recurrent weight matrix eigenvalues. This made the invention of more complex architectures like Long Short-Term Memory networks(LSTMs) and Gated Recurrent Units(GRUs) which can handle long-term dependencies more effectively. The idea of unfolding iterative networks into layered networks, as in the early work of Rumelhart et al. [14], formed the basis of these novel sequence learning methods.

The article [14] also considers the equivalence between layered feedforward networks and recurrent networks by unfolding the temporal dimensions of recurrent connections. In a recurrent network, each time step can be viewed as an independent layer in a feedforward network as well, hence creating a deep network that has the same set of weights applied across varying time steps. This assumption is key in BPTT, a variant of backpropagation specifically designed for RNNs. The weight sharing across time steps significantly reduces the number of parameters, allowing RNNs to generalize across varying sequence lengths and inputs. [11, 12]

In BPTT, the gradients of the losses are aggregated over time, wherein the error at each time step is included into the gradient of shared weights. Temporal dependencies are captured by altering the standard gradient descent update. One example of this type of weight update used is given by 2.9:

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{2.9}$$

An improved version 2.10 of this update incorporates momentum, allowing the network to build up speed in the direction of persistent gradients:

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w(t)} + \alpha \Delta w(t-1) \tag{2.10}$$

Here, t is incremented by 1 for each sweep through the whole set of input-output cases; η is the learning rate; α is the exponential (decay) factor between 0 and 1, determining the relative contribution of the current and previous gradients to the weight update; E represents the total error; and w denotes the network weights. [14]

2.7 Long Short-Term Memory

LSTM networks are a specific type of RNNs intended to describe sequential input while effectively capturing long-range dependencies. It was introduced by Hochreiter and Schmidhuber in 1997 [15], as a solution to the vanishing and exploding gradient problems.

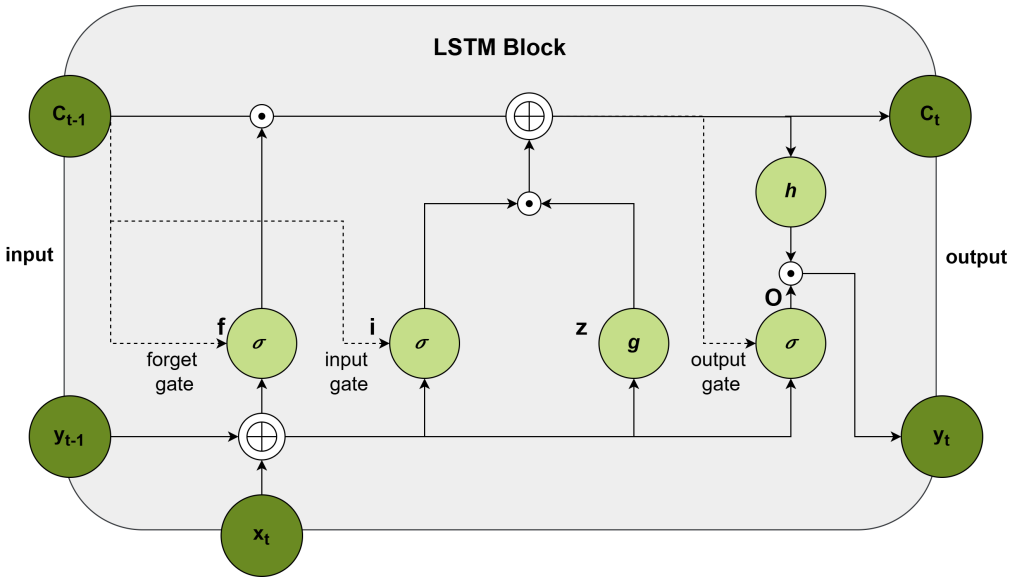


Figure 7: An illustration of LSTM Cell Architecture. [16]

Table 2: Legends used for LSTM Block Notation.

Symbol	Description
———	Connection
--- ---	Peephole connection
$\odot$	Element-wise multiplication
$\oplus$	Sum over all inputs
σ	Gate activation function (always *sigmoid*)
g	Input activation function (usually *tanh*)
h	Output activation function (usually *tanh*)

LSTMs provide a special memory cell that can retain information across several time steps, in contrast to standard RNNs, which suffer this problem while processing lengthy sequences data. This memory feature overcomes a major drawback of traditional RNNs by allowing the model to record temporal patterns on lengthy sequences without losing important prior information. [15]

LSTMs solve this issue by adding a special structure of recurrently connected sub-networks called a memory cell. It is able to regulate information flow using non-linear gating units, while maintaining its state over time. The behavior of memory cell is controlled by three gates i.e. input, forget, and output; which decide what to keep, what to modify, and what to discard at any given time. It also involves activation functions and peephole connections. [16]

- **Block Input:** Let us assume we are processing a sequence over time. At each time step t, the LSTM receives an input $x(t)$, remembers its previous output $y(t-1)$ of last iteration, and keeps track of a cell state $c(t-1)$ from the past. Equation 2.11 presents a generated block input, that act as a candidate value for updation of memory. This is computed using current input and previous output values, where W_z and R_z are weight matrices, and b_z is the bias weight vector. [16]

$$z(t) = \tanh(W_z x(t) + R_z y(t-1) + b_z) \tag{2.11}$$

- **Input Gate:** The input gate decides how much of this new information $z(t)$ should be written into the cell. Here, point-wise multiplication $\odot$ represents element-wise multiplication between vectors. The parameters W_i, R_i, and p_i are the respective weights associated with the input $x(t)$, the previous output $y(t-1)$, and the previous cell state $c(t-1)$, while b_i is the bias vector corresponding to the input gate [16]. The formulae can be written as 2.12.

$$i(t) = \sigma(W_i x(t) + R_i y(t-1) + p_i \odot c(t-1) + b_i) \tag{2.12}$$

- **Forget Gate:** The forget gate at time step t determines how much of the previous memory should be retained. The activation value $f(t)$ is computed using the current input $x(t)$, the previous output $y(t-1)$, the previous cell state $c(t-1)$, the peephole connection weights, and a bias term b_f [16]. This is mathematically expressed by equation 2.13:

$$f(t) = \sigma(W_f x(t) + R_f y(t-1) + p_f \odot c(t-1) + b_f) \tag{2.13}$$

- **Cell State:** Combining the outputs of these gates, the cell state is updated as shown in 2.14, where $\odot$ denotes element-wise multiplication. This equation ensures that relevant new information is added while unnecessary old information is forgotten. [16]

$$c(t) = z(t) \odot i(t) + c(t-1) \odot f(t) \tag{2.14}$$

- **Output Gate:** The output gate as shown in equation 2.15 determines what portion of the current cell state should act on the output. [16]

$$o(t) = \sigma(W_o x(t) + R_o y(t-1) + p_o \odot c(t) + b_o) \tag{2.15}$$

The last hidden state is calculated by applying a non-linearity to the new cell state. Equation 2.16 computes the final output of the LSTM block by applying a non-linear activation function $g(\cdot)$ to the updated cell state $c(t)$, and modulating it with the output gate $o(t)$ through element-wise multiplication. [16]

$$y(t) = g(c(t)) \odot o(t) \tag{2.16}$$

In the preceding stages, σ, g, and h represent point-wise non-linear activation functions. The logistic sigmoid $\sigma(x)$ is used as the gate activation function, while the hyperbolic tangent is commonly applied for the block input and output activations. [16]

Together, these gates enable the LSTM to effectively learn what information to retain, update, or forget over long sequences, making it highly effective for sequential tasks such as time-series forecasting, language modeling, and speech recognition.

3 Related Works

The chapter begins with state of the art section in detail. Emphasizing their benefits in adaptability and resilience, it describes the change from conventional rule-based models to deep learning techniques. In the framework of developing responsive systems, some difficulties including sensor choice and gesture classification are addressed. The chapter also compares current models and supports the selected method for the use in intralogistics. Finally, it presents the suggested modular gesture recognition system, intended for real-time operation with both static and dynamic gestures provided to various interaction requirements.

3.1 State of the Art

This section aims to give an overview on various gesture recognition methods, with a particular focus on hand gesture recognition using deep learning architectures. In the last few years, the robotics and automation industry has experienced faster growth in human-centered environments like factories, warehouses, hospitals etc. One of the driving forces behind these trends and findings is artificial intelligence and its advancements, which have enabled more natural HRI modalities. Surprisingly, gesture-based communication has been a point of interest because it presents an available and stable means of communication other than speech, particularly in a noisy or dynamic environment. As a result, hand gesture recognition has become a significant research field, facilitating a wide range of diverse applications like robotic assistance in medical field, industrial cobots for collaborative tasks.

This chapter begins with an exploration of various sensor technologies, gesture recognition systems, and gesture classification in particularly on vision-based algorithms. It then addresses some key challenges like selection of appropriate sensory inputs and categorization of gesture types, which are essential for building both strong and responsive interactive systems. The thesis is primarily focused on achieving human-robot collaboration through deep learning based gesture recognition. To validate this assertion, the chapter follows a detailed understanding of the key elements of gesture recognition. This is followed by a review of existing models, their characteristics, and the justification for selecting the most suitable approach for the given intralogistics application.

3.1.1 Sensor Technologies

Sensor technologies plays a critical role in capturing the input data to preprocessing. In recent years, a wide range of sensor technologies has been explored for gesture-based human-robot interaction. Some of them are marker [17], single camera [18], stereo camera [19], and depth sensors [6]. A major drawback in single camera-based approach is discussed in [20], where systems robustness is affected due to restricted angle view. Fottner et al. (2021) discuss the modern intralogistics scenarios where autonomous systems rely heavily on advanced sensor technologies like stereo cameras to perceive and interact with their environment. Further emphasize the importance of 3D perception in intralogistics, where stereo cameras and LiDAR enable precise localization, and RFID systems ensure reliable item tracking [3]. In addition, safety-critical sensors such as laser, light barriers, infrared, and vision-based human detection systems are required to provide safe and effective collaboration [3]. Gesture recognition

has been dominated by vison-based sensors [21]. Additionally, non-image-based sensors like data gloves requires wired connections, accelerometers, gyroscopes, although they often limit natural hand movement and require calibration [22].

Recent study on multi-modal approaches from Börold, Schweers and Freitag (2023) in [23] combined visual, auditory, and tactile modalities to reduce cognitive load and enhance usability in industrial environments. These studies highlight the importance on selecting context appropriate sensors to balance robustness, usability, and system complexity.

3.1.2 Gesture Dataset

In gesture recognition applications, a wide range of datasets has been used, each addressing specific application scenarios and modalities. Current gesture recognition research has focused on creating robust datasets that address variability in user, environment, and gesture types. The HaGRID dataset in [24] is unique in its size and variability, with over 550,000 images from 18 intuitive gesture classes gathered from over 37,000 subjects in varying lighting conditions, scenes, and subject-to-camera distances, accommodating both static and dynamic gesture modeling. Lieret et al. in [25] suggested a gesture dataset for the control of unmanned aerial vehicles (UAVs), in which gestures are angle-based skeletal joint gestures and are processed in real-time from RGB-D data, thus providing safe and interactive drone control in industrial environments. Similarly, Guyon et al. [26] introduced the ChaLearn Gesture Dataset, which contains multi-modal annotations in terms of RGB, depth, and skeleton data, while emphasizing continuous gesture recognition in a challenging real-world scenarios.

Cucurull (2022) in [27] suggested the HAGIL model, an incremental continual learning model that can learn static hand gestures continually from RGB images via hand landmarks. The HandGest dataset, comprised of 21 gestures, to test continual learning capabilities under real-world HRI conditions using an inexpensive RGB camera system. Similarly, Peral et al. (2022) in [28] used custom IPN Hand dataset, which consists of 200 videos containing 4218 consecutive gesture instances distributed across 14 classes, making it particularly well-suited for real-world continuous gesture recognition applications.

3.1.3 Gesture Recognition

In a gesture survey by Mitra and Acharya (2007), they defines foundational idea on defining types of gesture, namely static and dynamic. In static gesture, the user estimate a certain configuration or pose that will be intutive, whereas dynamic gesture comes with three phases: with a prestroke, stroke, and post-stroke phases. The prestroke prepares or transitions into the gesture, the stroke is the core movement conveying meaning, and the post-stroke concludes or transitions out of the gesture. Sometimes a gesture can also be altered by the context of preceding and following gesture. They also classified gestures based on three types: 1) hand and arm gestures involving symbolic movements and sign language, 2) head and facial gestures expressing emotions and indicate direction of focus, and 3) full-body gestures involve greater movements such as dancing or walking, commonly used in interactive or rehabilitative settings. [17]

During 2018, Liu and Wang in [6] provided a comprehensive method to solve gesture identification tasks in their review paper. This consists of three technologies particularly based on

visual features, learning algorithms, and skeleton based models. Visual features are broadly based on color, local features, shape and contour. A gesture identification based on kinect-based computer vision by Han et al in [29], reviewed the way RGB-D sensing enhances computer vision applications like gesture recognition, activity analysis, and indoor 3D mapping with more reliable and lighting invariant data. While enabling real-time applications in practice, they also indicated limitations such as range limitations and the absence of proper data fusion algorithms. They summarize that depth-based approach dominates RGB-based approach. To support this statement, contributions from Elmezain et al. [19] shows a real-time gesture recognition system using stereo color image sequences and 3D depth maps for accurate segmentation and tracking of hand motion trajectories. They highlights the effectiveness of depth information.

Gesture recognition systems have seen significant advancement through a variety of model based methodologies. Based on the algorithms used, it can be broadly classified into two; namely mathematical model based and soft computing based approaches [17]. Mathematically driven approaches involves traditional algorithmic techniques which models the temporal nature of dynamic gestures by learning different state transition over a period of time. These models struggles with unstructured inputs such as raw video data from cameras and as the complexity of gestures increases, state explosion of these models gets less efficient towards large-scale applications.

Following the discussion from model-based gesture recognition approaches, it is equally important to consider the enabling technologies for capturing and training gesture data. This can be categorized as: vision-based and contact-based approaches [21]. This thesis concentrates mostly on vision-based recognition architectures that are more cognitive and natural for humans to interact with rather than contact-based systems. Contact-based methods are based on physical contact with devices, e.g., data gloves, or multi-touch screens. But they tend to make users wear specialized gear, which proves interruption and non-intuitive, especially for untrained or inexperienced users. Although such systems achieve a high degree of detection accuracy, their usability is restricted by discomfort, limited degrees of freedom, and also possible health issues, such as allergic reactions to the device materials [30]. Thus, the remaining part of this chapter centers around vision-based approaches.

3.1.4 Gesture Classification

Current studies in gesture recognition through the utilization of neural networks have yielded consistent results. Yet the proper capturing of temporal dynamics is still a key issue in building an effective, real-time system for effortless human-robot interaction. Terreran et al. [31] successfully applied a skeleton-based approach to stable real-time full-body action and hand gesture recognition. Their system uses 3D keypoints from OpenPose and ensemble classification, demonstrates the strength of skeleton representations in dynamic and indoor environments. While Du et al. [32] proposed a CNN-based method that encodes skeleton sequences as images to capture spatio-temporal features effectively, Li et al. [33] addressed the limitations in temporal modeling by combining LSTM and CNN to leverage their respective strengths in learning long-term temporal dependencies and spatial features from 3D skeleton data. Although, CNNs can effectively model spatial patterns, they discard temporal information when sequences are represented as images. LSTMs, on the other hand, preserve temporal dependencies and thus are well-suited for sequential gestures [16].

Zhang et al. [34] proposed a deep architecture that combined 3DCNN, bidirectional ConvLSTM, and 2DCNN for learning spatiotemporal features efficiently for gesture recognition, also enabling robustness to varying gesture durations and modalities. On the basis of this, Zhang et al. [34] presented a better model architecture which takes advantage of 3D separable convolution coupled with ConvLSTM and attention mechanisms. The aim was at reducing the computational redundancy with robust recognition capacity in dynamic noisy environments. Building on these concepts, Nayan et al. [35] proposed a CNN Bi-LSTM-based multimodal continuous hand gesture recognition method, fusing RGB, optical flow, segmented hand information, and temporal difference features for improved input representation. Their work reflects the increasing direction of fusing multiple modalities with sequential models to achieve better robustness and continuity for gesture recognition. Likewise, Xia et al. [36] designed a lightweight LSTM-CNN model specifically for mobile sensor-based activity recognition. This model uniquely applies LSTM layers before CNNs to capture temporal dependencies prior to spatial abstraction. Their network focuses on effective temporal-spatial feature extraction and obtains competitive performance on benchmark datasets with very few parameters. It is best suited for real-time, resource-conscious gesture recognition systems [36]. To summarize, table 3 shows the advantages and disadvantages associated within respective models.

Table 3: Discussed Model-based Gesture Recognition Approaches. This table solely includes approaches related to the project's proposed methodology.

Model / Architecture	Advantages	Disadvantages
CNN-based skeleton encoding [32]	Encodes skeleton sequences as images; useful for spatial features	Discards temporal dependencies during image conversion
CNN + LSTM [33]	Combines spatial and temporal learning; ideal for sequential gesture modeling	Requires large datasets and high computational resources
3DCNN + ConvLSTM + 2DCNN [34]	Learns spatiotemporal features effectively; handles varied durations	Complex multi-stage architecture; requires fine-tuning
3D Separable Conv + ConvLSTM + Attention [34]	Reduces computation while preserving accuracy; handles noisy data well	Increases architectural depth and tuning complexity

Continued from Tabelle 3 on the next page

Discussed Model-based Gesture Recognition Approaches. This table solely includes approaches related to the project's proposed methodology. (continued)

Model / Architecture	Advantages	Disadvantages
Multimodal CNN–BiLSTM [35]	Uses RGB, optical flow, and temporal features; handles continuous gestures robustly	Requires synchronized inputs and preprocessing
LSTM + CNN [36]	Lightweight and mobile-optimized; captures temporal patterns early	Less effective for capturing spatial features
Skeleton + Ensemble Classifier [31]	Real-time and privacy-safe; full-body and hand gesture recognition using keypoints	Accuracy depends on precise skeleton detection; affected by occlusion

3.2 Research Gap

The field of human-robot interaction has seen huge improvements in the past few years. One of the main contribution to this would be advancements in gesture controls. Gesture recognition has evolved from classical rule-based and mathematical techniques, such as Hidden Markov Models (HMMs) [37], to deep learning-based systems that use spatiotemporal features of vision-based input. As discussed by Mitra and Acharya (2007), traditional approaches tend to be low in scalability, robustness to noise, and context-dependent modeling, making them less appropriate for dynamic real-world environments [17]. Although they are interpretative, they cannot offer the required adaptability for intricate, unconstrained human-robot interaction.

According to [38], a structured HRI taxonomy analysis is based on context, robot type, and team roles but most gesture recognition systems disregard these factors, focusing on one task at a time. In FiFi [39] project, they demonstrates real-world gesture-driven AGVs and functional modes of operation[1]. They implemented following mode and maneuvering modes as designed. But some limitations in obstacle detection is being addressed at the end. This reflects an advanced gesture systems that are both technically solid and contextually aware, capable of responding to shifting user roles and interaction contexts [38, 40]. However, the functional segmentation of gesture control into separate operational modes is useful, but it limits flexibility in human intention interpretation during multitasking or transitional workflows [1].

The majority of deep learning architectures in current use prioritize performance over interpretability and hardware efficiency, generally requiring substantial computational power and

labelled data.There is still a gap between modular gesture vocabulary i.e as seen in FiFi's user-centric gesture design and contemporary sequence modelling approaches capable of handling ambiguous or overlapping gestures in real time. Furthermore, context-driven decision-making e.g., utilising head orientation, posture, or proximity is rarely integrated into gesture classification despite the fact that it can contribute for seamless HRI.

ROS 2 has developed as a strong middle-ware for mobile and self-driving robotic systems. They support in navigation, SLAM, and human tracking. But the use of flexible and context-adaptive gesture recognition with it remains limited. Even though recent research, such as that by Lindner et al. (2023) and Phueakthong et al. (2021), demonstrates gesture-driven robots in ROS 2 environments [41, 42], these papers are concentrated more on ROS implementation rather on gesture recognition systems. There is a less discussion of functional modes in which the system can seamlessly decode alternate gestures on the basis of relevance in an available context. For example, switching to static gestures for critical commands and then back to dynamic gesture recognition mode for navigational and user-centric operations can bring multiple models together. This suggests a broader gap to create modular, multi gesture interfaces for ROS 2 that can adapt switching to different recognition models in real-time based on operating state.

Therefore, this thesis identifies a key issue in the development of gesture recognition systems that are context aware, functionally adaptive, and suited for real-time collaborations. This encompasses requirements to leverage depth and stereo vision information to gain full spatial comprehension, combining static and dynamic gestures within an integrated continuous recognition model with task prioritize knowledge, and limit dependence on pre-segmented input by applying temporal modeling methodologies, e.g., a CNN-LSTM architectures. In addition, the development of task-oriented and ergonomic gesture vocabularies is not well-studied, particularly in systems where natural and intuitive command interfaces are paramount for operational efficiency and safety. Though system-centric approaches can be technically oriented, and needs only simple recognition of unique gestures are defined. They can lack in connecting the gestures with its respective gesture commands. The core of a recognition system should be intuitive, user-centric, easy to understand and memorize, have a semantic reference to its functionality and ergonomics.

3.3 Proposed Methodology

This thesis introduces a real-time, modular gesture recognition system. They are adaptive to human-robot collaboration in intralogistics applications, focused on usability, robustness, and system-level integration. The system supports both static and dynamic gesture modalities, functionally distinguished based on the urgency and functionality of the operation. Static gestures (like open palm, closed fist, etc.) are linked to safety-critical operations such as pause/resume and emergency stopping situations respectively. Dynamic gestures (like waving, pointing) are reserved for non-critical high-level commands only, such as initiating user login/logout or initiating a target-point for robots navigation respectively.

The system is embedded with a ZED 2i stereo camera that captures spatial and motion features, leveraging both RGB and depth information to make gesture detection more robust under varying lighting and environmental conditions. For static gesture classification, the method uses a lightweight MLP (Multilayer Perceptron) model trained on keypoint landmarks from

google mediapipe hand and pose estimation keypoints. It has shown some exceptional results in terms of efficiency and accuracy of model [43, 44]. For dynamic gesture recognition, it uses a CNN-LSTM model trained on numpy array file(.npy) datasets containing frame sequences. This model learns motion patterns over time and is trained to classify sequences of gestures in the last five frames received during inference. This second model uses a highly connected network to extract a temporal representation of the detected hand, estimate its pose, and forecast the gesture. This is a simple, yet effective solution that operates in real time and is resistant to uncontrolled situations with changing backdrops and lighting conditions.

A runtime model-switching is used wherein the system keep on looking for movements in frames. When a correct dynamic gesture (e.g., waving the hand) is found, it starts dynamic recognition model, processes the sequence, and performs the corresponding high-level task such as user login. The system then goes back to the static model, if no movement is detected for a period, then processing safety critical gesture commands. This allows for seamless mode switching and guarantees operation continuity. To enhance usability and promote intuitiveness, a custom set of gesture commands has been defined, specifically designed for frequent intralogistics tasks and developed to be ergonomically simple to recall and execute. The commands have a clear semantic relationship with the corresponding robotic actions.

The gesture recognition system is fully incorporated in a ROS 2-based framework, where gesture outputs are fed through dedicated gesture and state-space ROS 2 nodes. These nodes interact with the navigation stack to manage high-level commands, i.e., positioning, stopping, and task assignment. This enables seamless communication between the user and the AMRs. Through the combination of temporal modeling, stereo-depth perception, and functional gesture mapping within a ROS 2 system, this approach presents a versatile and scalable solution for real-time human-robot collaboration within complex logistical settings.

4 Methodology

In this section, a successful methodology of the intended gesture recognition system is implemented, to attain the desired objectives of the project thesis. This chapter contains the comprehensive implementation process starting from sensor integration, designing of system architecture, ROS 2 integration, and finally real-time deployment. The implementation process involves three key steps: starting from the initial setup of the hardware and sensing platform, to designing advanced gesture recognition architecture and its state model, followed by model definition, training, and deployment, and ultimately the integration of consolidated gesture recognition model inference into the ROS 2 platform.

The initial step contains the setup of key hardware elements required for visual perception of gestures. A ZED 2i stereo camera is employed to obtain synchronized RGB and depth streams that are important for hand gesture recognition and spatial context interpretation. Mounted on a mobile robot platform that has onboard processing, the sensor enables both accurate gesture localization and visual recognition, especially for pointing movements that depend on depth information. Then the system's high-level architecture is defined to handle the data flow from perception to control. It involves frame preprocessing, googles mediapipe driven keypoint extraction for hand and pose landmarks, and conditional model switch logic for gesture classification. The architecture ensures real-time responsiveness via dynamic switching between static and dynamic models based on user movement, and publishing gesture outputs to downstream ROS 2 nodes for robot control. This architecture also includes the definition of gesture-command mappings and a runtime state model that governs system transitions during interaction.

Next, the models for gesture classification are defined, trained and subsequently evaluated. A lightweight MLP model is deployed to classify discrete static hand poses, whereas a CNN-LSTM based model is utilized for recognizing temporally extended dynamic gestures. Each model is separately trained on corresponding datasets format consisting of keypoint coordinates values and motion sequences respectively. Further, a model for object detection and its parameter are discussed. These models are shown to be efficient regarding classification accuracy in addition to inference reliability, thus their applicability in real-time and real-world robotic interaction.

Finally, the integrated system is deployed in a ROS 2 communications framework so that gesture outputs can influence navigation and task execution directly. The recognized gestures are mapped to ROS 2 messages, which are interpreted by the robot controller to trigger actions like moving to a certain region, stopping movement, or starting task sequences. The following sections detail each step in the implementation process, including the results from experimental trials that support the overall effectiveness of the system. The performance of the gesture recognition models in real-world is assessed in the next chapter, along with recommendations for possible advancements and future developments.

4.1 Overview of Robotic System Architecture

This section presents the robotic system configuration, which includes the hardware arrangement, system architecture, and the gesture recognition state model employed for real-time

human-robot interaction. Figure 8 shows the AMR for this project thesis. The AMR configuration is detailed in the following sub chapters.

4.1.1 Hardware and Sensor Configuration

The Linde P50C electric tow tractor served as the base working model for the mobile robot platform utilised in this project. This was modified for autonomous operation in indoor intralogistics settings. With a nominal tractive force of 1800 N and a maximum load capacity of 5000 kg, the robot features a sturdy and small chassis. It can reach speeds of up to 8 km/h and has two rear support wheels and a precise driving wheel positioned in the middle of front [45].

Figure 8: This is the AMR with hardware enabled configuration discussed in KoLAMeRo Project [5]. Only the ZED 2i stereo camera sensory system is relevant for this particular project work.

A ZED 2i stereo camera from StereoLabs Inc is installed on the front of the AMR to provide visual perception for gesture-based interaction and navigation. Hand posture detection and 3D spatial localisation of movements are made easier by the ZED 2i's synchronised RGB and depth information. Thia enables features like accurate pointing to ground locations. In addition to vision, the AMR integrates standard industrial sensing for safe autonomous operation. The vehicle features two front and rear Sick OutdoorScan3 LiDAR sensors, each with a 275°

horizontal field-of-view, enabling strong obstacle detection and environmental mapping. The AMR is also equipped with an IMU and odometry modules for control and localization. This project mainly focuses on utilizing the ZED 2i sensory inputs for object detection and gesture recognition system.

ZED2i Stereo Camera

Stereo ZED 2i camera from Stereolabs is a rugged, industrial vision system for high-end 3D perception and AI-based applications. It is the central hardware in the gesture recognition system for synchronized RGB and depth capture-based precise 3D perception. With a broad 120° view angle with the two optical sensors separated by twelve centimeters, neural depth sensing of up to 20 meters, and built-in IMU and magnetometer, it offers accurate tracking of body and hand gestures. Its industrial-rated, IP66-protected enclosure is durable, while the body tracking SDK offers real-time gesture keypoint extraction. This makes the ZED 2i ideally suited for depth-aware gesture control and localization in dynamic intralogistics applications.

For best performance, the ZED 2i requires a high-end computing platform with a CUDA-enabled NVIDIA GPU (at least 4 GB of VRAM is highly recommended), like an NVIDIA Jetson or desktop-grade GPU system. The configuration used in this project is an RTX 4060 GPU, 16 GB RAM, and an Intel Core i7 14th Gen processor, which enables real-time inference and seamless integration with ROS 2. This hardware platform has the computational bandwidth necessary for depth processing, keypoint detection, and gesture classification in real-world intralogistics scenarios.

4.1.2 High-level System Architecture

The suggested system leverages an open multimodal gesture recognition pipeline that combines visual sensing, deep learning based gesture inference, and ROS 2 based robot control in an intralogistics settings. Figure 9 shows the designed system architecture and its information flow in detail.

The start of system utilizes a ZED 2i stereo camera, offering synchronized RGB and depth streams with onboard IMU and wide-angle optics. These data are first passed through a frame pre-processing module that performs color space conversion i.e from RGB to BGR and pulls out aligned depth maps. Pre-processed frames are then used to keypoint extraction via Google MediaPipe libraries pose and hand landmark model bundle, an open-source application oriented model that can return a holistic set of 2D keypoint landmarks for the upper body and hand of the user.

The primary logic behind switching is movement detection. This logic is employed to dynamically switch between static and dynamic gesture recognition models, which are called initially. This logic uses inter-frame pose keypoint differences to assess motion between the frames. If movement is detected, then pose vectors are appended to a temporal buffer until it reaches a sequence of 30 frames. These buffered sequences are then classified using a CNN-LSTM dynamic gesture model, trained to recognize temporally dependent gestures.

When no movement is observed in N frames (where typically N = 5), the pipeline would instead extract hand landmarks and feeds them into an MLP static gesture classification model that is tuned for one-shot detection like *"open palm"* etc. The models are trained both with normalized landmark coordinates and use confidence thresholds (greater than or equal to 0.75) for filtering

predictions.

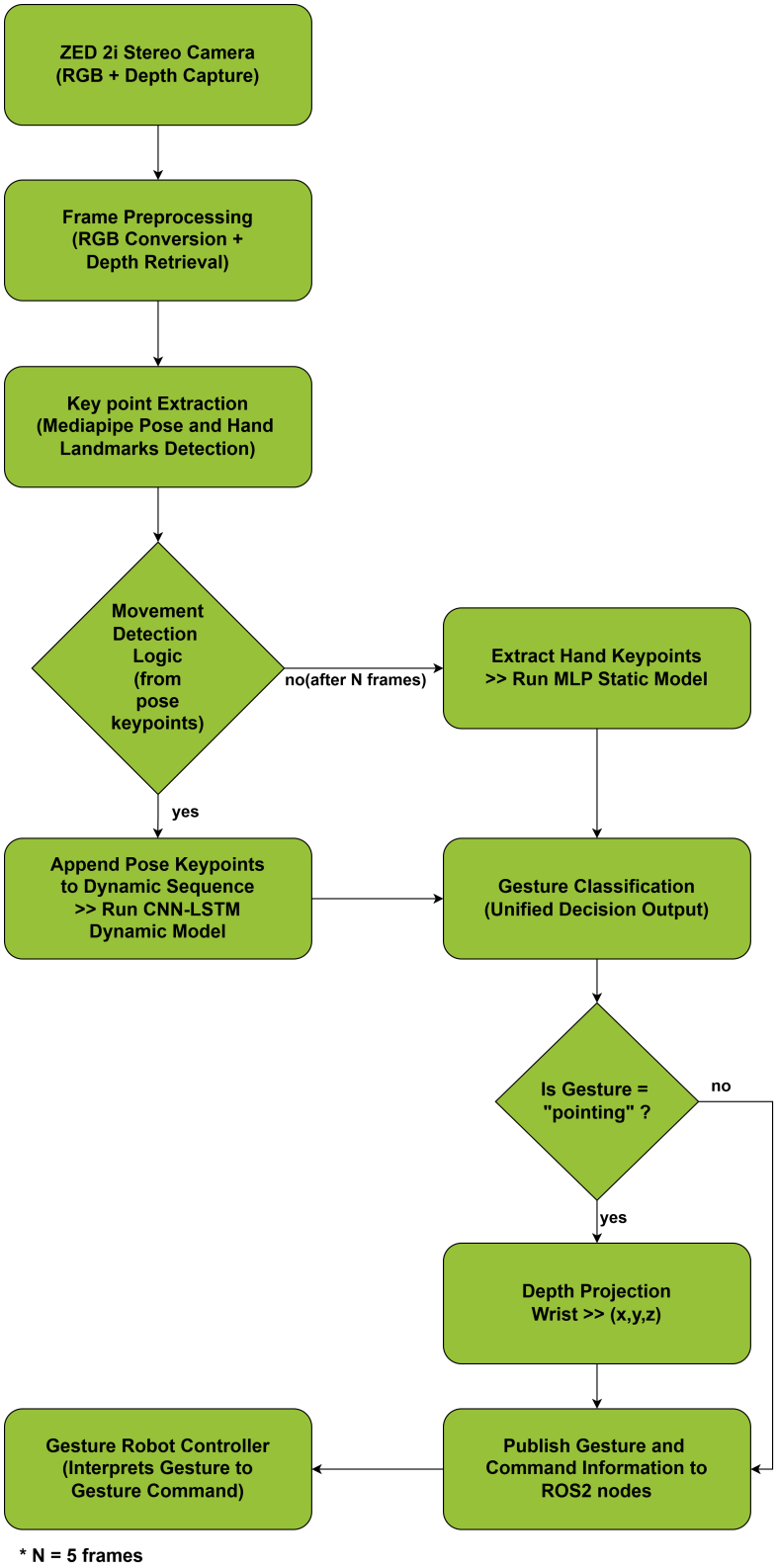


Figure 9: System architecture for gesture recognition. Keypoints are extracted via Mediapipe library and classified using static or dynamic models based on motion. Pointing gestures trigger depth projection and ROS 2 integration.

The resulting gesture prediction is input to a uniform gesture classification interface. If the gesture is *"pointing"*, the system projects the 3D wrist coordinate from the ZED 2i depth frame using the wrist landmark (left or right) with the higher visibility score. This coordinate is mapped to the robot workspace to enable location-aware interaction. All recognized gestures and depth data (if available) are published to ROS 2 topics i.e `/recognized gesture` (`std_msgs/String`) and `/pointing_coordinates` (`geometry_msgs/Point`). These are subscribed to by the gesture robot controller node, which operates as a finite state machine with states such as `STANDBY`, `WAITING`, `ACTIVE`, and `PAUSED`.

Each gesture triggers a predefined action. For example, *"waving"* toggles between `STANDBY` and `WAITING`, *"thumbs up"* signals successful task completion, *"closed fist"* triggers an emergency stop, *"open palm"* can pauses or resumes motion, *"v sign"* enters task learning mode, and *"pointing"* initiates forward movement until the target depth is reached. The controller converts these gestures into motion commands via `/cmd_vel` (`geometry_msgs/Twist`) or task instructions via `/robot_command`.

4.1.3 Classification Process Model

Figure 10 depicts the whole pipeline of the proposed gesture recognition system, encompassing camera input to the ultimate gesture categorisation. MediaPipe is used to analyse input frames (1) and extract 2D and 3D keypoints (2–3). To ensure location and scale invariance, the collected landmarks are turned into normalised feature representations using relative coordinates with respect to a reference point(wrist with index 0) and scaling. While temporal sequences of upper-body landmarks are employed for dynamic movements, single-frame hand landmarks are used for static motions. To generate the final gesture label (5), these characteristics are encoded (4) and categorised using specialised neural networks.

Feature Representation

The 2D hand keypoints that are extracted using MediaPipe's hand tracking model serve as the foundation for feature representation in this static gesture recognition pipeline. Twenty-one key landmarks that correspond to important finger joints and the wrist are used to represent each hand. The ZED 2i camera feed is used to first record these landmarks as absolute pixel coordinates, which are then converted into a more reliable representation.

The (x, y) locations of each landmark are extracted using the calc landmark list() function, and the representation is invariant to hand position and scale after pre-process landmark() centres the coordinates with respect to the wrist and normalises them by the maximum absolute value. By using this preprocessing, the gesture classification model is guaranteed to concentrate on the hand's relative spatial structure rather than its exact location within the image. The resulting flattened and normalized list of coordinates serves as a compact and meaningful feature vector for training the static gesture classification model.

In case of dynamic gesture, the system uses MediaPipe's pose landmark model to extract 13 upper-body keypoints for dynamic gestures, including the nose, eyes, ears, shoulders, elbows, wrists, and hips. Each frame's 39-dimensional feature vector is created by flattening the landmarks' 3D coordinates (x, y, and z). A CNN-LSTM-based dynamic gesture recognition model is trained using 30-frame sequences saved as.npy files, which record motion over time.

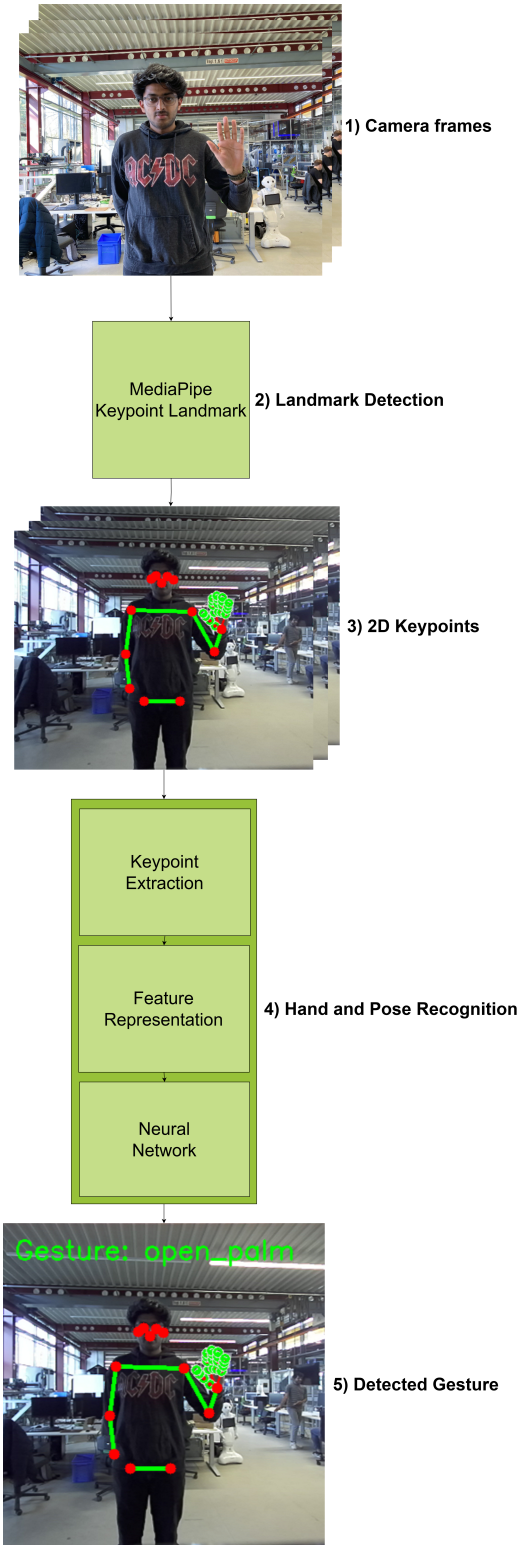


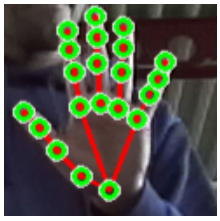
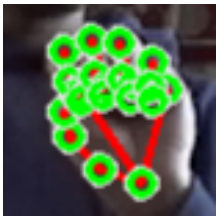
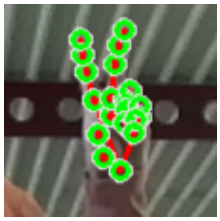
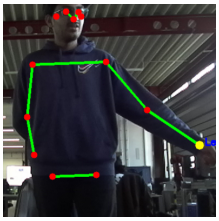
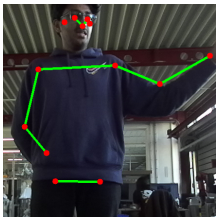
Figure 10: Figure demonstrates the process pipeline i.e from input frames that are processed for landmark detection, then keypoints are extracted and encoded, and finally gestures are classified via a neural network.

4.1.4 Gesture Recognition and State Model

Gesture Definition

The gesture recognition framework developed in this thesis incorporates both *static* and *dynamic* gestures, forming an intuitive and ergonomic interface. Six distinct gestures are implemented, and respective gesture commands are defined. They are *"open palm"*, *"closed fist"*, *"thumbs up"*, *"v sign"*, *"pointing"*, and *"waving"*. Static are defined as hand poses captured in a single frame, classified using a lightweight MLP model. They are primarily assigned to critical safety and control operations. On the other hand, dynamic gestures involve a sequence of movements captured over time and interpreted via a CNN–LSTM architecture. These are reserved for context-sensitive tasks such as user login via *"waving"* and robot navigation via *"pointing"* gestures. Following table 4 shows defined gesture that needs to be classified by the algorithm.

Table 4: Defined Gestures Used in the Recognition Framework

Gesture	Image	Gesture	Image	Gesture	Image
open palm		closed fist		thumbs up	
v sign		pointing		waving	

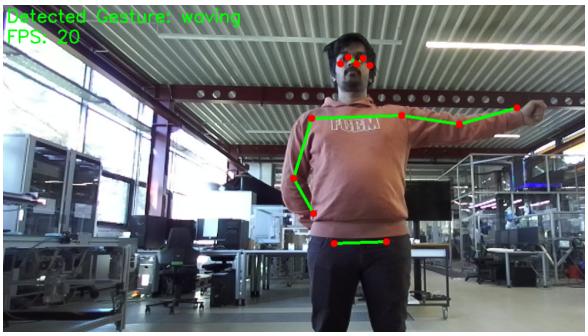
Functional Modes

- **Login Mode**: This mode handles user access control. It is accountable for managing access and initiating the interactive session between robot and user. A dynamic gesture, i.e., *"waving"* of the hand, serves as the authentication trigger for login and logout actions. The motion of waving is completed using both forearm and hand. Waving with hand is ignored, since the pose landmark detection will not get triggered. The gesture is identified by the model, and upon identification, enables the system to transition between `STANDBY` and `WAITING` state.

  The login gesture also serves as a contextual boundary for subsequent gesture commands, thus keeping control through gestures active only during sessions of legitimate users. The interaction window is clearly defined, i.e gestures are recognized only when the system is in the `WAITING` state, thus preventing accidental actions. After following a successful login or logout operation, the system gives clear feedback in the form of messages or status indications to the robot controller node, thereby acknowledging the state

change to the user. This strategy makes the process of entering or exiting the interaction loop natural. Figure 11 shows the start and end position of Login/Logout mode.

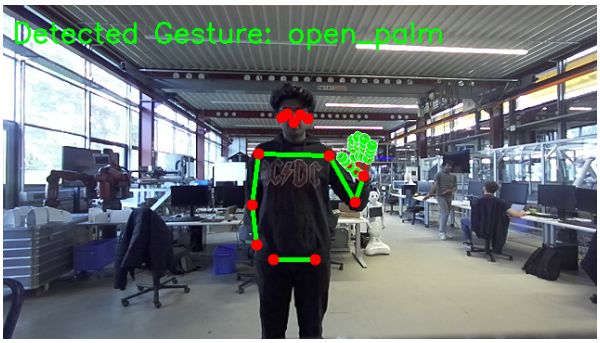
a) Start position



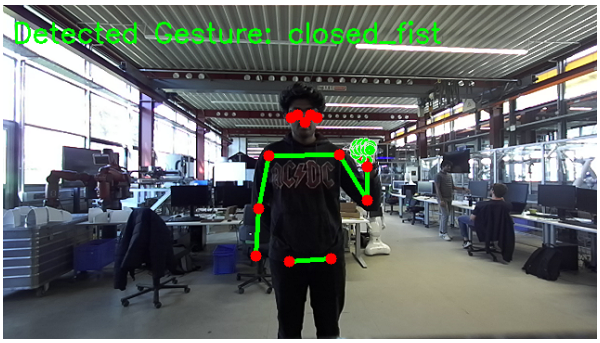
b) End position

Figure 11: Figure demonstrate sequences of the waving gesture from start and end positions.

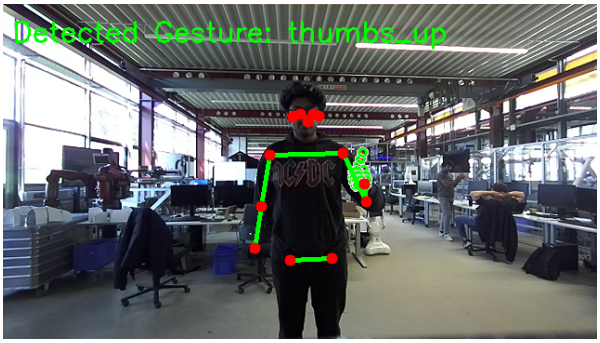
- **Maneuvering Mode**: Under this operation mode, the robot is set to take static gestures for real-time modulation of the task and thus enabling precise control of its operation mode during material handling tasks. In figure 12 a), the gesture *"open palm"* enables the user to switch between the `PAUSED` and `ACTIVE` states, which can pause or resume the ongoing operations respectively. Whenever the user presents a *"closed fist"* like in figure 12 b), the robot performs a hard stop and switches to the `STOPPED` state, halting any movement instantly. This is extremely useful when the user get interfered suddenly with the robot, further avoiding an accident.



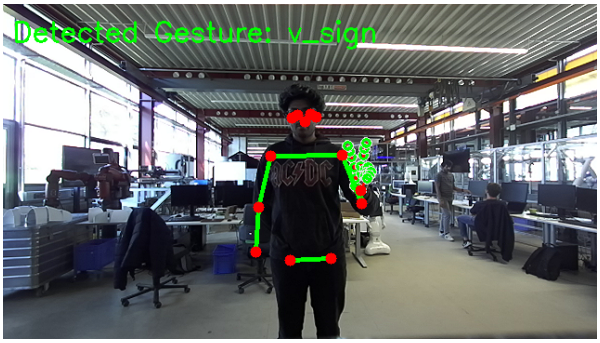
a) open palm



b) closed fist



c) thumbs up



d) v sign

Figure 12: Different types of static hand gestures discussed in this project thesis.

In figure 12 c), *"thumbs up"* gesture is utilized to signal successful completion of a task, commanding the system to revert to the `WAITING` state in preparation for the next command. A *"v sign"* gesture in figure 12 d) serves as an event trigger to enter the learning mode, allowing the robot to store or recall the current task context for adaptive behavior. This can be in future, integrated with continual learning algorithms. These movements define a natural and comprehensive set of commands for users to dynamically and securely modulate robot activity. Figure 12 represents the various static gestures.

- **Pointing Mode**: This interactive gesture mode allows spatial interaction with the robot. Once *"pointing"* gesture is detected, 3D wrist keypoint landmark coordinates are captured by the ZED 2i stereo camera using synchronized RGB and depth data. Using pose keypoint and hand keypoint, the system identifies whether wrist (left or right hand) is more visible and obtains the appropriate 3D coordinates from the 2D keypoints and depth value. Figure 13 demonstrates the start and end positions of *"pointing"* gesture.

  The robot motion is initiated by the robot until desired depth value Z is reached. According to pose landmarks from mediapipe library, the index value for left and right hand is 15 and 16 respectively. The landmark coordinates are discussed more in the upcoming subsections. This allows intuitive direction without contact. For example, a point-to-move scenarios can be supported, such as moving the robot to a new location for material handling.

a) Start position *b) End position*

Figure 13: Figure shows the start and end sequence of pointing gesture with depth values.

- **Following Mode**: This mode is built as a separate module using the ZED cameras inbuild feature models. In this mode, rather than gesture classification model, an object detection task is executed. The goal is to provide increased safety and intuitive usage in close human-robot contact settings. It does not use gesture model and instead rely completely on real-time object detection model using the ZED 2i stereo camera. Inspired by the FiFi robot's person-following idea [1], the robot tracks and follows a user autonomously while maintaining a safe and consistent distance with 3D positional data.

  Using the ROS 2 object detection module from Stereolabs, different models can be configured. The system here utilizes ZED SDK's object detection model `"MULTI CLASS BOX MEDIUM"` and positioning tracking capabilities to enable continuous tracking. The model employs multi-class object detector, which has the capability to detect several kinds of objects such as `PERSON` that is needed in our scenario [46]. When the tracked users step out of sight or get occluded, then the robot makes a safe stop and awaits

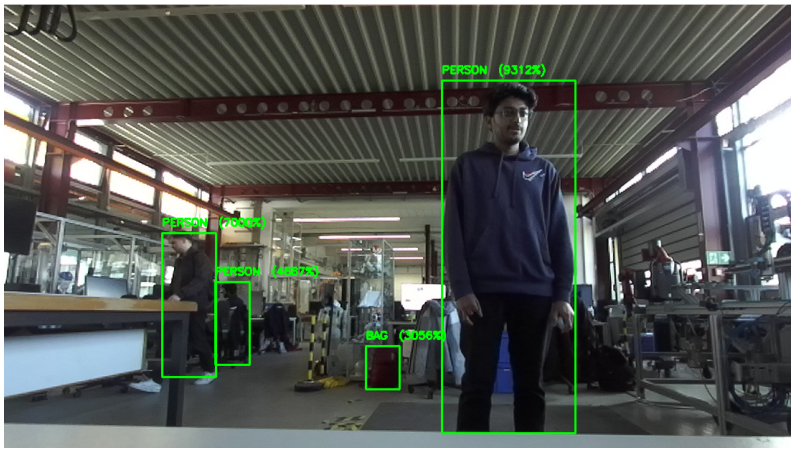


Figure 14: Results from object detection module showing the bounding box.

re-acquisition. Figure 14 shows some random images from the experiment conducted. In this, the model clearly identifies different objects with a bounding box and accuracy percentage.

Gesture State Mapping

The designed recognition system initializes in `STANDBY`, awaiting a *"waving"* gesture to log in and enter an interactive mode with user. After logging in, the user can take control by demonstrating static hand gestures ie. an open palm for pauses or resumes tasks, a closed fist immediately terminates execution, and a thumbs up signals task completion. Each gesture is uniquely assigned to a specific robot command, allowing for quick recognition and little ambiguity.

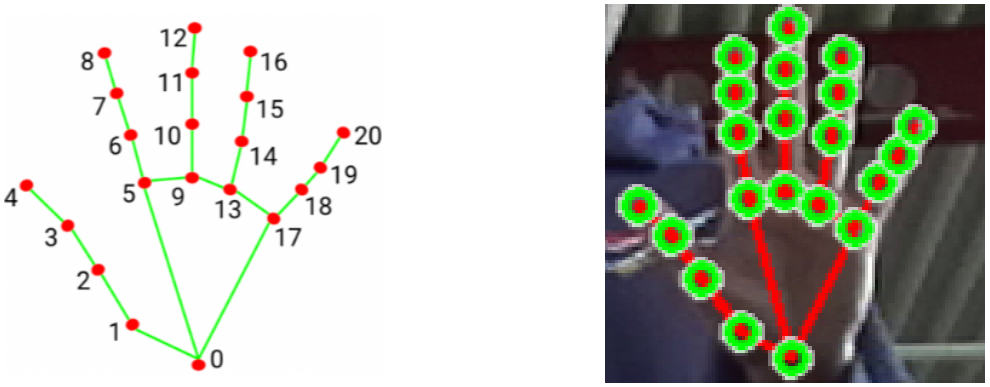
Table 5: Simplified Mapping of Gesture Types to Commands

Functional Mode	Gesture	Type	Gesture Command
Login Mode	*waving*	Dynamic	User login
Manoeuvring Mode	*open_palm*	Static	Pause process
	open_palm	Static	Resume process
	closed_fist	Static	Stop process
	thumbs_up	Static	Task completed
Pointing Mode	*pointing_A*	Dynamic	Move to point A
Following Mode	object detection	–	Follow user
Logout Mode	*waving*	Dynamic	User logout

For spatial commands, the user performs a pointing gesture, and the robot navigate to the indicated target location. In contrast, no gesture input is needed for following mode detection of a person. It is planned to initiates automatic user following task, with the robot tracking user at a safe distance. The interaction loop is logged out, and the system returned to `STANDBY`, by a repeated *"waving"* gesture. This ensures the AMRs are free after their assist to humans.

Keypoint Extraction and Preprocessing

Pose and hand landmark detection is a critical function in facilitating smooth HRI using gestures. The MediaPipe library [43] which was developed by Google offers a fast and stable solution for undertaking this function. As a cross-platform, open-source tool, it utilizes ML techniques on color images to effectively detect and track major body and hand points in real time using a ZED2i Camera. The system supports both static and dynamic gestures, each requiring distinct types of spatial-temporal input preprocessing. Figure 15 and 16 represents the hand and pose keypoint landmarks used in our experimental sections.



a) MediaPipe hand keypoint [43] *b) A sample image with keypoint landmarks*

Figure 15: Figure shows the 21 hand landmarks keypoint from Google MediaPipe documentations and sample image with keypoints landmarks respectively.

For static hand gestures, which is already defined in previous sub-section, the MediaPipe hand landmark is used to identify 21 hand landmarks in each frame. Only the 2D coordinates (x, y) of each landmark are used in our case. The landmark coordinates are first transformed into relative coordinate values with regard to the wrist joint (labeled as index 0) in order to increase resilience to hand translation, scale, and rotation. In second step, normalization is performed by dividing by the highest absolute coordinate value. This produces a normalized and compact feature vector of 42 values (21 landmarks with a factor of 2), which is stored as a row within a CSV file [44] for model training purposes.

For dynamic gesture as in figure 16, the system utilizes pose landmarker's 33 body keypoints to derive 13 upper-body landmarks, which are relevant for pose tracking of defined dynamic gesture. To follow 13 upper-body landmarks over a sequence of frames, the chosen landmarks includes nose, eyes, ears, shoulders, elbows, wrists, and hips, offering a good balance between motion and gesture posture.

The 3D coordinates (x, y, z) of these keypoints are taken from every frame and flattened into a 39-dimensional feature vector (13 landmarks $\times$ 3). Out of this, sequences of 30 frames are collected and stored in the form of `.npy` files for temporal modeling. These sequences record

a) MediaPipe pose keypoint [43] *b) A sample pose with keypoint landmarks*

Figure 16: A 33 pose landmarks keypoint from Google MediaPipe is shown. In this project, only 13 upper-body landmarks are used for pose tracking.

motion dynamics and are then used to train a CNN-LSTM based dynamic gesture recognition system.

4.2 Setup of Model Training

This section details the setup for model training of the proposed gesture recognition system. It discusses the training methods employed for static and dynamic gesture models. The detailed classification pipeline employed during the inference stage is demonstrated at the end. Results are derived from a dataset captured using the ZED 2i stereo camera that utilizes left camera for RGB based images to get two-dimensional keypoint features and depth information to ensure accurate recognition.

4.2.1 Static Model Training

The static gesture classification model was compiled using a MLP model consisting of fully connected layers. The input layer received 42 normalized features derived from 2D hand landmark coordinates. The network architecture included two hidden layers with 128 and 64 units, respectively, each utilizing ReLU activation functions to introduce non-linearity. The output layer comprised four neurons corresponding to the gesture classes, activated using a softmax function to generate class probabilities. Class labels 0–3 correspond to *"closed fist"*, *"open palm"*, *"thumbs up"*, and *"v sign"*, respectively.

The model was trained using the categorical cross-entropy loss function, which is suitable for one-hot encoded labels in multi-class classification tasks. Optimization was performed using the Adam optimizer with a learning rate of 0.001. Training was conducted over 30 epochs with a batch size of 32, employing mini-batch gradient descent. A validation split of 20% was applied during training to evaluate the model's ability to generalize to previously unseen data. Classification accuracy was used as the primary evaluation metric.

Algorithm 1 Static Gesture Model Training using MLP

1: Load 2D keypoint dataset and gesture labels from $.CSV$ file.
2: Normalize keypoints and one-hot encode labels.
3: Split data into training and test sets.
4: Define MLP with input, hidden, and output layers.
5: Compile model with cross-entropy loss and Adam optimizer.
6: Train model for N epochs with batch size B and validation split.
7: Evaluate performance on test set.
8: Save trained model for inference.

4.2.2 Dynamic Model Training

The dynamic gesture classifier uses a CNN-LSTM network learned from pose keypoint sequences. The input sequences are 30 frames long, with each frame consisting of 13 upper-body landmarks (nose, eyes, shoulders, elbows, wrists, hips) as 3D coordinates. The resulting has an input shape 30×39. The network structure is temporal 1D convolutional layers for short-term motion encoding, followed by LSTM layers for long-range temporal dependence modeling. A final dense layer with softmax activation outputs class probabilities.

Algorithm 2 Dynamic Gesture Model Training using CNN-LSTM

1: Load (30×39) gesture sequences from `.npy` files.
2: Normalize 3D keypoints and one-hot encode labels.
3: Split data into training and test sets.
4: Define CNN-LSTM model with 1D conv and LSTM layers.
5: Compile with cross-entropy loss and Adam optimizer.
6: Train for N epochs with batch size B and validation split.
7: Evaluate on test set and save model.

The CNN-LSTM model underwent training for 40 epochs with a batch size of 32, utilising the Adam optimiser and categorical cross-entropy loss function. A 20% validation split was utilised to track generalisation performance during training.

4.3 Integration with ROS 2

ROS 2 has accelerated the development and integration of autonomous mobile robotic systems by providing a real-time, flexible, and scalable middleware architecture. Its built on the Data Distribution Service (DDS), and enables deterministic communication, increased fault tolerance, and decentralised message exchange, which are crucial characteristics for safety-critical navigation in dynamic situations [42].

This integration method takes use of ROS 2 robust middleware and scalability, as proven in previous research on autonomous mobile robots and logistics platforms [47]. The ROS 2 system allows for safe and flexible navigation in complicated situations, as well as gesture-based instructions performed with spatial and temporal coherence, thanks to the system's intuitive, real-time human-robot interface.

The gesture robotic control system on gesture is developed as a distributed ROS 2 system with three core nodes. They are `zed_camera.launch.py`, `gesture_node.py`, and `gesture_robot_controller.py`. The system is built with the ROS 2 modest middle-ware on Ubuntu 22.04 LTS, which uses DDS-based communication for node discovery and low-latency data transport. This architecture uses ROS 2's publish-subscribe paradigm to ensure asynchronous and scalable communication across the perception, decision-making, and actuation layers.

ZED ROS 2 Wrapper

The ZED ROS 2 wrapper, created by Stereolabs, is the official interaction layer between ZED stereo cameras and the ROS 2 environment. It offers access to a wide range of data streams, such as rectified colour and greyscale images, depth maps, 3D point clouds, IMU measurements, positional tracking, and human skeletal identification [48]. It makes available rectified color images and depth maps used for MediaPipe-based keypoint extraction and spatial reasoning. The wrapper enables real-time camera streaming, making it an important link between the hardware sensor and the ROS 2 software stack. Figure 17 shows the zed 2i version used in this project work.

Figure 17: ZED 2i stereo camera enabling 3D perception through human-like stereo vision and depth triangulation. It provides spatial and motion-aware 3D understanding of the environment.

Object Detection Using ZED

With the Stereolabs SDK, the ZED 2i stereo camera enables real-time 3D object detection using integrated deep learning models. After camera initialization, object detection is configured with tracking and segmentation enabled, using the `MULTI_CLASS_BOX_MEDIUM` model for a balance between speed and accuracy. Detected objects are retrieved via `zed.retrieve_objects()`, providing 3D spatial data, class labels, confidence scores, and 2D bounding boxes. OpenCV overlays these on the live camera feed with bounding boxes and labels, while low-confidence detections are filtered using a threshold. This setup offers efficient 3D object detection suitable for robotics and real-time applications. In situations involving multiple sensory input, this integration improves environmental perception, facilitating intelligent interaction and decision-making.

Gesture Recognition Node

The `gesture_node.py` node utilizes the ZED 2i stereo camera's video and depth inputs to classify static and dynamic hand and upper-body gestures in real time. Static gestures (e.g., *thumbs_up*, *open_palm*) are recognized from single-frame 2D

hand landmarks, while dynamic gestures (e.g., *waving*, *pointing*) are inferred from temporal sequences of 3D posture keypoints. The node processes synchronized image and depth data received from `/zed/zed_node/left/image_rect_color` and `/zed/zed_node/depth/depth_registered`. Figure 18 illustrates topics and nodes as discussed.

Gesture Robot Controller Node

The `gesture_robot_controller.py` node acts as the primary decision-making module. It responds to both `/recognized_gesture` and `/pointing_coordinates` topics and switches between internal robot states (e.g., `STANDBY`, `ACTIVE`, etc.) based on gesture input. For example, a waving gesture triggers a login/logout toggle, while a pointing gesture commands the robot to navigate to the specified 3D location. The controller publishes velocity commands to `/cmd_vel` using `geometry_msgs/Twist`, and system feedback is sent via `/robot_command` for debugging and visualization.

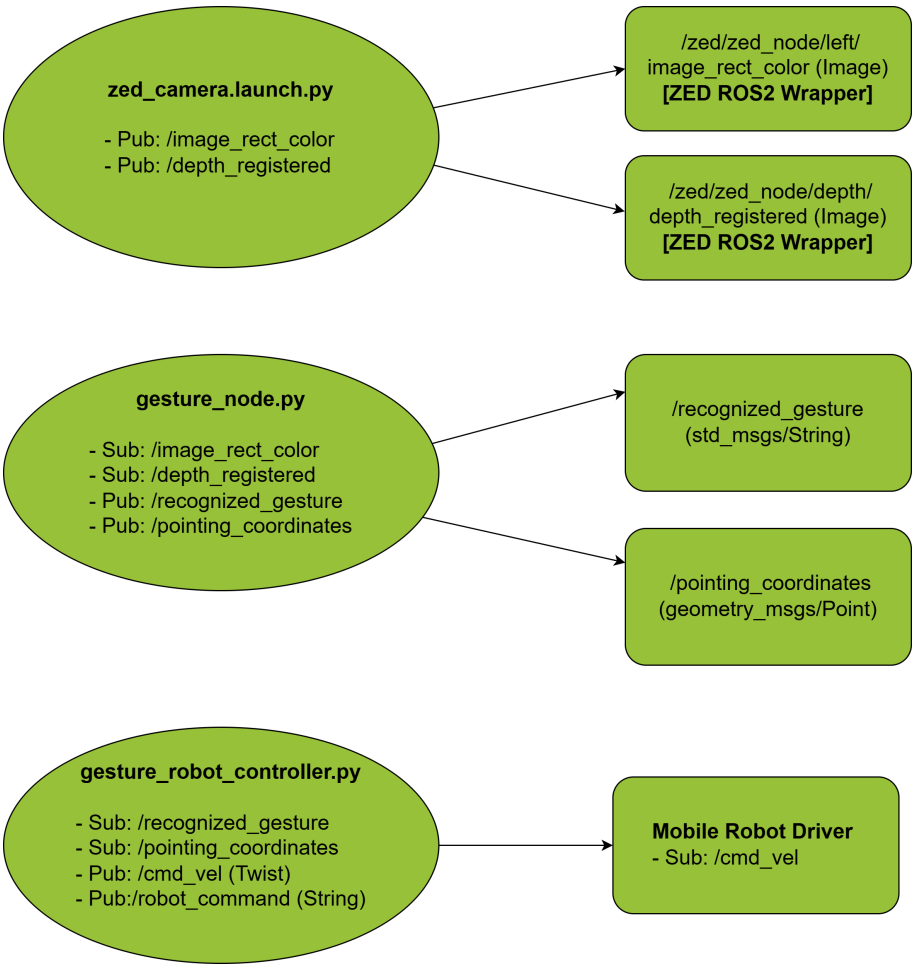


Figure 18: ROS 2 node architecture linking the ZED 2i camera, gesture recognition, and robot controller through topic-based communication.

Gesture-Controlled State Transition

The gesture recognition system recognizes both static and dynamic motions in real time. Six distinct gestures are implemented, each mapped to a specific robotic behavior via a finite state

machine in the `gesture_robot_controller.py`.

- ***closed_fist***: Triggers an *emergency stop* by transitioning the system into the `STOPPED` state and halting all motion.
- ***open_palm***: A static gesture used to either *pause* or *resume* ongoing robot actions. If the system is in the `ACTIVE` state, it pauses and stores the velocity; if in `PAUSED`, it resumes the previous velocity and returns to `ACTIVE`.
- ***thumbs_up***: A static gesture signaling task *completion*. This transitions the system to the `WAITING` state, acknowledging the end of an activity.
- ***v_sign***: A static gesture used to initiate *task learning mode*, updating internal memory structures and switching to the `LEARNING` state.
- ***waving***: It functions as a toggle for user *login/logout*, switching the robot between `STANDBY` and `WAITING`. This gesture is critical in defining the interaction window and is highlighted in the system interaction diagram.
- ***pointing***: A dynamic gesture that initiates spatial interaction. Upon recognition, the system extracts the 3D wrist coordinate using ZED 2i depth data. This gesture activates forward navigation (via `/cmd_vel`) until the target depth is reached, returning to `WAITING` state thereafter.

All gestures are published to `/recognized_gesture` (`std_msgs/String`), with `pointing` also publishing the 3D coordinate to `/pointing_coordinates` (`geometry_msgs/Point`). The finite state machine of the robot controller interprets these gestures to determine transitions across modes such as `STANDBY`, `WAITING`, `ACTIVE`, and `LEARNING`, forming a tight gesture-to-behavior loop for seamless human–robot interaction.

5 Results and Discussion

This chapter presents the evaluation outcomes of the proposed gesture recognition system. It includes quantitative and qualitative analysis of model performance under real-world conditions. The results demonstrate the effectiveness of the integrated MLP and CNN-LSTM models in recognizing a range of hand and upper-body gestures using ZED 2i camera input.

5.1 Model Evaluation with Hold-out Test

The static model was then tested using a hold-out test set to estimate generalization performance after training. The final test accuracy was around 96.7%, which indicates very good classification capability. The confusion matrix indicates that the majority of samples from each of the four gesture classes were well-classified by the model. Notably, the *"thumbs up"* and *"v sign"* gestures were nearly perfectly classified with hardly any misclassifications.

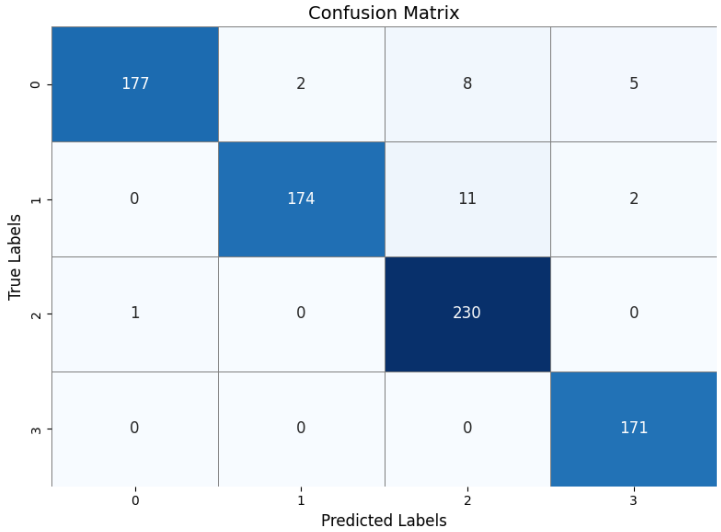


Figure 19: Confusion matrix showing the classification results of the MLP-based static gesture recognition model on the test set. Class labels 0–3 correspond to closed_fist, open_palm, thumbs_up, and v_sign, respectively.

However, figure 19 shows there was some class confusion, particularly between *"open palm"* and *"thumbs up"*, and *"closed fist"* and *"thumbs up"*. These are likely caused by the geometric hand structure resemblance for these gestures when projected to 2D space. Despite this, the inter-class confusion was nonetheless low, and precision, recall, and F1-scores of all classes were consistently high. The outcomes validate the effectiveness of MLP model for recognizing static gestures through the utilization of normalized keypoint data obtained from ZED 2i camera.

Following dynamic training, test set evaluation revealed class-wise performance using a normalised confusion matrix: waving motions were properly detected with 100% accuracy, pointing movements had 67% accuracy, and the remaining 33% were misclassified as waving. The algorithm for model training is shown below. Figure 20 shows the confusion matrix associated with the training.

The misclassification of pointing shows that the spatial-temporal characteristics retrieved from the 13 upper-body landmarks may not offer adequate separation from waving, especially in overlapping motion patterns of the arms and wrists. Because both motions may entail extended arm trajectories, particularly in comparable directions, the CNN-LSTM model may fail to detect tiny posture differences or temporal signals specific to pointing. Furthermore, pointing movements are often more motionless at the end, which may be under-represented in temporal characteristics recorded by convolutional filters. The results highlights the strength of CNN-LSTM model in modeling expressive motion sequences on less dataset, with opportunities for enhancement through improved gesture-specific feature encoding.

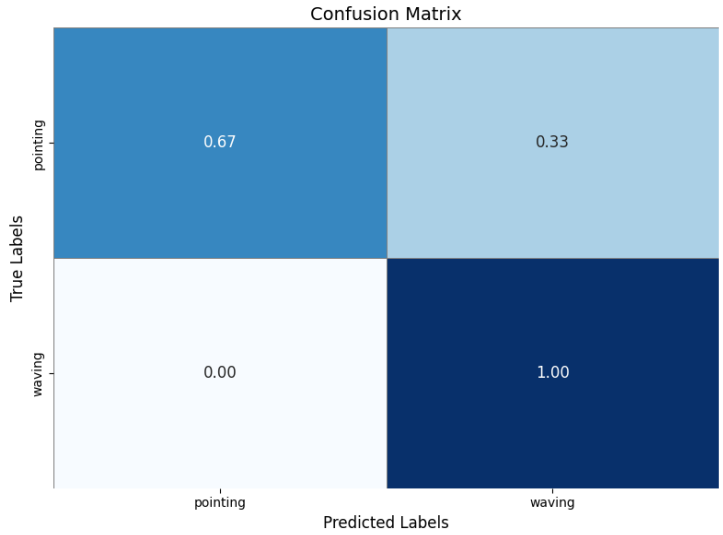


Figure 20: Normalized confusion matrix for dynamic gesture classification. The CNN-LSTM model achieved perfect accuracy on waving, with partial confusion observed for pointing.

5.2 Model Evaluation

The assessment of suggested gesture recognition system follows mixed quantitative and qualitative measures, as suggested by the pre-established research methods outlined by Bortz and Döring [49]. Quantitative assessment looks at quantifiable results such as classification accuracy and confusion matrices that reflect the effectiveness of system in categorizing six gestures across multiple subjects. Other model performance was assessed using accuracy, precision, recall, and F1-score. This supports the focus on using statistical measures to evaluate system performance in controlled experimental settings.

In addition to calculating overall accuracy, the system was assessed using precision, recall, and F1-score measures to enable a more detailed overall performance analysis. Precision calculates the accuracy of positive predictions, and recall measures the capability of model in finding all relevant instances. The F1-score, being the combination of the two, offers balanced performance across several classes. In the static model, gestures such as *"thumbs up"* and *"v sign"* recorded high F1-scores, reflecting high class separability. The dynamic model demonstrated superior performance with *"waving"* having better recall than *"pointing"*, which experienced infrequent misclassifications owing to faint motion trends. These performance re-

sults confirm the stability of the system over various gesture classes. Below table 6 and 7 shows the classification reports from static and dynamic model training respectively.

Table 6: Classification Report for Static Gesture Recognition Model

Class	Precision	Recall	F1-score	Support
Closed Fist (0)	0.99	0.92	0.96	192
Open Palm (1)	0.99	0.93	0.96	187
Thumbs Up (2)	0.92	1.00	0.96	231
V Sign (3)	0.96	1.00	0.98	171
Accuracy			0.96	781
Macro Average	0.97	0.96	0.96	781
Weighted Average	0.96	0.96	0.96	781

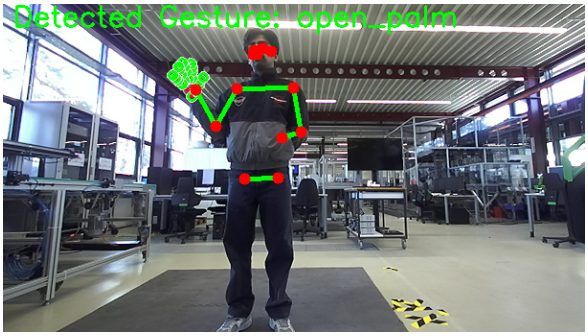
Table 7: Classification Report for Dynamic Gesture Recognition Model

Class	Precision	Recall	F1-score	Support
Pointing	1.00	0.67	0.80	3
Waving	0.75	1.00	0.86	3
Accuracy			0.83	6
Macro Average	0.88	0.83	0.83	6
Weighted Average	0.88	0.83	0.83	6

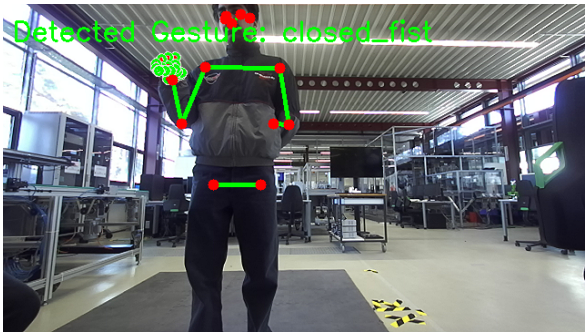
Qualitative Evaluation

The system was qualitatively evaluated at a constant distance of 1 meter from the ZED 2i camera, enabling depth perception and keypoint extraction consistency. Gesture detection validation was obtained using real-time overlays, showing correct landmark tracking with negligible latency. Furthermore, the system proved to be robust over participants with varying physical attributes, thereby validating its user independence and practical feasibility. Along with the well-specified test criteria, this two-mode testing guarantees both the technical validity and the usability of the suggested gesture-based control interface.

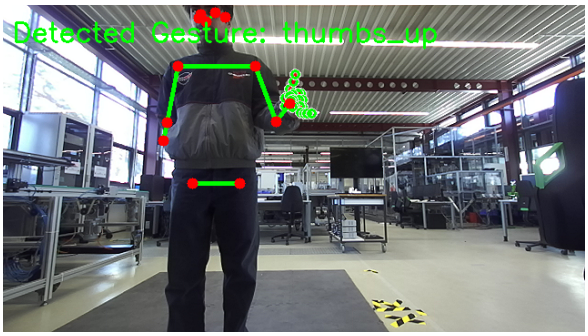
To evaluate the performance and robustness of the proposed gesture recognition system, benchmarking was conducted on first selected person performing all six defined gestures: *"closed fist"*, *"open palm"*, *"thumbs up"*, *"v sign"*, *"pointing"*, and *"waving"*. The results, illustrated in the figure 21, confirm the system's ability to generalize across individuals with varying physical and manual traits.



a) open palm



b) closed fist

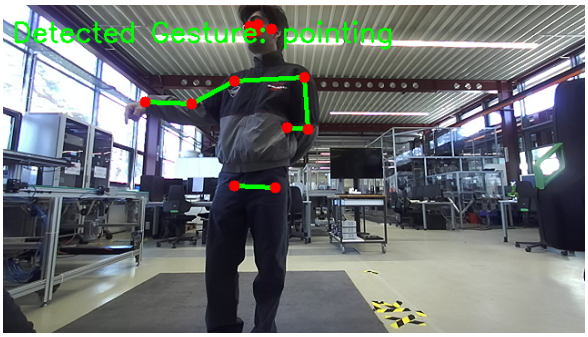


c) thumbs up

d) v sign

Figure 21: Different types of static hand gestures performed by person 1.

In figure 22, performed by person 1 demonstrate accurate identification of the pointing gesture, for left and right, by the suggested CNN-LSTM-based dynamic gesture recognition pipeline. The detection is temporal modeling of 3D upper-body keypoint, specifically emphasizing the arm extension and direction alignment of the wrist, elbow, and shoulder. The system effectively differentiates between left and right pointing under spatial arrangement, even with arm orientation and body position changes. The red and green keypoints show that MediaPipe landmark detection has been strong enough to provide steady feature extraction for temporal classification. These findings qualitatively confirm the model's generalization of dynamic motion cues and maintain semantic gesture integrity in real-world scenes.



a) Pointing gesture using left hand

b) Pointing gesture using right hand

Figure 22: Start and end sequence of pointing gesture by person 1 with tracked landmarks and detected gesture.

The static gesture demonstration by person 2 is carried out. The second person showed dis-

tinct recognition of all four static gestures. Each gesture is at a fixed distance of around 1 meter from the ZED 2i camera. The hand landmarks detected are correctly overlaid and gesture classification is clearly visible with stable frame rates of about 10–11 FPS. This indicates robust model generalization despite different subjects in different environmental light conditions. Technically, the system effectively extracted 2D keypoints with MediaPipe, normalized the landmark coordinates to the wrist, and passed them through a trained MLP model for classification. Despite slight pose differences and articulation variations between hands, the model was able to accurately predict the right labels, demonstrating its resilience for real-world static gesture recognition.

a) open palm

b) closed fist

c) thumbs up

d) v sign

Figure 23: Different types of static hand gestures performed by person 2.

Technically, the system effectively extracted 2D keypoints with MediaPipe, normalized the landmark coordinates to the wrist, and passed them through a trained MLP model for classification. Despite slight pose differences and articulation variations between hands, the model was able to accurately predict the right labels. While the system successfully recognizes single-hand gestures, it is currently limited to one-hand input due to the `max_num_hands=1` setting in MediaPipe. Consequently, bimanual or simultaneous gestures are not detected, potentially reducing interaction richness. Extending support for dual-hand recognition would increase gesture diversity but would require modifications to both the model architecture and the dataset.

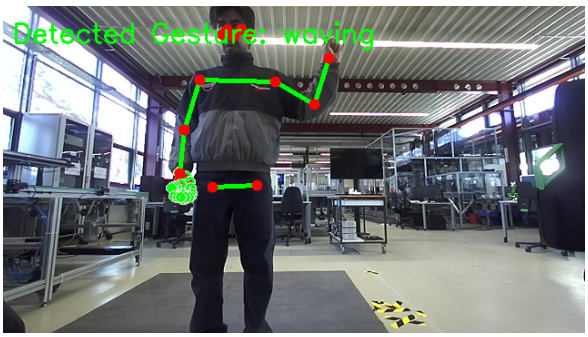
The inference mechanism self-adapts in real time by observing upper-body motion, enabling a transition between static and dynamic gesture models. This kind of feature enables the system to qualitatively differentiate between gesture types by detecting the presence or absence of motion, mimicking human-like perception. Figure 23 shows the static result corresponding to person 2.

As an example, dynamic gestures such as pointing are triggered only after motion surpasses some threshold, increasing resilience to noise. The system overlays landmark visualizations and gesture labels on the RGB feed to enable qualitative confirmation of prediction accuracy and spatial coherence. Pointing depth estimation adds additional improvements to the interpret-ability of 3D interactions in cluttered indoor settings.

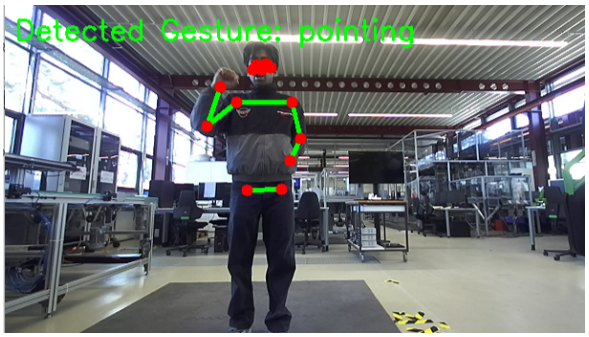
Misclassifications

Some gestures evaluated from person 1 shows misclassifications due to various reasons. Based on the observations, the misclassifications in dynamic gesture recognition can be explained by a training data limitation. That is, the dynamic gesture recognition model was trained only on gestures executed by the left hand, with just 30 sequences for each category i.e. *"waving"* and *"pointing"*. Thus, the model has learned to associate these gestural patterns based on one spatial orientation and alignment, making it sensitive to variations that fall outside the learned distribution.

In the given first example i.e figure 24 a), the model misclassified "waving" from static images, likely because there were inadequate motion cues or temporal information. In the second classification from figure 24 b), half arm lift was incorrectly classified as *"pointing"*, even though it was *"waving"*. This shows that the CNN-LSTM model is struggling to distinguish similar upper-body poses with no clear motion trajectories.



a) Misclassification 1



b) Misclassification 2

Figure 24: Examples of dynamic gesture misclassifications

When gestures are performed using the right hand or vary in lighting, posture, or location, the model will not generalize and may make incorrect predictions or class confusion between gestures. This bias shows how important dataset diversity is for temporal models like CNN-LSTM. To mitigate these issues, and also improve the model's robustness, training data must be augmented to include left and right hand variations, several individuals, and additional sequences per gesture. For that data augmentation methods, such as horizontal flipping of gesture sequences, might also simulate artificially balanced handedness without the requirement for significant additional data acquisition.

The system recognises single-hand motions, but is presently restricted to one-hand input owing to the `max_num_hands=1` parameter in MediaPipe. As a result, bimanual or simultaneous movements are not identified, which may reduce interaction richness. Extending support for dual-hand recognition will broaden gesture diversity, but it would need changes to both the model architecture and dataset.

6 Conclusion

This chapter concludes the thesis by summarizing the main contributions and findings from the experimental analysis and model benchmarks. It also addresses current limitations and potential extension of the proposed model frameworks to explore new boundaries in deep learning applications, thereby contributing to cutting edge developments in the field of Intralogistics.

6.1 Summary

In this thesis, a deep learning-based gesture detection system was developed to enable natural human-robot interaction in intralogistics contexts. Using a hybrid MLP and CNN-LSTM architecture, the system recognizes both static and dynamic hand gestures and is optimized for real-time operation with the ZED 2i stereo camera. The dynamic gestures like *"pointing"* and *"waving"* were detected from sequences of upper-body 3D posture data over time using a CNN-LSTM model. The static gestures were identified using MediaPipe 2D hand keypoint and annotated using a trained MLP.

The system concept was built using ROS 2, with real-time gesture mode switching based on motion movements. Inference was assessed subjectively and quantitatively by classification reports and live scenario testing with two separate users at a fixed distance. The final test accuracy for static gesture model was around 96.7%, and for dynamic gesture model with waving motions detected with 100% accuracy, and pointing movements had 67% accuracy. The results demonstrated good accuracy for both gesture categories, with just a few misclassifications for dynamic gestures, mostly due to inadequate movement patterns or the model's sensitivity to posture changes. The depth computation of the pointing motion also provided actionable robot commands through coordinate projection.

Although the model successfully distinguished gestures from diverse users, limitations include single-hand tracking, lack of support for bimanual gestures, and restricted training data for some dynamic classes. Nonetheless, the system proved robust, extensible, and suitable for real-time applications in robotic systems requiring non-contact human input.

6.2 Future Scope

For further developments, several new directions still remains unexplored. To summaries, while the proposed gesture recognition technology displays strong real-time inference, it does have certain drawbacks. The MediaPipe setting limits the system to single-hand recognition, excluding bimanual gestures and more complex interaction patterns. The dynamic gesture model was trained on a short dataset (.npy format) with few modifications in perspective, hand position, and user variety. Expanding the training dataset with more sequences and many users executing gestures from various perspectives might improve generalisability. Although real-time performance is good, it might be improved further using TensorRT or ONNX acceleration pipelines.

Future research might include dual-hand tracking and contextual merging of full-body pose and hand landmarks to improve expressive gesture interpretation. With the ZED 2i's support

for 3D spatial input, future designs may incorporate spatio-temporal graph convolutional networks (ST-GCNs), which better describe joint dependencies for dynamic gestures. Similarly, transformer-based models like TimeSformer and Video Swin Transformer, which are gaining prominence in action identification, may be used to classify fine-grained gestures while preserving temporal awareness between frames.

Using advanced machine learning approaches such as Reinforcement Learning (RL) and Transformer-based models may considerably improve the flexibility and accuracy of gesture detection systems. RL allows robots to learn optimal responses via trial and error, making it ideal for dynamic contexts where predetermined behaviors may not be sufficient. For example in [50] , RL has been used to categorise EMG and IMU data for hand gesture detection, proving its ability to adapt to user-specific variances. Furthermore, RL frameworks have been used for surgical gesture detection, demonstrating their use in complicated, real-time decision-making circumstances. While CNN-LSTM methods are good at dynamic sequence based gestures, RL can be applied for adaptive interaction in unknown environments.

Transformer-based models, which are noted for their attention processes and ability to capture long-range relationships, have demonstrated promise in gesture identification tasks. Hand gesture categorization uses Vision Transformers (ViT) to successfully model spatial and temporal characteristics [51]. Furthermore, hybrid architectures integrating Transformers with other neural network models have achieved great accuracy in recognizing dynamic hand motions from surface EMG signals, demonstrating their robustness in handling multimodal data. These developments indicate that combining RL with Transformer-based techniques can result in more sensitive and accurate gesture recognition systems, improving human-robot interaction in a variety of applications [52].

Bibliography

[1] A. Trenkle, Z. Seibold, T. Stoll, and K. Furmans, “Fifi – steuerung eines ftf durch gesten- und personenerkennung,” *Logistics Journal*, vol. 2013, no. 10, 2013.

[2] A. K. Grover and M. H. Ashraf, “Autonomous mobile robots for warehousing and distribution industry: A step toward intralogistics 4.0,” in *Digitization in Supply Chain Management* (S. Yeniyurt and S. Carnovale, eds.), vol. 02 of *Trends, Challenges and Solutions in Contemporary Supply Chain Management*, pp. 153–183, WORLD SCIENTIFIC, 2024.

[3] J. Fottner, D. Clauer, F. Hormes, M. Freitag, T. Beinke, L. Overmeyer, S. N. Gottwald, R. Elbert, T. Sarnow, T. Schmidt, K. B. Reith, H. Zadek, and F. Thomas, “Autonomous systems in intralogistics – state of the art and future research challenges,” *Publications of Darmstadt Technical University, Institute for Business Studies (BWL)*, vol. 124957, 2021.

[4] C. Bartneck, *Human-Robot Interaction: An Introduction*. Cambridge University Press, 2020.

[5] FAPS – Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik, “KoLAMeRo – adaptive und benutzerfreundliche kollaboration von menschen und autonomen mobilen robotern durch kontinuierlich lernende algorithmen,” May 2024. Accessed: 2024-04-09.

[6] H. Liu and L. Wang, “Gesture recognition for human-robot collaboration: A review,” *International Journal of Industrial Ergonomics*, vol. 68, pp. 355–367, 2018.

[7] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: MIT Press, 2004.

[8] M. A. Goodrich and A. C. Schultz, “Human-robot interaction: A survey,” *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[9] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4 of *Information science and statistics*. Springer, 2006.

[11] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[12] H. Dong, Z. Ding, and S. Zhang, *Deep Reinforcement Learning Fundamentals, Research and Applications: Fundamentals, Research and Applications*. 01 2020.

[13] Y. Lecun, *Generalization and network design strategies*. Elsevier, 1989.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, 1986.

[15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] G. van Houdt, C. Mosquera, and G. Nápoles, “A review on the long short-term memory model,” *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5929–5955, 2020.

[17] S. Mitra and T. Acharya, “Gesture recognition: A survey,” *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.

[18] T. Starner, *Visual recognition of american sign language using hidden markov models*. PhD thesis, Massachusetts Institute of Technology, 1995.

[19] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis, “A hidden markov model-based continuous gesture recognition system for hand motion trajectory,” in *2008 19th International Conference on Pattern Recognition*, pp. 1–4, 2008.

[20] N. Howe, M. Leventon, and W. Freeman, “Bayesian reconstruction of 3d human motion from single-camera video,” in *Advances in Neural Information Processing Systems* (S. Solla, T. Leen, and K. Müller, eds.), vol. 12, MIT Press, 1999.

[21] S. S. Rautaray and A. Agrawal, “Vision based hand gesture recognition for human computer interaction: A survey,” *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[22] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi, “Accurate, robust, and flexible real-time hand tracking,” in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI ’15, (New York, NY, USA), p. 3633–3642, Association for Computing Machinery, 2015.

[23] A. Börold, D. Schweers, and M. Freitag, “Towards multimodal information systems for assisting humans in production and logistics processes,” *Procedia CIRP*, vol. 120, no. 10, pp. 1089–1094, 2023.

[24] K. Alexander, K. Karina, N. Alexander, K. Roman, and M. Andrei, “Hagrid – hand gesture recognition image dataset,” in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2024.

[25] M. Lieret, M. Hübner, C. Hofmann, and J. Franke, “Human detection and gesture recognition for the navigation of unmanned aircraft,” in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pp. 831–838, SCITEPRESS - Science and Technology Publications, 2022.

[26] I. Guyon, V. Athitsos, P. Jangyodsuk, and H. J. Escalante, “The chalearn gesture dataset (cgd 2011),” *Machine Vision and Applications*, vol. 25, no. 8, pp. 1929–1951, 2014.

[27] X. Cucurull and A. Garrell, “Continual learning of hand gestures for human-robot interaction,” 04 2023.

[28] M. Peral, A. Sanfeliu, and A. Garrell, “Efficient hand gesture recognition for human-robot interaction,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10272–10279, 2022.

[29] J. Han, L. Shao, D. Xu, and J. Shotton, “Enhanced computer vision with microsoft kinect sensor: A review,” *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

[30] A. Nishikawa, T. Hosoi, K. Koara, D. Negoro, A. Hikita, S. Asano, H. Kakutani, F. Miyazaki, M. Sekimoto, M. Yasui, Y. Miyake, S. Takiguchi, and M. Monden, "FAce MOUSe: a novel human-machine interface for controlling the position of a laparoscope," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 5, 2003.

[31] M. Terreran, L. Barcellona, and S. Ghidoni, "A general skeleton-based action and gesture recognition framework for human–robot collaboration," *Robotics and Autonomous Systems*, vol. 170, no. 4, p. 104523, 2023.

[32] Y. Du, Y. Fu, and L. Wang, "Skeleton based action recognition with convolutional neural network," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 579–583, 2015.

[33] l. Chuankun, P. Wang, S. Wang, Y. Hou, and W. Li, "Skeleton-based action recognition using lstm and cnn," 07 2017.

[34] Liang Zhang, Guangming Zhu, Peiyi Shen, Juan Song, Syed Afaq Shah, and Mohammed Bennamoun, "Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition,"

[35] Navneet Nayan, Debashis Ghosh, and Pyari Mohan Pradhan, "A cnn bi-lstm based multimodal continuous hand gesture recognition,"

[36] K. Xia, J. Huang, and H. Wang, "Lstm-cnn architecture for human activity recognition," *IEEE Access*, vol. 8, pp. 56855–56866, 2020.

[37] B.-W. Min, H.-S. Yoon, J. Soh, Y.-M. Yang, and T. Ejima, "Hand gesture recognition using hidden markov models," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, pp. 4232–4235 vol.5, 1997.

[38] L. Onnasch and E. Roesler, "A taxonomy to structure and analyze human–robot interaction," *International Journal of Social Robotics*, vol. 13, no. 4, pp. 833–849, 2021.

[39] K. Furmans, F. Schonung, and K. R. Gue, "Plug-and-work material handling systems," in *Proceedings of the 11th International Material Handling Research Colloquium (IMHRC)*, (Milwaukee, Wisconsin, USA), Progress in Material Handling Research, 2010. Paper 10, ISBN: 9781882780167.

[40] A. Trenkle, M. Göhl, and K. Furmans, "Interpretation of pointing gestures for the gesture controlled transportation robot "fifi"," in *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, pp. 721–726, 2015.

[41] T. Lindner, D. Wyrwał, and A. Milecki, "An autonomous humanoid robot designed to assist a human with a gesture recognition system," *Electronics*, vol. 12, p. 2652, 06 2023.

[42] P. Phueakthong and J. Varagul, "A development of mobile robot based on ros2 for navigation application," in *2021 International Electronics Symposium (IES)*, pp. 517–520, 2021.

[43] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “Mediapipe hands: On-device real-time hand tracking,” 2020.

[44] Kazuhito, “Hand gesture recognition using mediapipe.” https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe/blob/main/README_EN.md, 2020. Accessed: 2024-04-12.

[45] C. Kronast, C. May, P. Ziegler, and J. Franke, “Conception and evaluation of a digital twin for an autonomous intralogistics solution.” Project Work, B.Sc. Industrial Engineering and Management, 2023. Supervised by Prof. Dr.-Ing. Jörg Franke.

[46] StereoLabs, “Object detection - stereolabs,” 2024. Accessed: 2025-04-14.

[47] R. Bernardo, J. M. Sousa, and P. J. Gonçalves, “Survey on robotic systems for internal logistics,” *Journal of Manufacturing Systems*, vol. 65, pp. 339–350, 2022.

[48] Stereolabs, “ZED ROS2 Wrapper: ROS2 Interface for ZED Stereo Cameras.” https://github.com/stereolabs/zed-ros2-wrapper, 2025. Accessed: 2025-04-16.

[49] J. Bortz and N. Döring, *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler: Limitierte Sonderausgabe*. Springer-Lehrbuch, Springer Berlin Heidelberg, 2015.

[50] P. Cruz, J. Vásconez Hurtado, R. Romero, A. Chico, M. Benalcázar, R. Álvarez, L. Barona, and L. Valdivieso, “A deep q-network based hand gesture recognition system for control of robotic platforms,” *Scientific Reports*, vol. 13, 05 2023.

[51] M. Montazerin, S. Zabihi, E. Rahimian, A. Mohammadi, and F. Naderkhani, “Vit-hgr: Vision transformer-based hand gesture recognition from high density surface emg signals,” 2022.

[52] S. Hussain, K. Saeed, A. Baimagambetov, S. Rab, and M. Saad, “Advancements in gesture recognition techniques and machine learning for enhanced human-robot interaction: A comprehensive review,” 2024.

A Appendix

Gesture Recognition Package

Table A.8: Library Versions Used in the Gesture Recognition

Libraries	Used Version
Python	3.10.13
MediaPipe	0.10.9
NumPy	1.26.4
OpenCV	4.9.0.80
TensorFlow	2.15.0
PyTorch	2.2.0 + cu121 (official PyTorch build)
ZED SDK	3.9.0 (latest with CUDA 12 support)
`pyzed.sl`	3.9.0 (ZED Python API)
scikit-learn	1.4.1.post1
CUDA Toolkit	12.1 (compatible with PyTorch, ZED SDK, TensorFlow)
cuDNN	8.9.2 (required for TensorFlow 2.15 and PyTorch 2.2)

B Appendix

Gitlab Repository

The full source code and documentation are available on Gitlab: `https://git.faps.uni-erlangen.de/jahartmann/ma-vishakh.git`

Directory Structure

```
gesture_recognition/
|- gesture_recognition/
|  |- gesture_node.py
|  |- gesture_robot_controller.py
|  |- model/
|     |- static_gesture_final.keras
|     |- dynamic_gesture_final.keras
|- launch/
|  |- gesture_system.launch.py
|- config/
|- setup.py
|- package.xml
```