UNIVERSITY
OF OSLO

Master's thesis

Gesture Control of Quadruped Robots

A Study of Technological and User Acceptance Barriers in Real World Situations

Tale Hisdal Sandberg

60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2023

Tale Hisdal Sandberg

Gesture Control of Quadruped Robots

A Study of Technological and User Acceptance Barriers in Real World Situations

Abstract

With the recent commercial availability of advanced mobile robots, it will become increasingly common for people to interact with them. A promising approach to robot control is through visual gestures, as it is an intuitive and natural way for humans to communicate. Other methods of human-robot interaction such as using voice can be difficult in loud environments, and wearable sensors and external controllers can pose logistical challenges if several people must be able to control the robot at the same time. It then stands to question why gesture control of mobile robots is not extensively used in real-world use cases.

This thesis investigates whether there are technological or user acceptance barriers preventing the use of gesture control systems in real-world use cases. A gesture control system consisting of a human-pose estimator, OpenPose, and a long short-term memory (LSTM) classifier is implemented on a PC connected wirelessly to a quadruped robot, Spot, from Boston Dynamics. The gesture control system is then tested in a live user experiment where data on user experience and attitudes, as well as technical system performance, is gathered.

The findings are that users are positive about the implemented gesture control system, indicating that user acceptance is not a barrier to real-world application but that technological challenges to adoption remain. OpenPose requires high-resolution input images where the controlling user is clearly visible with good contrast to the background to produce reliable output. This can not be guaranteed in all real-world use cases. It is also too computationally expensive to run on the robot with acceptable latencies. Continuous wireless connectivity to remote processing is therefore needed, reducing the flexibility of the current solution.

With the state of the art of human pose estimation and the processing power available on mobile robots, gesture control is at present limited to controlled scenarios. Still, the positive attitude and feedback of the users of the solution indicate that as technology advances, gesture control will be an attractive mode of human-robot interaction.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost I would like to thank my supervisor Tønnes Frostad Nygaard for his invaluable guidance and support during this research period. Our regular meetings allowed for constructive discussions and questions which significantly contributed to the successful completion of this thesis.

I would also like to thank my co-supervisor Benedikte Wallace for her insightful and helpful contributions and valuable feedback to my research, keeping me on track.

Thank you to the Robin group, and my fellow students and friends for an excellent and enjoyable learning environment. I would also like to thank FFI for the great opportunity to work with this robot.

Finally, a huge thank you to my parents for their endless support and encouragement from start to finish, and to my significant other Ulrik Johan Vedde Tranvåg for his ceaseless support and motivation during this work.

Chapter 1

Introduction

In the last two decades, legged robots have gone from walking on flat ground in controlled settings to having the ability to maneuver common indoor and outdoor environments with more irregular terrain [1]. In June 2020, the perhaps best-known robot canine, Spot from Boston Dynamics, went on sale [2]. Out of the box, Spot can be programmed to move and gather data on its own, and it can be controlled remotely by a human operator using a wireless tablet.

As legged robots are deployed in the real world, how well they interact with humans becomes increasingly essential. This is a core topic in the study of Human-Robot Interaction (HRI) [3]. HRI seeks to *address the challenge of achieving efficient and seamless interaction* between robots and humans. The field is rapidly evolving and is applied in almost all robot tasks.

For humans, it is natural to communicate using a combination of speech and gestures [4], and it is a short leap to investigate the use of gestures to enrich human-robot interactions.

1.1 Motivation

Different approaches to gesture control have been explored, from the person using wearable sensors [5, 6], to only using onboard cameras and depth sensors [7, 8], or a combination of wearable and non-wearable sensors [9].

The Boston Dynamics Spot quadruped robot has integrated sensors, allowing it the ability to traverse difficult terrain, avoid obstacles, and either follow a pre-programmed route or be controlled via a wireless tablet. The system also has an Application Programming Interface (API) that allows user-defined software control of the robot and live video extraction from the robots' cameras. This opens up the possibility of experimenting with con-

trolling the robot using visual signals.

For humans, nonverbal communication such as gesturing, is both natural and instinctive. If gesture-based robot control can be achieved reliably in different environments and conditions, it can help reduce barriers to the adoption and use of robots.

We can also imagine situations where a quadruped assistant can not be pre-programmed or controlled using voice or radio. The environment may be too noisy for reliable voice control. There may be only one radio, while the robot should be controllable by more than one person.

Removing the need for extra equipment can also reduce the complexity of robot operations as fewer things must be kept track of, operable, and charged.

1.2 Research Question

The thesis investigates the following research question:

> What is preventing visual gesture recognition from being an efficient way of controlling mobile, legged robots in real-world use cases?

The scope of the investigation will be limited to the *readiness of available technology* and to *user attitudes* to a gesture-based control scheme.

1.3 Research methods

The chosen research approach is a combination of quantitative/qualitative and experimental. First, a gesture-based robot control system is developed. Then, an experiment is conducted with test users to collect qualitative feedback on their experience and record their attitudes to operating the robot using gestures. Quantitative measurement data on the technical performance during the experiment is also collected. The combined qualitative/quantitative data set forms the input to a discussion and contributes to an answer to the research question.

1.4 Ethical considerations

The development and use of Robotic Systems raise ethical considerations. A review was done by Torresen [10] where *Ethical Societal Challenges Arising*

with Artificial Intelligence and Robots are discussed. Several of the dilemmas raised are relevant to this thesis, in particular:

- Robots must be *safe* and *responsive* to be accepted. You should not need to fear harm from a robot and it must do what you tell it to do with acceptable delay - such as stopping or changing direction.
- Data collected by a robot must be secure and not be misused. This applies both when collecting and using data to build models and when the robot is operated using gestures.

Others argue that training machine learning models on large data sets is energy intensive and raises issues when electricity is not from renewable sources [11, 12].

Robot safety The robot used in this research has built-in object detection which stops it from walking into and potentially hurting people. Additionally, an emergency stop function that cuts all motor power was implemented to stop the robot should it be necessary.

Data security and privacy Video was recorded by the robot's cameras and transmitted using the robot's built-in wifi modem. Only the controlling laptop and tablet were connected, and the robot was never connected to a public network. Any risk of video of test participants leaking is thus limited.

We applied for permission to process personal data at Sikt, which was granted to this thesis, allowing us to process personal data and keeping us accountable if any data misuse were to happen. The participants were informed of their rights. The participant consent form with all the details can be found in the appendix.

Participants in the experiment had consented to data being recorded and answers to questionnaires being used. The raw video was deleted once the post-experiment technical analysis was completed. See section 5.2 for a brief discussion.

Data used to train the model came from recording the author and, to a limited extent, through synthetic generation.

Energy/climate impact The most energy-intensive part of the project was training the LSTM machine learning model. Multiple iterations were necessary and each took around four hours to complete on a standard laptop - see section 4.1.

According to Boston Dynamics [2], the robot has a battery capacity of 564 Wh and an average running time of 90 minutes, which equates to an average power consumption of 364 W.

Given the relatively modest power requirements of the equipment used and that both model training and robot operation was done using Norwegian grid electricity, which according to [13], was 98% from renewable sources in 2020, the project's climate impact can be considered negligible.

1.5 Outline

This thesis is divided into eight chapters. Chapter 2 gives an introduction to the theory relevant to the thesis. Chapter 3 explains the equipment and tools used to develop the gesture control system and conduct the user experiment. Chapter 4 describes the implementation process and how the final solution was reached.

In Chapter 5, the user experiment and the obtained results are presented. Chapter 6 discusses the experiment results in relation to the research question. Chapter 7 concludes the thesis, and Chapter 8 contains suggestions for future work.

Chapter 2

Background

This chapter introduces the theory and other work that is relevant to this thesis, with an emphasis on Machine Learning and Human-Robot Interaction.

2.1 Machine Learning

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that focuses on how a computer system can find patterns in complex data in order to solve a task. Flach [14] defines ML as *finding a mapping* from data to an output that allows a given task to be achieved. A task can be predicting a value, known as *prediction*, or deciding between classes, known as *classification*. ML *algorithms* are generic and can be used across application areas. The choice of algorithm depends on the type of problem to be solved, and the *model* that is built will depend on the data available for learning and what *features* in the data that are used in training. A trained model is a *simplified representation of the phenomenon of study* and it must be *specific for the task we want to perform* [15]. A model built to predict the next word in a sentence will be different from a model to classify hand-written digits.

ML can be divided into three main categories as depicted in figure 2.1. The categories differ by the type of data used to train models.

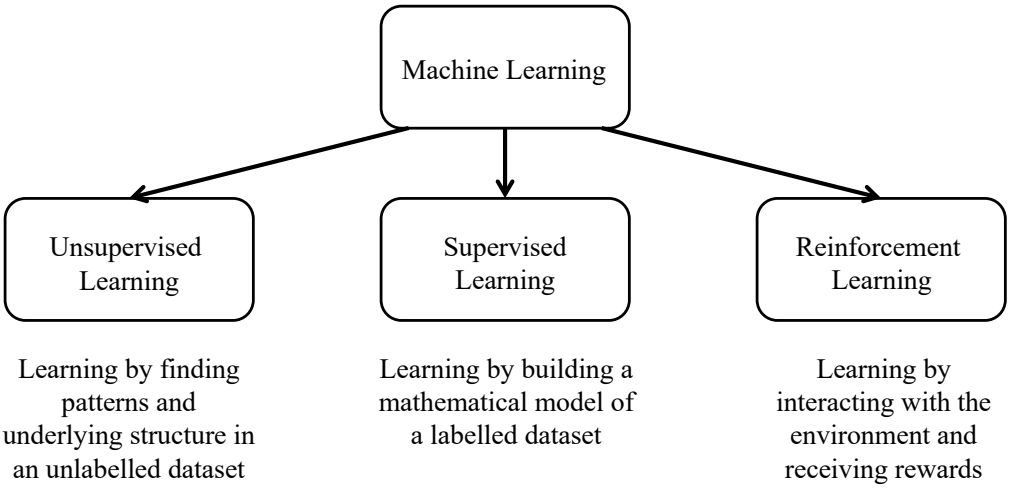


Figure 2.1: Machine Learning categories

2.1.1 Supervised Learning

Supervised learning (SL) is among the more commonly used ML methods [16]. SL uses labeled data sets to train algorithms to classify data or predict outcomes accurately. In a labeled data set, for each input training example, the algorithm knows what the correct output is. The algorithm uses this knowledge to try to generalize to new examples it has never seen before. The algorithm learns by iteratively making predictions on the input data, measuring its accuracy using a loss function, then adjusting itself using the difference between the predicted output and the correct output. A much-used training algorithm in SL is gradient descent, where a function for the gradient of the loss (error) function is derived, and its parameters are then tweaked until the error reaches a minimum. The derivation of these parameter values is the actual 'learning'. Once a minimum has been reached, the resulting parameter values are used in the original function.

A large number of supervised learning algorithms have been devised. Their use is often divided into *classification* and *regression*, and many can be used for both tasks, such as linear models, k-Nearest Neighbors, decision trees, and neural networks.

- Classification is when a model is trained on examples where the ground truth is a distinct class, such as "spam" or "not spam" in an email spam filter. The model's success is then measured by how accurately non-training test data examples are assigned to the correct categories.

- Regression is when the relationship between dependent and independent variables is modeled. A trained model is then used to predict an output value, given a particular input, such as the number of a certain product one can expect to sell given certain conditions.

2.1.2 Unsupervised Learning

Unsupervised learning (UL) analyzes and clusters unlabeled data sets. This method discovers hidden patterns or groupings in data without the need for human intervention. UL models are most commonly used for clustering, association, and dimensionality reduction.

- Clustering is where the algorithm groups the data on similarities found in the data.
- Association methods look for relationships between variables in the data.
- Dimensionality reduction aims to transform high-dimensional data into low-dimensional data by reducing the number of input variables in the data while preserving as much of the information as possible. Often dimensionality reduction is used in the pre-processing data stage to remove noise and improve the quality of the data.

UL has been an essential factor in creating renewed interest in deep learning but has since been overshadowed by the success of SL [16].

2.1.3 Self- and Semi-Supervised Learning

This is a middle ground between supervised and unsupervised learning. In self-supervised learning, the computer generates labels (in some way, e.g. reading the caption on images or headings on articles) in the first stage of learning and then does supervised learning based on these in the second stage. In semi-supervised learning, we make use of a data set with both labeled and unlabeled data. This is useful when there is a high volume of data, and only a small portion of it is labeled due to labeling being expensive or difficult to accomplish.

The most visible success of these approaches is the recent surge in large language models (LLM), such as the GPT-x family from OpenAI [17]. GPT-4 can deliver human-level performance on a range of university exams, and it is already available in products to aid software developers such as Git-Hub Co-Pilot [18].

2.1.4 Reinforcement Learning

Unlike SL and UL, reinforcement learning (RL) does not learn from a pre-existing data set. Rather, an RL agent learns from interacting in an environment and receiving rewards for wanted behavior and punishments for unwanted behavior. The RL framework is described by the problem of optimal control of the Markov Decision Process (MDP) [19]. The MDP environment consists of a set of states, a set of actions, and a set of rewards. A state can be an agent's position in the environment or its observation of the environment. The agent can take different actions in each state to progress to the next state. Every action will give a different reward. The RL agent has a reward function that maps each perceived state of the environment to a single number. The reward function defines a goal for the agent, giving it the immediate benefit of being in a specific state. The agent also has a value function. The value function captures the amount of reward the agent can expect to accumulate starting from the state it is in and into the future. The RL agent aims to discover the optimal action policy which maximizes the cumulative reward it can extract from every state in the environment.

2.2 K-Nearest Neighbors algorithm

KNNs can be used for classification and regression tasks [14]. It works by placing the labeled examples of a 'training' - no parameter training is involved - set into an n-dimensional space, one dimension per feature in the examples. The algorithm works on the premise that examples that are close to each other in this space belong to the same class. When new, unlabeled examples are to be classified or given a predicted value, the distance - typically Euclidian distance - to all of the examples in the training set is computed, and the new example is labeled the same as the majority of the k-nearest members of the training set in a classification task and a computed average of the distances when predicting a (continuous) value.

2.3 Neural Networks

Neural Networks (NN) are inspired by the structure of the human brain. A NN consists of layers of nodes; an input layer, one or more hidden layers, and an output layer. Each node in one layer is connected to every node in the next and previous layer, as shown in figure 2.2. Every node has an associated weight, an activation function, and a threshold value. All inputs to a node are multiplied with their respective weights and then summed. The output is

then passed through the activation function. If this output exceeds the node's threshold value, then that node is activated, sending data to the next layer in the network. NN's can, for instance, be trained through backpropagation, a gradient descent approach, which calculates the error associated with each neuron enabling the models' parameters to be adjusted appropriately.

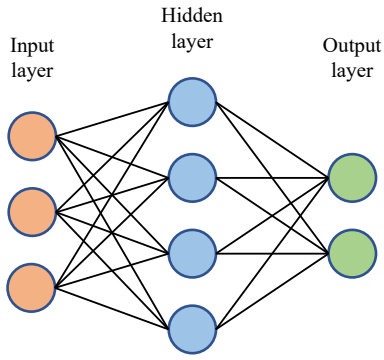


Figure 2.2: Neural network with one hidden layer

2.3.1 Deep Neural Networks

Deep Learning (DL), as shown in figure 2.3, is a subset of ML that refers to complex or 'deep' models, including NN's with more than three layers; an input layer, two or more hidden layers, and an output layer.

DL is often referred to as scalable ML because it automates a lot of the feature extraction process away, eliminating some of the human intervention involved in the pre-processing of data [16]. DL can use both labeled and unlabeled data. A deep neural network operates by refining and optimizing its categorization or prediction in each layer through forward propagation. Then it proceeds with backpropagation to calculate errors in predictions, adjusting the weights and biases. The combination of forward propagation and backpropagation enables the network to become more accurate over time as errors in predictions can be corrected correspondingly.

DL algorithms have a complex structure, and there are various types of NNs that target specific tasks or datasets. An example of a task is detecting features or patterns in image data for object detection or recognition. For such a task, a Convolutional neural network (CNN) could be implemented [20, 21]. For data such as natural language and speech recognition, a Recurrent neural network (RNN) would be appropriate, as it utilizes time dependencies in sequential time series data [22, 23].

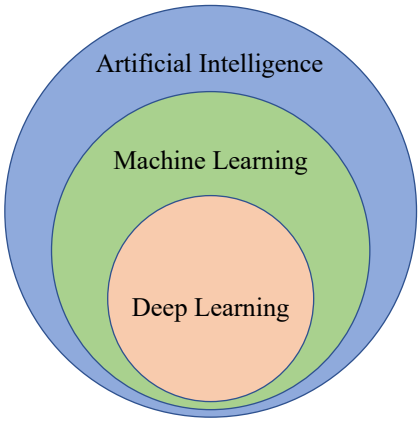


Figure 2.3: Overview of sub-fields: machine learning is a sub-field of artificial intelligence, deep learning is a sub-field of machine learning and includes deep neural networks.

Convolutional Neural Networks

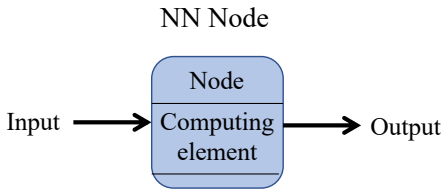
CNNs stand out from other NNs due to their exceptional performance with image, audio, and speech data [24, 25]. As a DL network algorithm, CNNs provide a scalable approach to image classification and object identification due to replacing the need for manual human feature extraction methods. CNNs apply matrix multiplication from linear algebra to identify patterns in an image. A CNN comprises three main types of layers; a convolutional layer, a pooling layer, and a fully-connected layer. The convolutional layer is the first layer and can be followed by a combination of more convolutional layers or pooling layers. The last layer is a fully-connected layer. The first layers of the CNN target simple features in the image data, such as edges and colors. The deeper layers focus on the larger shapes of the object being detected. As a result, CNNs perform exceptionally great when processing spatial data.

Recurrent Neural Networks

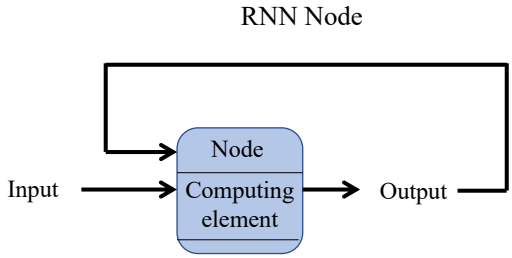
CNNs are suitable for classifying data - but not when dealing with time sequence data with long-term dependencies. When working with time sequence data, it can be challenging to figure out the correct output when there is no memory element present to remember past sequence information. RNNs present a method of remembering by taking information from previous inputs to affect the current input and output [26].

Similar to a Neural Network (NN) node, in the first step, the Recurrent Neural Network (RNN) node takes in some input which it processes with its

computing element, resulting in some output. In the next step, the RNN differs from the NN node, as it not only receives a new input but the output from the previous step as well, as shown in figure 2.4. The new input and the output from the previous step are processed resulting in a new output. This allows the RNN node to remember previous steps in the sequence.



(a) A neural network node. Receives new input which is processed in computing element and results in new output.



(b) A recurrent neural network node. The new input and the output from the previous step are processed in the computing element and result in new output.

Figure 2.4: Difference between a neural network node and recurrent neural network node

RNN, however, suffers from the long-term dependency problem. The long-term dependency problem is if the current state prediction (output) is dependent on the influence of a previous state. However, if this previous state is not in the recent past, then the RNN can struggle to predict the current state accurately. The gap between relevant information from previous states and the current state where the information is needed becomes too large.

In theory, RNNs are able to handle such long-term dependencies; however, in practice, they fail due to the *vanishing gradient problem*. The vanishing gradient problem occurs because the neural network uses the gradient descent algorithm to update its weights. The gradients become increasingly smaller, closer to the lower layers. At a point, the gradients are so small that they stay constant. A constant gradient results in no changes or improvements to the model leading to no differences in the output. The RNN stops 'learning' and fails to give accurate predictions [27, 28].

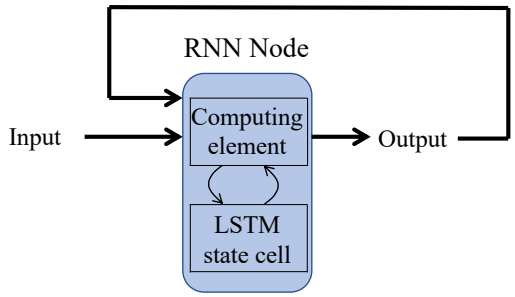
LSTM

The Long short-term memory (LSTM) is an RNN architecture that presents a solution to the long-term dependency problem (and vanishing gradient problem) [29]. The LSTM adds an internal state to the RNN node, see

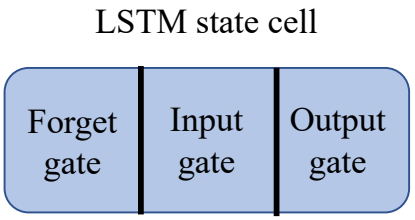
figure 2.5a. Now the RNN node receives an input from the current step, the output from previous steps, and state information from the LSTM state.

The LSTM state is a cell containing three gates; the forget gate, the input gate, and the output gate shown in figure 2.5b. These gates control the flow of information needed to predict the output. The forget gate filters the stored state information and removes data that is no longer contextually relevant. The input gate decides what information should be added or updated in the working storage state information. The output gate controls which part of the stored state information should be output in this particular instance. The gates are assigned numbers from zero to one, where zero means the gate is effectively closed, and one means the gate is wide open and everything gets through.

The processing part of the RNN now has this additional state information which provides some additional context.



(a) A LSTM node. A RNN node with a LSTM state cell.



(b) LSTM state cell showing the three gates it consists of.

Figure 2.5: The RNN node with the additional LSTM state cell.

Sliding Window Technique

The sliding window technique is a well-known approach used for reducing time complexity [30] by reducing the use of nested loops. The technique is applied to an array, list, or string-type data structure. A fixed *window size* is chosen, then looped over the data structure, moving the window by one in every subsequent step in the loop, as shown in figure 2.6.

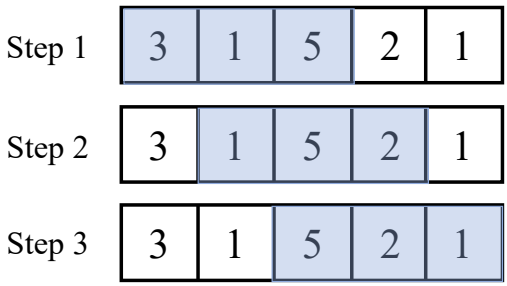


Figure 2.6: Sliding window with window size = 3. For each new step the window moves one value to the right across the array.

2.4 Robotics

Definitions of robotics center around the study and development of machines that can act in the physical world; *The design, construction, and use of machines (robots) to perform tasks done traditionally by human beings* [31] and *The science of making and using robots (machines controlled by computers that are used to perform jobs automatically)* [32]. Redfield [33] proposes a new definition; *The scientific and engineering discipline concerned with the creation, composition, structure, evaluation, and properties of embodied artificial capabilities.* Following this definition, robotics is the creation or study of artificial physical entities that have capabilities. A completely virtual capability, e.g., an online chatbot, is not part of the field, while a program that controls a robot is.

2.5 Human-Robot Interaction

Human-Robot Interaction (HRI) is a multi-disciplinary field of study concerned with how robots and people interact [4]. The field brings together engineers, psychologists, designers, anthropologists, and sociologists as well as experts in specific application areas, e.g. manufacturing or healthcare, to design robots and the *interaction between robots and humans.*

Interaction means communication in some form. For any robot operating in the *proximity of humans*, communication between human and robot is necessary. The absolute minimum requirement is to have a way of halting the robot for safety reasons.

A mobile robot such as Spot [2] from Boston Dynamics is controlled through commands sent using a wireless pad, and sends video back to the operator. It can also be equipped with a speaker and microphone to allow

voice communications.

2.5.1 Visual human-robot communication

While speech and verbal communication are clear choices when deciding how to convey a message and have been used in many HRI applications [34, 35], non-verbal communication offers interaction cues that are central to interaction between robots and humans [4].

Human-to-human communication is rich in non-verbal signaling. We convey information through facial and hand gestures and body poses. HRI is concerned with non-verbal communication both to and from the robot. Well-designed gesturing by the robot will help convey intent and meaning to humans and vice versa when the robot has the capability to detect and interpret gestures made by a person. Examples can be pointing to an object asking the robot to "pick it up" [36]; the pointing gesture makes it clear which object is meant. Scratching the head can convey "I did not understand".

Implementing non-verbal communication in robots can span from the simplistic - e.g. pre-programmed actions to accompany spoken feedback, such as acknowledging an understood command by making a "thumbs-up" gesture, to the more complex case of the robot visually detecting and interpreting a gesture made by a person in a timely manner - not too quickly and not too slowly. Machine recognition of gestures can be done by interpreting poses from camera input or other onboard robot sensors [37, 8, 38, 7, 39, 40, 41], by using data from sensors worn by a person [5, 6], or in a combined sensor/visual setup [9].

Collaborative robots (cobots) A growing use case for visual human-robot communication is with collaborative robots, primarily in industrial settings.

The first industrial robot, Unimate, started work at the General Motors car factory in Trenton, USA, in 1961. It was capable of doing a single operation, extracting parts from a die-casting machine [42]. Today industrial robots are ubiquitous and 3 million were in operation in 2021, with this number expected to grow [43]. See figure 2.7 .

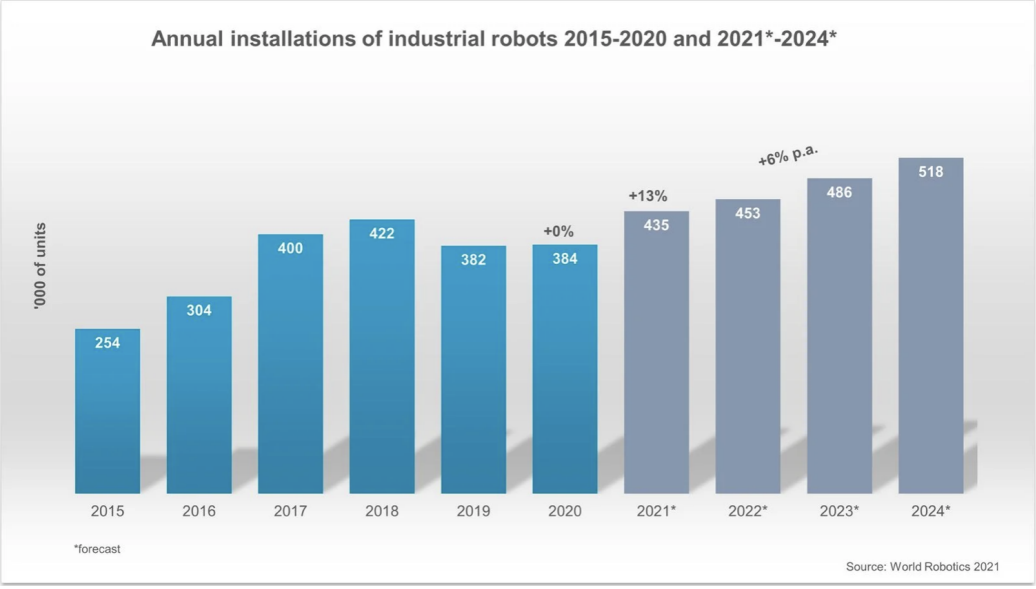


Figure 2.7: Historical and projected growth in deployment of industrial robots [43].

Traditional industrial robots are stationary and programmed to perform simple repetitive tasks. The robots only interact with people when they are being programmed, and they are kept separate from humans for safety reasons.

This paradigm is changing. In the factory of the future, people and robots are expected to work together as a team. The cognitive skills and dexterity of the human and the strength, tirelessness, and precision of the robot allow for both efficiency and flexibility in production [44]. To achieve this, the interaction between humans and robots must be intuitive, and the robot must be able to sense the people around it and understand what the human co-worker wants it to do [45].

In an industrial setting, the noise levels can make voice communication impractical. When teams consist of more than one person and one or more mobile robots, all human team members must be able to communicate with the robot to order it to stop for safety reasons and possibly to ask it to perform a task. Equipping all with sensors or wireless controllers is a logistical and cost challenge. Gesture and eye tracking are therefore proposed as a viable way forward [46]. An example cobot is the 'Walt' robot deployed by Audi. Walt can learn by observing how a person performs a task, and the robotic arm can be controlled using hand gestures [47].

2.5.2 Human Pose Estimation

There is growing agreement that non-verbal cues are central to successful human-robot interaction [4]. For the robot to pick up on human signaling, face, eye, and gesture recognition is important. Gesture recognition by the robot can be implemented through Human Pose Estimation (HPE). HPE attempts to map the *pose* of a person by creating a model and detecting the relative positions of the person's joints, as is the case in figure 2.8, or by creating a more lifelike model where the shape and volume of the body are retained as depicted by the middle and right illustration in figure 2.9.

Figure 2.8: Rendering of the output from OpenPose, a Human Pose Estimation system

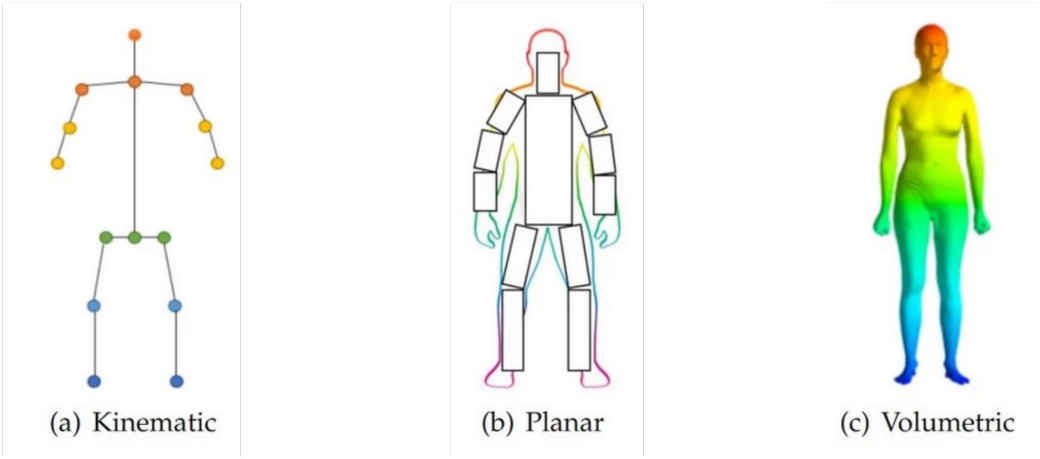


Figure 2.9: Human pose body models [48].

Pose detection is divided into two-dimensional (2D) and three-dimensional (3D) approaches. Three-dimensional pose estimation relies on input from at

least two cameras or points of view to estimate depth, while two-dimensional approaches take two-dimensional images as input and predict the pixel positions of joints in the picture. A comprehensive review of the latest research on HPE has been done by Zheng et al. [48].

Two main approaches exist for 2D multi-person HPE; Top-down, where the individual people in an image are detected first, and the joint position of each person is then estimated. The other approach is bottom-up, where all body-joint candidates are predicted from features in the image and then assembled and allocated to individual bodies.

Top-down approaches rely on standard box approaches for identifying individuals and then use CNNs to build heatmaps to identify joints. An example top-down model is TokenPose [49], where the top-down approach is expanded on by adding tokens for each keypoint (joint) in the training data images. The model uses the tokens to learn the relationships between joints and not just features of the image. This has allowed for performance on par with existing CNN-based counterparts while reducing the processing load.

OpenPose is an example of a bottom-up network for HPE. OpenPose can achieve real-time performance on specialized hardware but struggles when images are low resolution or when there are occlusions [50, 48]. Proposals to address these challenges have been made, among them PifPaf, which shows better accuracy than OpenPose on low-resolution images and when occlusions are present [51]. Still, challenges remain; Correctly allocating body parts in situations with significant occlusion, reducing the processing need to allow the networks to be implemented on edge devices, e.g., mobile robots, and the lack of training data for unusual poses [48].

Chapter 3

Tools and Frameworks

This chapter presents an overview of the main software and hardware used.

3.1 The Spot robot

Though primarily promoted by Boston Dynamics as a data-gathering *platform* [52] which can be tailored to a range of application areas, e.g., autonomous operation in hazardous locations [53], or as in Wetzel et al.'s [54] work where the robot is a lower cost alternative to human-operated in-doors LiDAR mapping in construction. Spot has also, for example, been evaluated as a 'tool tracker' on building sites [55] and a guide dog for the visually impaired [56].

The robot can be controlled with a tablet which is connected wirelessly. It can be pre-programmed to follow a path and then do so without direct supervision.

Extracting data from - and sending commands to - the robot can be done by scripts in Python that access the Spot API running on a remote computer.

By default, Spot has five user-accessible cameras, which together offer a 360 degrees field of view. In addition, it has an inertial measurement unit (IMU) and position/force sensors in its four legs. The robot uses the cameras to avoid obstacles when navigating a route, and imagery can be transferred in real-time to a controlling tablet or a computer connected to the robot. Additional cameras, such as the Boston Dynamics 'Spot CAM+', can be mounted on the device, see figure 3.1. Spot CAM has a 360 by 170-degree view, can be panned and tilted, and comes with LEDs for illumination and microphone/speakers, enabling two-way audio.

Figure 3.1: Spot from Boston Dynamics with the Spot CAM+ mounted on it.

3.2 OpenPose

OpenPose [57] is described as a *real-time multi-person human pose detection library*. OpenPose takes video or still images as input and generates arrays containing the body-part in-picture pixel positions when using its Python API or JSON-formatted text files when running on the command line. The recognition is achieved 'bottom-up' using a CNN to extract features and compute confidence maps - indicating the confidence that specific body parts are in specific pixel positions - and then predict the association between the parts, including which person they belong to. The result is the identification of all the people in the image and the location of their body parts. Figure 3.2 shows an example of where OpenPose has found the joints of the person in the image.

OpenPose has been proposed as a good alternative to current Motion capture approaches that use manual inspection, sensors attached to the body, or Kinect for human pose estimation and movement classification [58, 59]. It has been used in combination with the YOLOv5 object detection network to build models to detect if workers are wearing safety harnesses [60], and it is proposed as a solution to fall detection that does not require people to wear sensors [61].

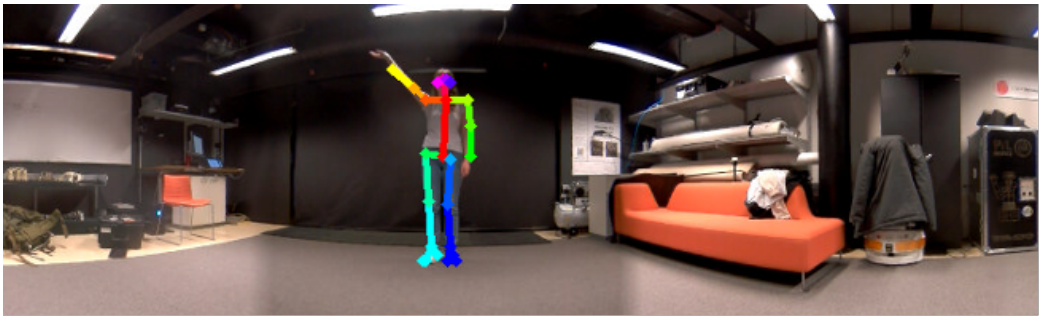


Figure 3.2: Image from OpenPose where the joints of the person in the image have been successfully identified and positioned

3.3 Tensorflow / Keras

TensorFlow is a machine learning platform developed by Google Brain and released under an Apache license in 2015. Version 2 came in 2019 and is the one used by Keras [62].

Keras is a set of APIs designed to simplify the use of Tensorflow when building machine learning applications in Python. Keras is open source but primarily maintained by Google. According to the Kaggle survey of 2022 [63], it is among the three most adopted machine learning frameworks.

Applications built using Keras can be run on hardware with and without GPU support, even small devices such as mobile phones and, as in our case, laptop computers.

3.4 OBS Studio

OBS Studio [64] is an open-source application developed by the OBS Project. It can do real-time capture, scene composition, recording, encoding, and broadcasting via Real Time Messaging Protocol (RTMP). Images can be extracted through a Python API as opencv2 objects. It allows live-streaming to YouTube, Facebook, Instagram, and Twitch.

The application runs on multiple operating systems, Windows, macOS, BSD, and Linux, and can encode video into H.264/MPEG-4 AVC or H.265/HEVC formats.

Chapter 4

Implementation

This chapter gives an overview of the process and choices made leading up to the final implementation. Building the solution which was used with the robot in the user experiment discussed in chapter 5 turned out to be a highly iterative process of gathering and processing data, designing and discarding gestures, and adjusting model parameters and input features. Rather than discussing them in a separate chapter, smaller experiments and tests done during the implementation process are explained here. The training code, data sets, trained model, and user experiment code are open source and available at[1].

The main steps we went through were:

1. An initial set of 11 full-movement gestures were designed.
2. Gesture performance data was gathered by recording the author using the robot's camera.
3. OpenPose was tested on captured data to establish a performance baseline.
4. Gesture separability was tested using a simple k-Nearest Neighbors (kNN) algorithm. This led to changes to the feature selection used in the rest of the work.
5. An LSTM model was developed. This led to additional data capture, changes to the gesture set to address OpenPose shortcomings, and the reduction of gesture duration to meet performance requirements.
6. Once we had a working LSTM, the final selection of a gesture set to use in operation was made. Code to filter the output from the LSTM

[1] https://github.com/UiO-Robotics-and-Intelligent-Systems/master_talehs

model and thus control when to send commands to the robot was implemented.

4.1 System overview and design goals

The overall objective was to implement a fast and reliable visual-only gesture-command system that would allow us to experiment with robot interaction.

When used, the system should receive robot-captured video of a person who performs gestures, extract features from the image stream, and use a machine learning model to predict gestures with high accuracy. Commands are then sent to the robot. The overall delay should be low enough for users to perceive the robot as responsive, ideally in the range of 1-2 seconds. The process is illustrated in figure 4.1.

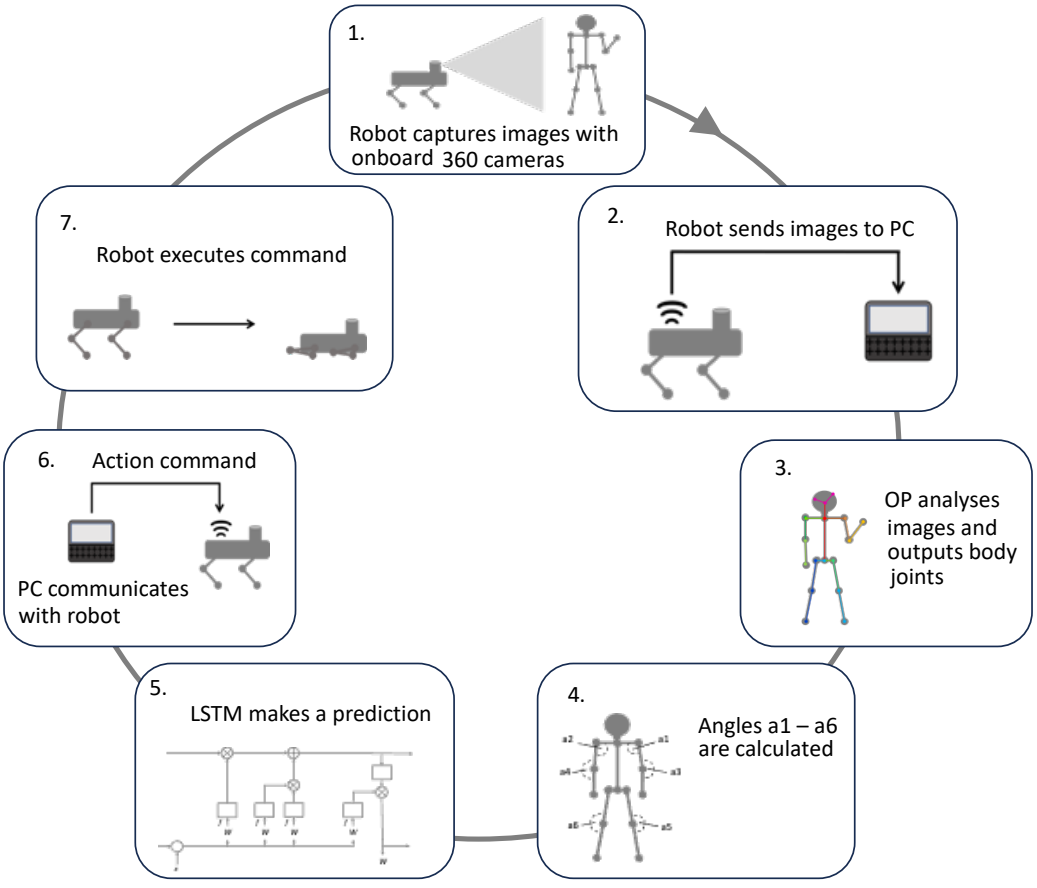


Figure 4.1: The main process steps from gesture capture to command execution

System overview The system consisted of the robot capturing video of the person performing gestures, which was then fed to OpenPose where joint positions were calculated. The OpenPose BODY_25 model was used, which is the most accurate but not the fastest available (the fastest is the COCO model but only when run on the CPU [65]). Features were extracted from OpenPose's output and given to an LSTM model for gesture prediction. The prediction output was filtered before commands were sent to the robot. The main hardware and software components are shown in the system diagram in figure 4.2.

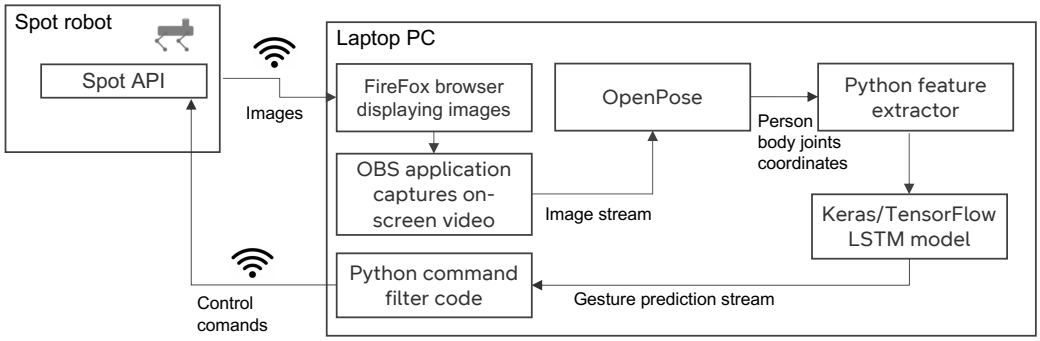


Figure 4.2: System overview

Laptop PC for processing A laptop PC running Microsoft Windows was used to capture video from the robot, do feature extraction, train machine learning models and do gesture recognition, and send commands to the robot. The PC had an Intel core i9-1088SH CPU running at 2.4Ghz, NVIDIA GeForce RTX 2070 Super with max-Q Design 24GB RAM GPU, and 32GB system RAM.

The machine was powerful enough to train an LSTM model on the entire data set in consistently less than four hours. This made it unnecessary to use larger GPU clusters during the project.

4.2 Initial Gestures

Gestures were initially chosen with the goal of making them different enough to enable high-accuracy recognition by an LSTM classifier. The classifier would be trained on sequences of feature vectors that were extracted from OpenPose joint-position output.

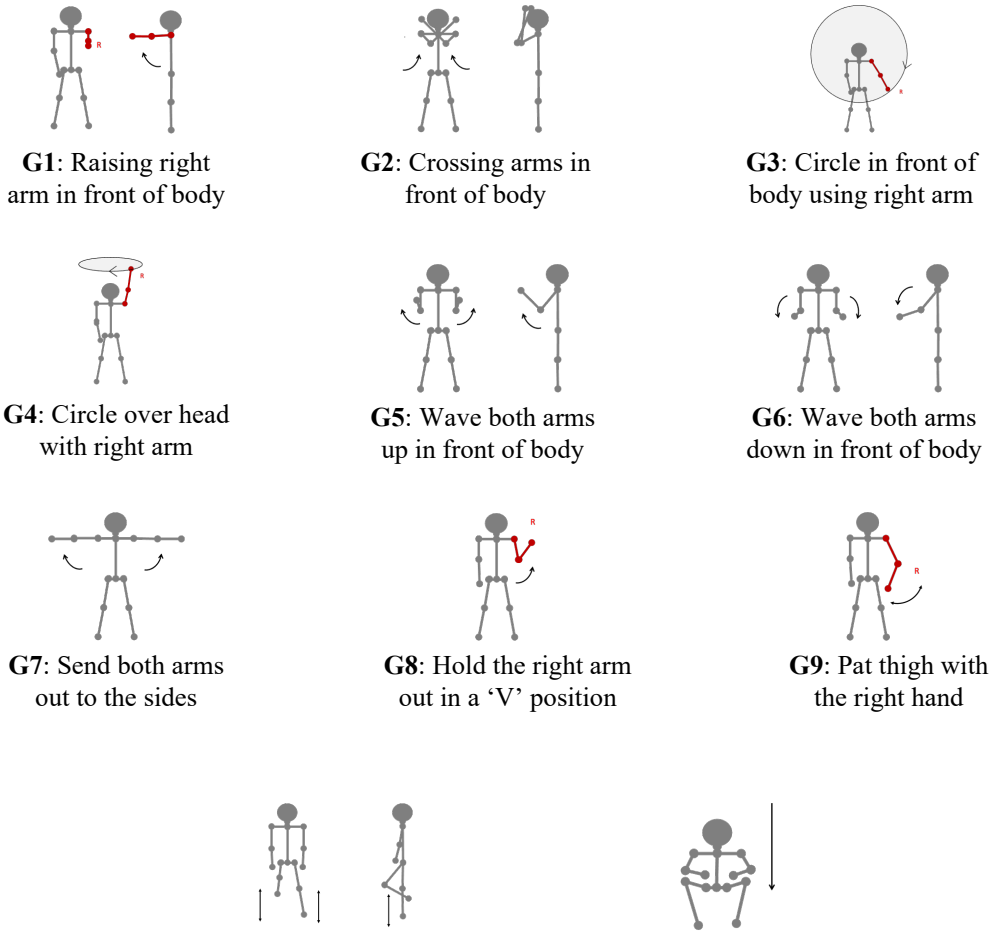


Figure 4.3: Eleven initial gestures. These were designed to be separable by an LSTM. 11 were made to have redundancy as only 6 were needed.

The original intent was to implement support for six different robot actions, each prompted by a different gesture. In the event that more commands would be needed in the future, or in case some of the gestures should perform poorly and have to be discarded, five more were designed. The initial set of 11 gestures, G1 to G11, are illustrated in figure 4.3. (As the work progressed, we decided to limit the movement in the gestures to reduce the number of images that had to be processed to recognize a gesture. Therefore, the final gesture set was closer to a set of different poses than full gestures. For simplicity, we will still use the term 'gesture' to refer to them in this thesis.)

In addition to the gestures in figure 4.3, the LSTM model was trained with

a gesture class labeled *neutral*. This class contained all the movements which a person performs that were not one of the command gestures. Examples of neutral gestures include standing still, walking, scratching the face or head, not facing the robot, and so on. A neutral class was included so that the model would have a *default* gesture to predict rather than always predicting one of the gestures which would initiate an action from the robot and then have to implement logic to prevent unwanted outcomes.

4.3 Gesture Data Gathering

The data was gathered by having a person, the *trainer*, stand in front of the robot and repeatedly perform the gestures. The trainer would start in a neutral position, which meant standing upright on both feet with their arms hanging at their sides, facing the robot. They would then perform a gesture and return to the neutral position once the gesture was complete before moving on to the next gesture. The trainer would go through each different gesture, then start at the first gesture again and repeat the same procedure.

Because all recordings were done of the same trainer, there was a risk that the resulting data set would have limited natural variation, and any model built using the data would not generalize well to other people. To mitigate this risk, the same gesture was never performed twice in a row. This reduced the probability that two performances of the same gesture would be very similar. Since live data capture was a very time-consuming process, we also decided to delay any potential additional data gathering which would involve other people until a real need was identified. As it turned out, we would later supplement with synthetically generated data and never needed to record additional trainers.

The gesture performances were captured by the robot's onboard 360 cameras at a rate of ten frames per second (fps).

Data capture conditions In order to test how well the robot's onboard cameras and OpenPose would work together under near-real-world conditions, gesture data was gathered with varying lighting, location, clothing, movement, and the trainer's silhouette. See tables 4.1 and 4.2 for a complete listing.

Much of the data was captured in a *motion capture room* (MOCAP), constructed to optimize motion capture. The room has controlled conditions for collecting data on gestures. The walls and ceiling are black, the floor is dark gray, and no natural light is present. Data was also captured outdoors in

the forest, with the trainer wearing camouflage clothing to test performance in a very challenging environment.

Table 4.1: Data capture variations in the MOtion CAPture room

Variables	**Variations**
Lighting	Variations in the lighting conditions in the room
Position	Trainer facing the camera
Position	Trainer facing sideways or with her back to the camera
Silhouette	Trainer facing the camera while wearing a big backpack to change the body shape of the person
Silhouette	Trainer not facing the camera while wearing a big backpack
Silhouette	Trainer facing the camera with different objects between the camera and the trainer, partially obscuring the trainers body
Silhouette	Trainer facing the camera holding a long object to deform the length and position of the arms
Silhouette	Trainer wearing camouflage clothing and facing the camera while holding a long object
Motion	Trainer walking past the camera while performing gestures
Motion	Robot walking past the trainer while the trainer was performing gestures
Motion	Robot jogging on the spot while the trainer was performing gestures
Distance from camera	Gestures were performed at a range of distances from the robot's camera, approximately [0.5 - 3] meters

Table 4.2: Data capture variations outdoors in the forest wearing camouflage clothing

Variables	**Variations**
Position	Trainer facing the camera
Position	Trainer facing sideways or with their back to the camera
Silhouette	Trainer facing the camera while wearing a big backpack to change the body shape of the person
Silhouette	Trainer facing the camera with a few trees between the trainer and the robot, partially obscuring the trainers body

4.4 OpenPose Performance on Robot imagery

Before building and training any models on the gathered data, we tested how well the *robot camera* $\rightarrow$ *PC* $\rightarrow$ *OpenPose* chain performed in order to create a baseline for what would be possible in a more extensive user experiment. The robot's onboard 360 camera (Spot CAM+) captures images which are sent to a computer where they are processed through OpenPose, and vectors of joint coordinates are produced. The process is illustrated in steps 1 to 3 in figure 4.1.

The different conditions in which the setup was tested while in the MOCAP room are listed in table 4.1, and the different conditions tested outside, in the forest, can be found in table 4.2.

The scenarios were constructed with the aim to get a good understanding of the combined performance of the robot's camera and OpenPose and, consequently, what scenarios would be possible in a real-world user experiment.

Robot camera performance The 360 camera performed well in all the different scenarios in tables 4.1 and 4.2. The images did not exhibit noticeable reductions in quality through blurring when the robot was in motion, nor did changes in lighting conditions significantly impact the produced images. The cameras adjusted for low and bright lighting very well but struggled when the lights were fully off, as would be expected.

The robot constructs a 360 view by combining images from five separate cameras. A scenario where problems occur because of this is conceivable but was not observed during testing.

OpenPose performance OpenPose performed well on robot imagery when the *contrast between the background and the person was sufficient* and *the person was not too small in the frame*. Even when objects partially obscured the person, or the person was at an angle to the camera, OpenPose was able to predict where the person's limbs would most likely be in the frame. Occasional errors could be observed when the colors of the person's clothes were close to the background or if the person had their arms in front of their body. OpenPose was very unreliable when the contrast between the person and the background became too low. Table 4.3 shows an overview of OpenPose performance and the conditions under which it was observed, and a more detailed write-up of the testing and results obtained can be found in the Appendix.

Table 4.3: Summary of OpenPose performance on captured data

Performance	**Conditions**
Optimum	The person was not obscured in the image and the person not too small in the image, the contrast between the person and the background was good (light and color contrast).
Occasional errors	When arms were placed in front of the body OpenPose would sometimes predict the placement of the arm incorrectly
Occasional errors	Wearing a big backpack when turned away from the camera. OpenPose would still generally make good predictions of shoulder and head positions
Occasional errors	Wearing camouflage in the MOCAP room with good lighting conditions, but where the background coloring could be confused with the clothing
Occasional errors	Wearing camouflage in the forest was challenging in general
Poor	Very low light in the MOCAP room generally meant the contrast was too low and OpenPose was unable to locate the person
Poor	Wearing camouflage in the forest, partly obscured by a thin line of trees made it impossible for OpenPose to detect the person

Learnings from the camera and OpenPose tests The camera and OpenPose testing showed us that robot image quality, in general, was good and not a concern for our future user experiment as long as completely dark conditions were avoided. OpenPose, however, proved to be more sensitive (see figure 4.4 for some scenarios), and a successful implementation and experiment would need to cater to that. Specifically:

- The test persons' clothing and the background seen by the robot need sufficient contrast. Since contrast is lost with very poor lighting, this also needs to be avoided.
- The test person must be tall enough in the frame sent from the robot to OpenPose. Since we did not control the vertical resolution or focus of the robot cameras, this meant that the distance from the robot to the person needed to be in a range of 1.5 to 2.5 meters. This would

ensure that the person was neither 'clipped' in the frame (as in out of frame) nor too small in the frame to be detected reliably by OpenPose.

- Since both robot operators and the robot may be moving during real-world operation, complete and consistent control of backgrounds will be difficult, and distances to the camera may vary. The implementation will thus have to manage intermittent poor data coming from OpenPose.

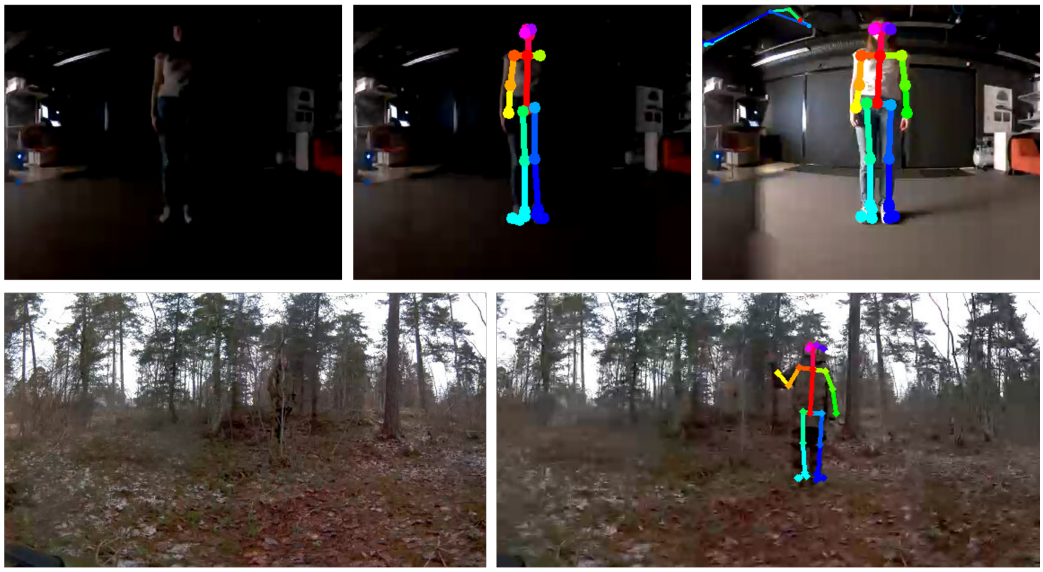


Figure 4.4: The top row shows how OpenPose performance improves with better illumination. Bottom left, the person wears camouflage clothing and is partly obscured by a small tree.

4.5 Testing Gesture Separability with kNN

After verifying that the setup with the robot camera and OpenPose worked, the next step was to see whether the initial gestures - encoded as sequences of OpenPose output vectors of joint pixel (x,y) positions - could be distinguished from each other by a classifier, given the variation in OpenPose output observed in the tests. We decided to use a simple k-Nearest Neighbor (kNN) classifier. Given a set of labeled feature vectors in a 'training' set, the kNN classifies vectors in the 'test set' with the majority class of their k nearest neighbors. In our tests, we used basic Euclidean distance calculations.

See the background chapter, section 2.2, for a brief introduction to the kNN algorithm.

The separability testing consisted of the following main steps:

- Test with the initial 11 gestures and a feature set made up of all the 25 x,y joint coordinates output by OpenPose. Frame sequences in the labeled 'training set' were padded with additional zero feature vectors when needed to make all gestures the 'same length'.
- Reducing the number of features by switching from joint coordinates to angles between joints.
- Cropping frame sequences to make them more distinct and adding a neutral gesture class.

In the next sections, we briefly discuss the tests before concluding on the learnings which are reflected in the LSTM model building. A summary table of all the tests done can be found in the Appendix, section 9.2.

Initial kNN separability test In the first test, the kNN was initialized with k=3. The training data set consisted of full gestures, i.e., all the frames in a sequence that made up a gesture. As some gestures took less time to perform than others, they would generate fewer input frames, and OpenPose would produce fewer feature vectors for the gesture. To ensure that the kNN training set had the same number of frames for each gesture, feature vector sequences were padded with zero vectors, as needed, to make each gesture sample the same length of 60 frames. The kNN had 2700 frames in the 'training set' and 660 frames in the test set.

The features used were the x- and y-pixel coordinates of the persons' joints as output from OpenPose, meaning there were 50 features, each a real number. The horizontal and vertical ranges were given by the resolution of the images coming from the robot; [0 - 1345] horizontally and [0 - 306] vertically. Please see figure 4.5, where the joints from 0 through 24 and the resulting feature vector are illustrated.

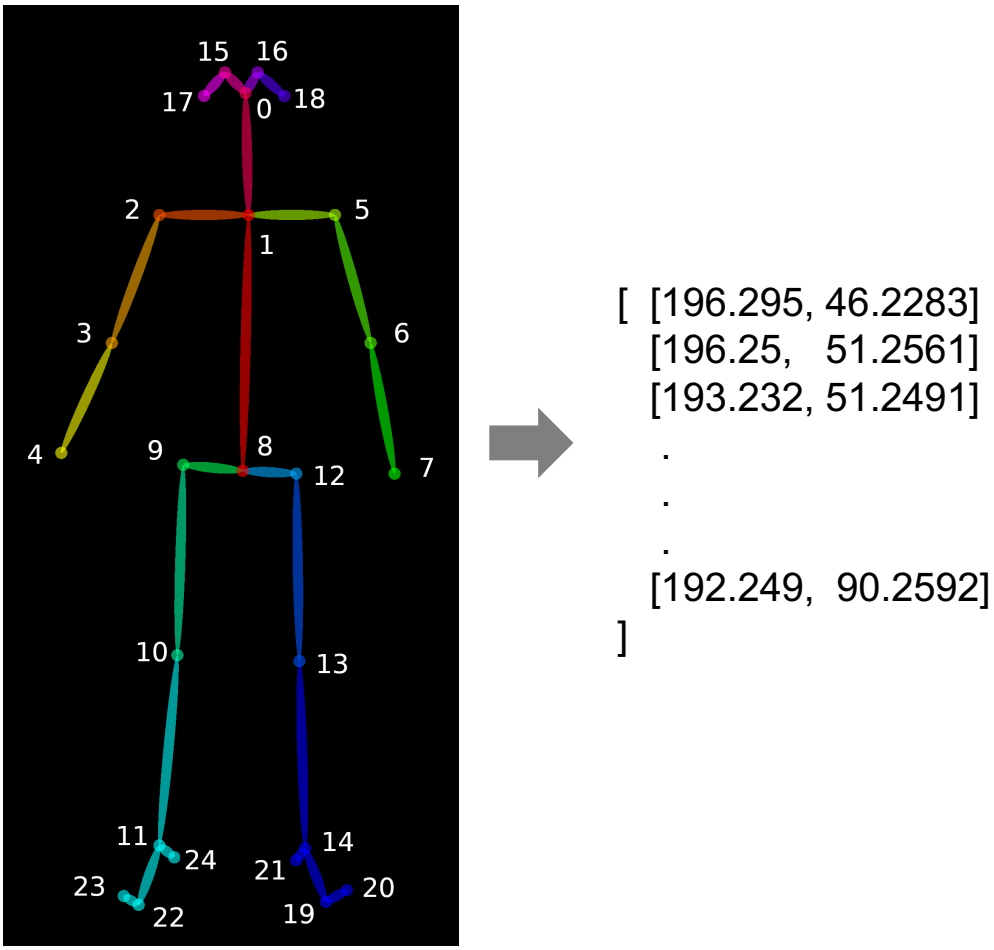


Figure 4.5: The 24 joint coordinate feature vector output from OpenPose [65]

The kNN achieved an overall accuracy of 36.36% on the test set. Though better than randomly allocating to categories, this poor result seemed to indicate that the gestures are not very separable, but there may be other explanations. Both the nature of the features (x, y coordinates that were not normalized), noise in the data from zero padding (frames with only zeros), and the sheer number of dimensions relative to the size of the data set ('curse of dimensionality [66]) may all help explain the initial performance.

Changes to the feature selection In the next rounds of testing for separability, the number of features was reduced from 50 to a range between six and 14.

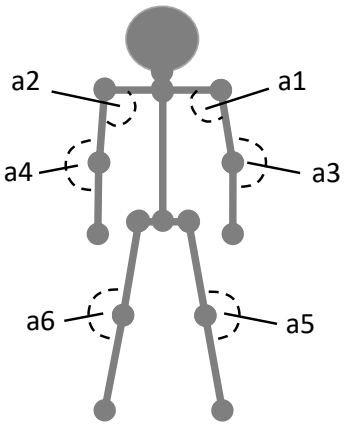
When removing features, we risk discarding information needed for dis-

crimination. To reduce dimensionality while avoiding this, different combinations of features were tested. Instead of simply reducing the number of body point coordinates and testing different combinations of these, selected *angles* between limbs were calculated and added as new features.

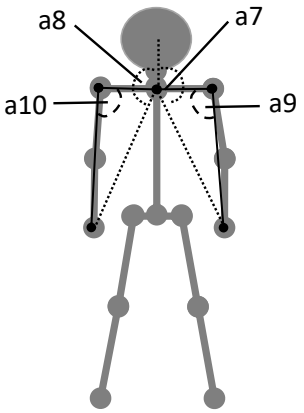
The chosen angles on the right and corresponding left side of the body were:

- **a1 and a2.** The chest-shoulder-elbow angles.
- **a3 and a4.** The shoulder-elbow-hand angles.
- **a5 and a6.** The hip-knee-foot angles.
- **a7 and a8.** The hand-chest-head angles.
- **a9 and a10.** The chest-shoulder-hand angles.

The angles are shown in figure 4.6. These angles were chosen as they were deemed to carry the most information for the designed gestures since the hips, chest, and head rarely changed positions compared to the other body parts. The angles had a range between 0° and 180°. Some of the tests also included the four x- and y-coordinate points of the left and right hands.



(a) Angles 1 to 6



(b) Angles 7 to 10

Figure 4.6: The angles calculated for use as input features

Using ten angles and no individual coordinates did not improve the results while reducing the feature set to angles one to six gave a slight improvement. See figure 4.7.

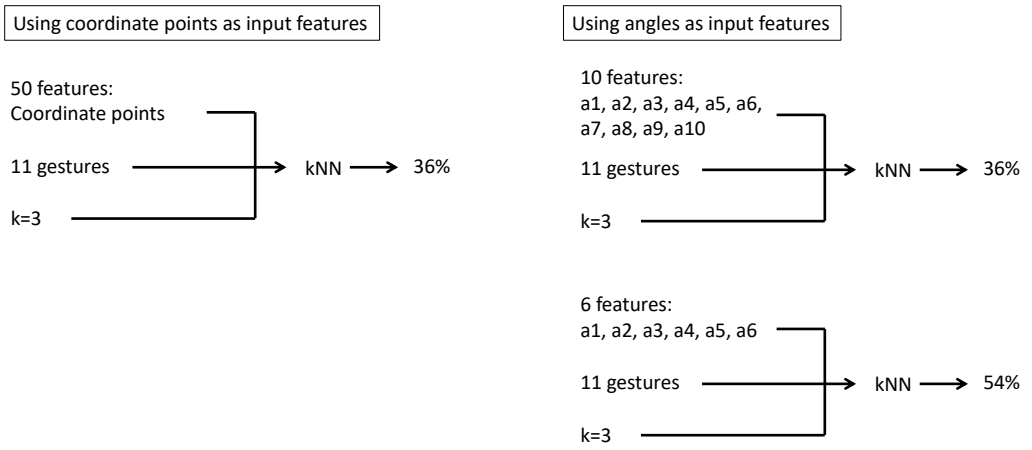


Figure 4.7: kNN when going from 50 features to 10 and subsequently 6 angles. Performance eventually did improve as the number of features was reduced.

Cropping frame sequences and adding a neutral gesture class While a performance accuracy of 54% when using angles 1-6 as features is an improvement, it is still not a great result. Therefore, in the third round of testing, reducing noise in the data set was carried out by removing the overlap between the gestures.

Each of the gestures 1-11 in figure 4.3 starts and ends in a neutral position. When a classifier that works with a single feature vector at a time - as opposed to the LSTM, which uses a whole sequence - is given a feature vector from one of the overlapping frames, the result is random. Reducing overlap was done by processing the image data of the gestures in the training set to only contain the end pose for each gesture. An additional *neutral* class was also introduced.

Once a cropped data set was ready, tests were repeated using combinations of the six- and ten-angle feature sets and combined angle/coordinate sets. See figure 4.8 for an overview.

Cropping the input data had a significant effect on the performance of the kNN. The best results were still obtained with a six-angle feature set while going to ten angles or combining angles and the four coordinate points lowered performance. A final set of tests were then done where the zero-padding of input data was removed, but this made the results worse. See figure 4.9.

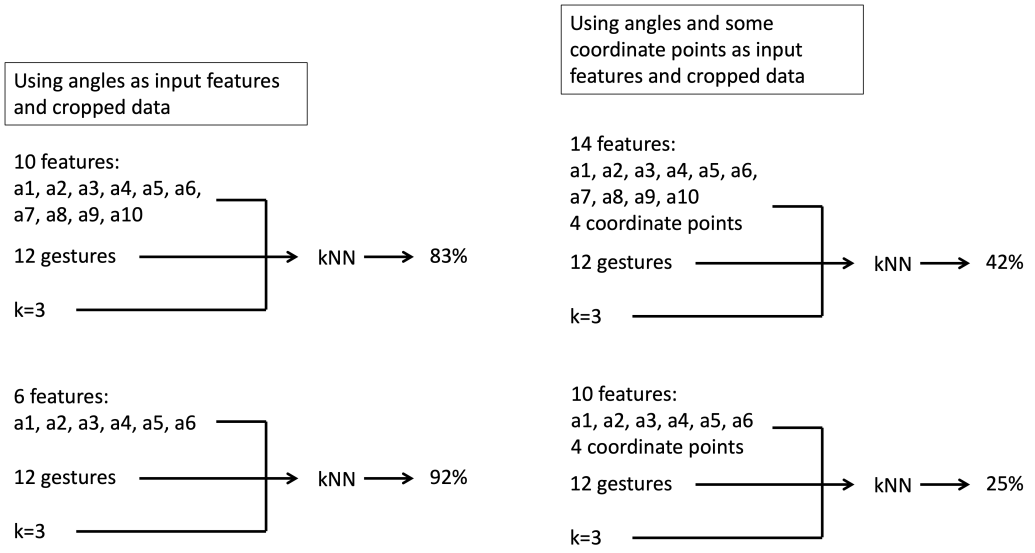


Figure 4.8: kNN accuracy when cropping data and varying the feature selection. The coordinate points are for the left and right hands.

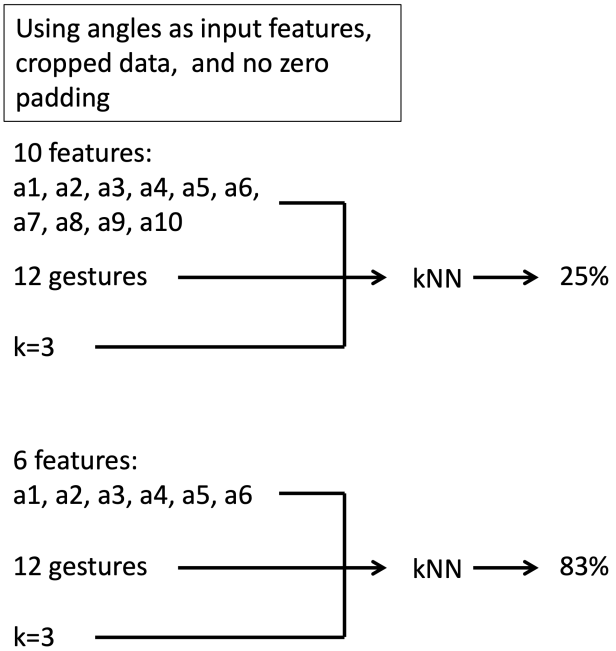


Figure 4.9: kNN accuracy on test set with cropped data, an additional *neutral* gesture class, zero padding removed and only angle features.

Learnings from the kNN separability testing The main finding from the kNN testing was that *going from 50 coordinate points to six angles as feature input significantly improved the separability of the gestures*. Restricting the feature set to the six angles a1, a2, a3, a4, a5, and a6 significantly improved the kNN model's ability to predict the correct class. Therefore, the six angle features will be used in the implementation of the LSTM model.

Another factor that clearly improved the kNN model's performance was cropping the input data to only contain the most distinctive pose of the gesture movement. Since each gesture started and ended in a neutral position, there was significant overlap between them. This should not be a problem when looking at the whole sequence, however, when looking at one frame at a time, it is impossible to know which gesture the frame represents if it is an overlapping frame.

4.6 Training an LSTM Classifier

Building on the results from the kNN tests, we decided to use the six-angle feature set that gave the best performance; a1, a2, a3, a4, a5, and a6 in figure 4.6a. Data was initially not cropped - each gesture's whole sequence of frames from start to finish was kept. The LSTM's other parameters are listed in table 4.4. The parameters were chosen based on previous research done by Noori et al. [59] and obtained from one of the authors (Benedikte Wallace).

Table 4.4: LSTM training parameters

Parameter	**Value**
Hidden units	256
Training batch size	32
Training epochs	200
Learning rate	1e-4
Dropout	0.2

For the first test with the LSTM, we used the same training data sets as for the kNN with zero padding, ensuring that frame sequences were of equal length. The LSTM performed with an accuracy of 63.63% on the validation set, significantly poorer than the kNN. This can be explained by the significantly larger number of labeled examples in the kNN set, as each frame in a gesture frame sequence is an example. To improve the LSTM's performance, we decided to increase the training data volume.

Adjusting the gesture set The original gesture set in figure 4.3 was modified by adding a neutral class and removing gesture G10, resulting in 11 classes. G10 (raise knees) was removed since we had seen in the OpenPose testing that it would struggle to recognize bending in knees when the person is facing the robot camera. Instead of correctly recognizing that the knees were making an angle, OpenPose would predict that the legs were getting longer and shorter as the knees bent.

4.6.1 Training, test, and validation data sets

One hundred examples of each of the 11 gestures were collected for the training set. An additional ten examples were recorded for the test set and ten for the validation set, totaling 1320 recorded gesture performances. The data were all gathered in the motion capture room. The person wore a white t-shirt and light blue pants, which contrasted well with the black background. We kept the lighting in the room at a bright level, and the distance between the person and the robot was at a range of 1.5 to 2.5 meters. The person was always facing the robot camera when performing the gestures. The data was processed the same way as for the data used with the kNN, by manually selecting all the frames making up the sequence of each gesture. The gesture sequences were of varying length, so they were made a uniform length of 84 frames by padding shorter sequences with -100. A padding value of -100 was chosen with the aim of the LSTM model learning to ignore the value as it was a value that the feature angles would never produce. The training, test, and validation sets were kept separate. The data set prepared for the training of the LSTM thus had 110,880 feature vectors grouped into 1320 full gesture examples.

4.6.2 Reducing latency

When running the system live, we need the robot to record the frames needed to classify a gesture, transmit them to the laptop over wifi, process the images through OpenPose, pass the feature vectors to the LSTM, and the model to return a predicted action with a latency low enough to be acceptable to a human operator, ideally no more than 1-2 seconds.

During the data capture phase, the frame rate achieved when processing training set images through OpenPose running on the laptop was 4.7 frames per second. Given that the same image resolution was kept during live operation, the minimum delay from image capture to prediction would be greater than this by some margin. If the gesture length was kept at 84 frames and

a whole gesture had to be processed before a prediction was returned, the minimum latency would be $84f/4.7fps = 17.9$ seconds.

Given the available hardware and software, the only way of getting fast enough predictions was then to *reduce the number of input images needed to make a prediction* - i.e. the model must predict reliably given much shorter sequences than the full 84 frames used until now. Achieving this, therefore, became a key design goal.

Optimizing the model for shorter input sequences An LSTM trained on data with a sequence length of 84 frames achieved an accuracy of 81.75% when given 84-frame gesture test sequences. We tested the model with shorter input frame sequence lengths and found that the performance was no better than random allocation to classes.

A significant part of the work on the implementation turned out to be optimizing the LSTM model to predict reliably on short frame sequences. In this section, we review the most salient tests and experiments done and the design changes implemented to finally arrive at a solution with good enough performance both time-wise and prediction accuracy to be used live. Table 4.5 gives an overview of the steps taken, and some are discussed separately below.

Table 4.5: Steps taken to optimize the LSTM for short input frame sequences

Action taken	**Outcome**
Hyper parameter tweaking and different padding strategies - padding the start versus at the end of a sequence	No improvement
Sliding window to generate multiple short sequences from full gesture sequences when training	Clear improvement with a 10 frame window width
Re-balancing the training set after introducing sliding window to get a similar number of short sequences for each gesture	Further improvement
Reducing the number of overlapping sequences by removing gesture G5 as it overlapped with G6 and cropping the sequences further	Accuracy dropped as the training data again became unbalanced
Re-balancing by generating synthetic training data for underrepresented gestures	Accuracy improved
Very close cropping of the full gesture sequences	Accuracy improved further, but gesture changes were now needed
Design changes where gestures were cropped down to their end pose, the gesture selection was modified and features tweaked by increasing the range of the feature angles.	Final accuracy of 96.59% achieved with a sequence length of only 8 frames

Sliding window. Instead of feeding the model the full, possibly padded, sequence of frames making up a gesture in a single sequence, only the frames currently *in the window* are used. The window is then progressed by one frame, and a new sequence is given to the model - see figure 4.10.

The full-length gesture sequences are broken up and each let us generate a number of shorter sequences, all labeled as the gesture. This gave us more training examples for each gesture; however, it also meant that more examples were created for gestures that initially had very long sequences than the gestures with fewer frames. Some of the gestures ended up having twice as many examples in the training set as the rest. Training with unbalanced data sets, where some gestures are over-represented, can skew the model to

predict the over-represented class more often as it learned that this class comes more often and is then more likely to be the correct prediction when two sequences are similar but belong to different gesture classes.

This problem was mitigated by cropping overlap between original sequences by removing gesture G5 as it was too similar to G6 and by generating synthetic data to add examples for underrepresented gestures.

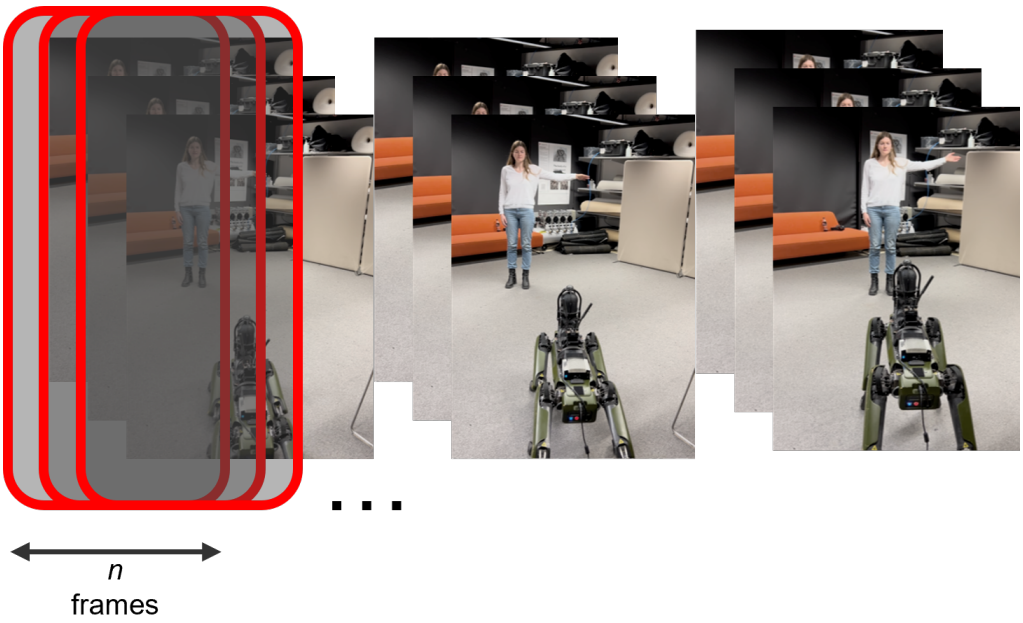


Figure 4.10: Sliding window protocol. The full input frame sequence is split into shorter sequences.

Synthetic data Using synthetic data lets us get more training examples without needing to record more sequences using the robot - a labor and time-consuming task. Synthetic data was only used in the training data set. By testing and validating using captured data, we could be confident that the measured accuracies would reflect real-world performance.

The data was created by first loading in an example frame (feature vector) from a gesture captured from non-synthetic data, then setting a value range for the feature angles we wanted to modify, and finally generating a chosen number of new feature vectors in a sequence, each frame a slight variation on the previous so that motion was simulated.

Changing the gesture selection To reduce the number of frames further, the gesture sequences were cropped down to the *main pose of each gesture*.

Once this was done, some of the existing gestures created issues. Gestures G3, G4, G6, and G9, for example, did not *have* obvious 'end poses' which the data could be cropped down to, and G8 and G1 could no longer be differentiated. We therefore decided to remove G3, G4, G6, G8, and G9 from the data set and introduce new gestures which *could* be cropped down to a single unique pose, these are shown in figure 4.11.

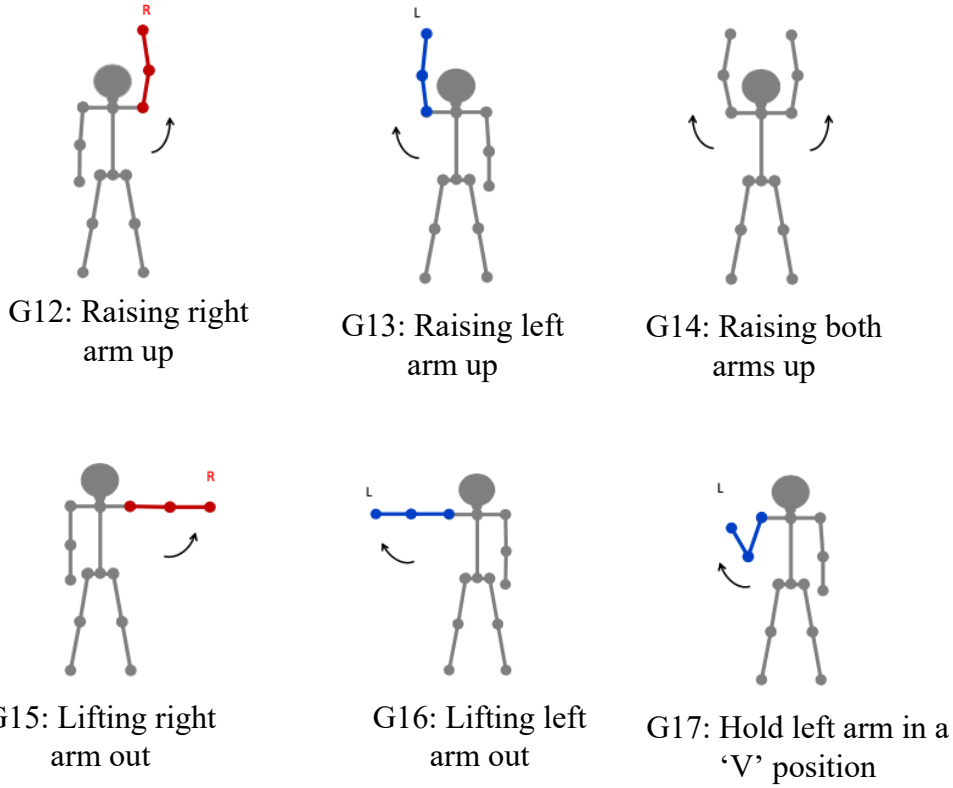


Figure 4.11: Six new gestures were designed with better discrimination between their *main poses*

Adjusting the value range of the angle features To make the gestures even more separable, we made adjustments to the way the feature angles were calculated. Up until now, the angles ranged between 0° and 180°. Restricting the angle values to this range meant that information about the direction of the vectors making the angles was being discarded. Therefore we changed the range to be from -180° to 180°.

Achieved accuracy The accuracy achieved on the test set was 96.59% with a sequence length of 8 frames (for both training and test).

4.6.3 Live testing the trained model with the robot

At this point, we had a model which performed well on the test sets. When we tested the trained model live in the MOCAP room it also performed well. The robot responded to most gestures within a second, while it sometimes struggled with gestures G1, G2, G15, and G16, which could take up to 5 seconds.

However, when we tested the model live in a less ideal room than MOCAP, the model performed poorly and barely managed to make any correct predictions. We recorded some of the sessions and investigated why the model suddenly failed.

As it turned out, OpenPose produced incorrect output leading the model to predict on input that was very different from what the person was actually doing.

The room in which the model failed had much more going on in the background than the MOCAP room. See figure 4.12 for a comparison between the two rooms. In contrast to MOCAP's mostly uniform background, this room had bright windows, black and white walls and ceilings, all of which affected OpenPose's performance. OpenPose seemed to struggle with poor contrast between the person and background.

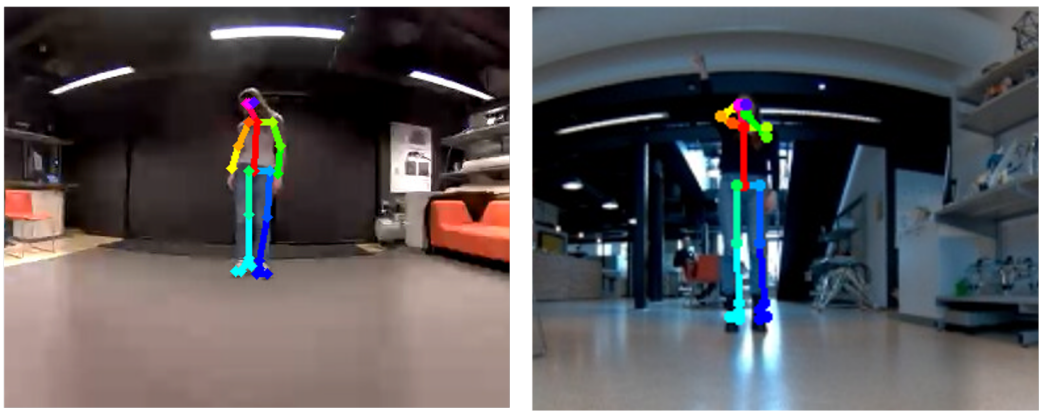


Figure 4.12: The MOCAP room (left) has a monotone background with fewer elements to confuse OpenPose than the indoor environment with similar background to the user experiment venue (right)

Since OpenPose could struggle when not in a suitable environment, such as the MOCAP room, changes had to be made to compensate. The ges-

tures which were to be used in the user experiment had to be very easy for OpenPose to detect and for the model to classify. Therefore, we removed the gestures which OpenPose struggled most with in the MOCAP room: G1, G2, G15, and G16. In order to have enough gestures to use when implementing all the actions with the robot, we added G8 back since G1, which had previously made difficulties when in the same set as G8, was now removed.

The last change we made was to increase the resolution of the images sent from the robot to OpenPose to the maximum available of 1870x470 pixels. This was because OpenPose performs worse on low-resolution images with little contrast between people and background or if the person is very small in the frame. Increasing the image size, however, would also increase the processing time for OpenPose, adding to the latency. The observed frame rate after increasing the image size fell to 3.6 fps.

4.6.4 Summary - LSTM development

Since feature extraction (OpenPose) from the image stream coming from the robot had to be done on a laptop, where the average frame rate was 4.7 fps for normal resolution images, and while ensuring that the robot would react to a gesture with acceptable latency, we had to look for ways to make the LSTM deliver predictions based on very short video sequences. This led us to break up the data into shorter sequences for training and operation. This again had consequences for data cropping, data balancing, and ultimately gesture selection. Finally, live testing with the robot led to further changes in the gesture set to compensate for OpenPose performance in non-ideal, real-world conditions. Final tweaks were also made to the feature value ranges and image resolution to avoid losing information.

The final set of gestures were very distinct and, in reality, poses rather than fluid movements. However, in return, the final LSTM delivered high accuracy - 96% on image sequences of 8 frames which, with the frame rate of 3.6 fps achieved with high-resolution images, would give us a calculated latency of 2.2 seconds.

4.7 Integrating the model with the robot

With a model ready, the main pieces to put in place before a user experiment could be done were:

- Decide on the set of commands we should be able to send to the robot and map these to gestures.

- Conclude on the frame sequence length to use when operating the robot.
- Implement the command interface and cater for errant output from OpenPose and subsequent LSTM predictions.
- Address the 'who to follow' problem when OpenPose detects more than one person in an input frame.

Mapping gestures to robot commands To be able to control the robot through an indoor or outdoor course (and be able to conduct a non-trivial user experiment), we decided we would need commands for: *rise up* - go from lying down to standing up, *lower down* - lie down when standing, *look at me* - turn to face the person, i.e. change direction from a standing up position, *take picture* - take a picture, *walk* - walk forward for as long as the gesture is held - then stop, and *walk stairs* - walk up stairs for as long as the gesture is held. The *stop* command would then be the same as 'no longer gesturing *walk*' (or *walk stairs*).

We had seven gestures at our disposal from the final testing with the LSTM model. We ended up not using gesture G11 *squat*, even though it was one of the gestures the trained LSTM model performed best on because the model would often incorrectly predict squat when the person was walking. The resulting mapping between gestures and robot actions is shown in table 4.6.

Table 4.6: Mapping of gestures to action commands from robot

Gesture	Action command
G12 - right arm up	Rise up
G13 - left arm up	Lower down
G8 - right arm V	Look at me
G17 - left arm V	Take picture
G7 - both arms out to the side	Walk
G14 - both arms up	Walk stairs

Deciding the final frame sequence length During model development, we achieved peak accuracy with a frame sequence length of 8. When testing live with a length of 5, we found that the actual performance was not noticeably worse. We, therefore, decided to use a 5-frame gesture sequence length. This was implemented using an *input buffer* of length 5 (room for five feature vectors). The input buffer contains feature vectors extracted from OpenPose output.

Given a steady stream of images from the robot to OpenPose, the input buffer would fill in 1.4 seconds at 3.6 fps. Once filled, the five frames would be sent to the LSTM to get a predicted gesture back. The input buffer would be updated each time OpenPose found a person in the newest frame. The oldest values would be removed from the buffer, and the feature angles calculated from the newest frame would be inserted into the end of the buffer (first in, first out). The LSTM would therefore produce a new prediction every time a frame came from OpenPose, once the buffer had initially filled up.

Filtering commands to the robot To mitigate against errors and fluctuations in the input to the model, a *filter buffer* of size five, containing the latest *predictions* the model had made, was implemented. The buffer would be refreshed every time the model made a new prediction, discarding the oldest and inserting the newest. If at least three out of the five predictions in the buffer were for the same gesture, then this was the gesture that would be chosen, and the robot would be sent the command mapped to the gesture.

To prevent the robot from performing the same action several times right after each other because the buffer still contained three or more predictions of the same gesture, a three-second time delay before any new command could be sent was imposed.

Minimum possible latency Given the buffer structure, the minimum latency between gesture and response happens when the input buffer (filled with feature vectors extracted from OpenPose) is pre-filled with *neutral* feature vectors and OpenPose generates a new non-neutral output. Since the buffer is full, it is sent to the LSTM, which predicts on the five frames. One non-neutral frame is enough for the LSTM to predict the non-neutral output. Since the filter buffer (filled with output from the LSTM) needs a minimum of three identical predictions to send a command, the *earliest* this can happen is after three OpenPose frames. The system's measured frame rate with the highest resolution images was 3.6 fps, so the minimum system latency is > 0.8 seconds. Figure 4.13 shows an example of how the two buffers work.

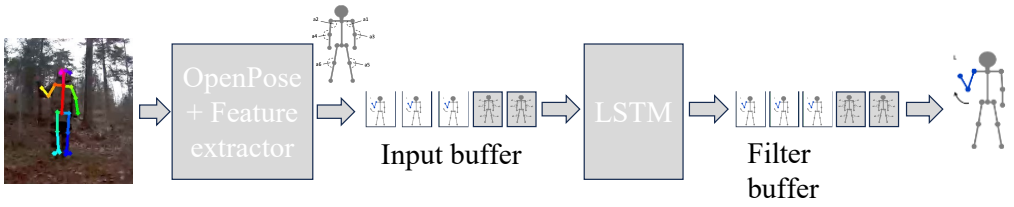


Figure 4.13: The LSTM is given five frames as soon as the input buffer is full. Once three predictions of the same gesture have been made by the LSTM, a command is issued to the robot. The minimum possible latency given a 3.6 fps frame rate is 0.8 seconds

Deciding who to follow Another challenge was knowing which person the robot should follow and receive commands from when multiple people were present in the frame. We decided that the robot would always follow the person with an OpenPose confidence score for the chest (point 1 in figure 4.5) and hip (point 8) above 25%, who was also closest to the center of the frame. The hip and neck confidence criteria were implemented because OpenPose would occasionally find 'people' in the walls and other objects in the room. See figure 4.14 for an illustration.

Figure 4.14: OpenPose occasionally found 'people' in the walls of the room - on the right in both the top and bottom images.

4.8 Summary

The operational system structure is depicted in figure 4.15, and the final gesture set in table 4.6.

The implementation process turned into an exploration that required multiple iterations of data collection, testing, and optimization. The choices made were constrained by available hardware and performance requirements.

The variable quality of input images and the limitations of the image recognition framework, OpenPose, led to choices of command gestures that were very distinct and, in reality, poses rather than fluid, meaningful gestures.

The kNN model-building showed us that a reduced feature set consisting of the six angles a1 - a6 in figure 4.6a led to better discrimination than using all angles available or the full set of 25 joint x- and y-image coordinates output from OpenPose. Filtering the predictions made by the LSTM model assisted in overcoming some of the issues with the output from OpenPose. In testing on the robot, a frame rate of 3.6 fps was achieved, and the solution showed reaction times between 1 and 3 seconds on average when no obvious technical issues were observed.

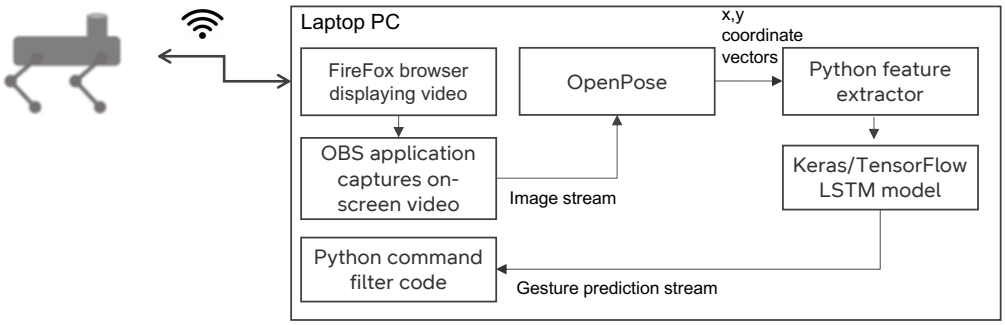


Figure 4.15: The operational system structure

Performance was acceptable, and all necessary robot control commands were available through gestures, setting the stage for the full user experiment described in the next chapter.

Chapter 5

User Experiment and Results

The previous chapter detailed the gesture-based robot control system and the development steps we went through to build it. In this chapter, we describe the live user experiment and discuss the results obtained with regard to technical performance, user experience with - and attitudes to - the solution.

The experiment was done to get data to help answer our research question of *what is preventing visual gesture recognition from being an efficient way of controlling mobile, legged robots in real-world use cases.*

5.1 Experiment setup

Eight test persons were tasked to operate the robot using gestures. The experiment was designed and piloted before it was conducted with the full set of test subjects.

The test persons would guide the robot through a path in an indoor environment which was different from the room used during development. Data was collected from the system, from participants answering questionnaires and by filming their interaction with the robot. The participants were given a brief introduction to the gestures and the course before the experiment started. A4-sized illustrations of the gestures and blue tape to show the course were visible on the floor to aid the test subjects. See figure 5.1.

Figure 5.1: A section of the experiment room seen through the robot's camera. Sheets with gesture explanations and tape to indicate the course can be seen on the floor

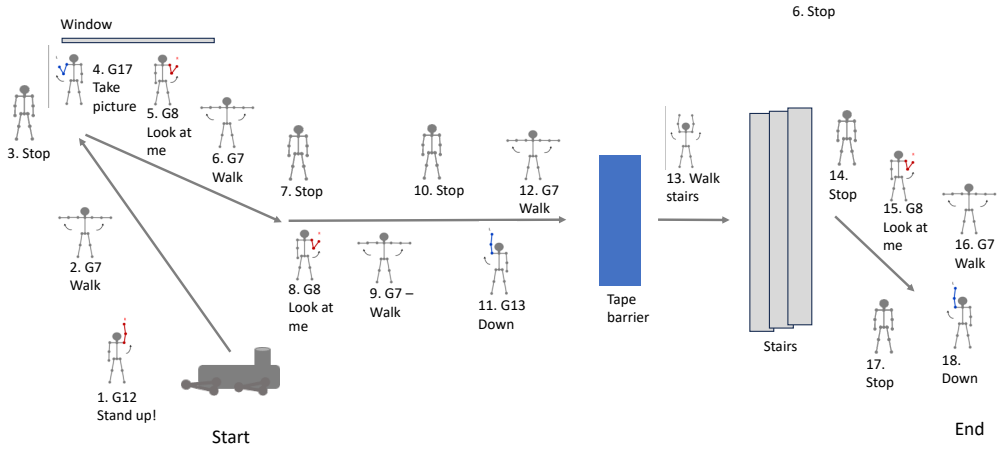


Figure 5.2: The course to be completed by test persons guiding the robot

The experiment had three parts:

1. Before interacting with the robot, the participants responded to four statements about their expectations of the experiment by grading their answers on a scale from *strongly agree* - *strongly disagree*. These statements are listed in table 5.1.

2. They then used gestures to guide the robot through a course, as depicted in figure 5.2. Starting from a lying down position, the robot was

to stand up and follow the test person's guidance to the corner, then stop. Here the robot was to rise to a stretched height position and take a picture out a window, then lower back to the normal standing height. Then the person would have the robot turn and walk over to a tape barrier which the robot would have to lower into a crouch height position and walk 'under' to pass. The robot would have to rise from crouch to normal height, then climb a set of stairs before lying down again.

3. Finally, after interacting with the robot, they were asked to grade six additional statements and answer four open-ended questions. See table 5.2 for a list of the statements and table 5.3 for the open-ended questions.

This setup allowed us to get insight into the participants' expectations before they worked with the robot and how they changed once the interaction was over. Open-ended questions were included in order to follow up on the graded questions and capture any additional observations the test persons might have. By asking test persons with no prior experience with the gesture control solution to guide the robot through a course, we would also be able to collect realistic system performance data.

Table 5.1: Statements the participants were asked to grade on a scale from *strongly agree* - *strongly disagree* before interacting with the robot.

Number	**Statement**
1	A fast response time from the robot after I have completed a command is very important to me
2	Having to repeat a command because the robot didn't register my movement/command or misinterpreted it as a different command is acceptable
3	A gesture-command system like this seems very useful to me
4	I expect this gesture-command system to work well on this robot

Two statements were identical before and after to allow some capture of changes in opinion caused by the interaction with the robot. In addition, pre-statement 3 captured much the same information as post-statement 6 as both were about the perceived usefulness of the solution.

Table 5.2: Statements the participants were asked to grade *strongly agree* - *strongly disagree* after interacting with the robot.

Number	Statement
1	The gesture-command system worked as I expected
2	A fast response time from the robot after I have completed a command is very important to me
3	Having to repeat a command because the robot didn't register my movement/command or misinterpreted it as a different command is acceptable
4	There were commands you wanted to give the robot which were not available
5	I felt uncomfortable while interacting with the robot
6	You see a gesture-command system like this being used in the real world, and expect to encounter it in industry in the near future (5+ years)

Table 5.3: Open-ended questions asked after the interaction with the robot.

Number	Question
1	What functionality would you want from a system like this? What improvements could be done?
2	If you answered 'strongly agree' or 'agree' on question 4. above, which commands are missing?
3	If you answered 'strongly agree' or 'agree' on question 5. above, please specify what instances made you feel uncomfortable
4	Please let us know whether you have any additional observations or input

A full list of the graded scores given by individual participants and their answers to open-ended questions are in the appendix. A summary of the user feedback can be found in section 5.4.2.

5.2 Participants and consent

The experiment was initially piloted by a person familiar with the thesis but not the experimental setup.

The eight main experiment participants were all current master's students in robotics and intelligent systems at the University of Oslo. They were given

a short introduction to the project and asked to sign a consent form to allow filming of the experiment. The participants have been anonymized in the data collection, and video recordings were deleted once this master thesis was completed.

5.3 Pilot

As time would be limited in the full experiment, it was decided to do a pilot run-through and use the learning to optimize the setup. The draft questionnaires were reviewed, the intended instruction was delivered, and the pilot test person led the robot through the intended course.

After the pilot, some changes were made:

- **Changes to the questionnaires**. A couple of questions were replaced with new formulations to make them clearer.
- **Compressed the instruction**. In the original plan, participants were to be given instructions in separate stages throughout the exercise. After the pilot, we concluded that it was better to deliver all the instructions up-front as this would save time and give participants the full picture immediately.
- **Gesture cue-cards**. Since the final set of gestures were not fully intuitive and enough training time for participants to memorize them was not available, It was decided that gesture memory aids had to be available. A4 sheets of paper with depictions of the gestures were therefore placed on the floor of the test venue.
- **Simplified the course**. In the pilot, a table which the robot needed to go under, was part of the course. The robot managed this, but it became clear that the horizontal space available without the risk of hitting the table legs was small and could cause problems during the experiment. We decided to remove the physical table and place a tape barrier on the floor instead.
- **Tape guide on the floor**. It became clear that remembering the intended path through the room and ensuring that all participants followed the same route could become a problem. It was therefore decided to draw a path on the floor using blue masking tape to help the participant.

5.4 The experiment

The experiment was carried out on a single day. Eight participants signed acceptance forms, graded pre-experiment statements, received instruction, guided the robot through the course using gestures, and finally graded post-experiment statements and answered questions.

It took nine hours to complete the user experiments. Each participant needed about 30 to 45 minutes, and the remaining time was spent doing the initial set-up, getting ready for each test person, and final clean-up.

All eight test subjects completed the course. Technical issues were observed, but these did not stop any of the participants from guiding the robot all the way through. Section 5.4.1 discusses the performance of the system and user issues, and section 5.4.2 the feedback given by the participants to the questionnaires.

5.4.1 System performance

A measure of system responsiveness during the experiment can be done by observing the time used, in number of seconds, from when the user performs a gesture until the robot takes action. This was calculated by dividing the number of frames processed before a reaction happened by the systems' 3.6 fps processing rate.

When user error or technical issues were observed, the robot did react, but only after a delay. The response would come because the user corrected the gesture, the wifi connection 'unfroze' and video transfer resumed, or the user changed position slightly to allow for a better image to be captured by the robot. Figure 5.3 illustrates the time elapsed from when the initial gesture is struck to the robot executing the intended command across the experiment.

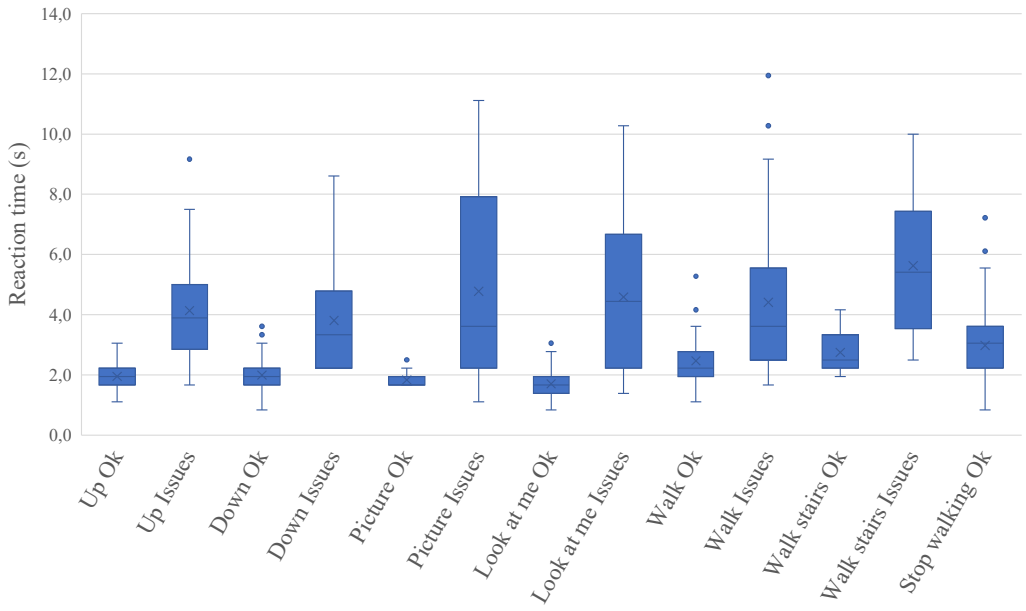


Figure 5.3: Number of seconds elapsed before the robot reacted after being shown a gesture with/without technical or user issues. *Stop walking* did not have sufficient data in the *user/tech issues* category to be plotted.

Performance when no technical or user issues observed When the user's gestures were correct, the robot camera fully captured the person, the wifi connection was stable, and the robot was not hindered by obstacles, then performance was consistent with a mean latency between 1.7 and 3.0 seconds. The longest observed delay, without a clear reason, was 7.2 seconds. See table 5.4 for an overview of the performance on the different commands.

Table 5.4: Latencies (seconds) in executing commands when no technical or user issues were observed.

Command	**Mean**	**Min**	**Max**
Up	2,0	1,1	3,1
Down	2,0	0,8	3,6
Picture	1,8	1,7	2,5
Look at me	1,7	0,8	3,1
Walk	2,5	1,1	5,3
Walk stairs	2,7	1,9	4,2
Stop walking	3,0	0,8	7,2

Technical and user issues Observed issues can be placed in 6 categories.

1. **Robot hindrance.** The Spot robot will avoid walking into obstacles or into cracks in the ground. Obstacles can be real but are sometimes 'false positives', such as the tape guide used in the experiment. The robot sometimes thought the tape was a tall object that it had to step around or lift its legs high to get over. This could alter the robot's course. At other times the robot had stopped beside the user and lost sight of them but remembered where the user used to be. When the user moved and gestured for the robot to walk again, it would alter its course to avoid hitting the user - which it believes is still standing in the way.

2. **Wrong pose.** This is a user error. The user has either made a pose the LSTM model is not trained to recognize, or the pose is illegal because of the mode the robot is in. For example, when the robot is lying down, it will not respond to any other command than *stand up*. Likewise, the robot would not *take picture* if it was not in *stretched mode*.

3. **Poor pose.** The user is performing a pose where the resulting angles are too far from the range expected by the LSTM model.

4. **User standing too close.** The user is standing too close to the robot. This can result in only part of the body being in the frame of the camera and OpenPose not getting images where all joints are visible.

5. **Poor image quality.** The person is too small in the images received by OpenPose (figure 5.4), or the contrast between the person in the images and the background is insufficient (figure 5.5). OpenPose produces erratic output, and it can take more frames than normal to get a stable prediction.

6. **WiFi lag or frozen frame.** In 11 percent of problematic cases, the robot's WiFi connection was observed to cause a lag between the user making a pose and OpenPose receiving an updated frame, causing a delay in the robot's reaction.

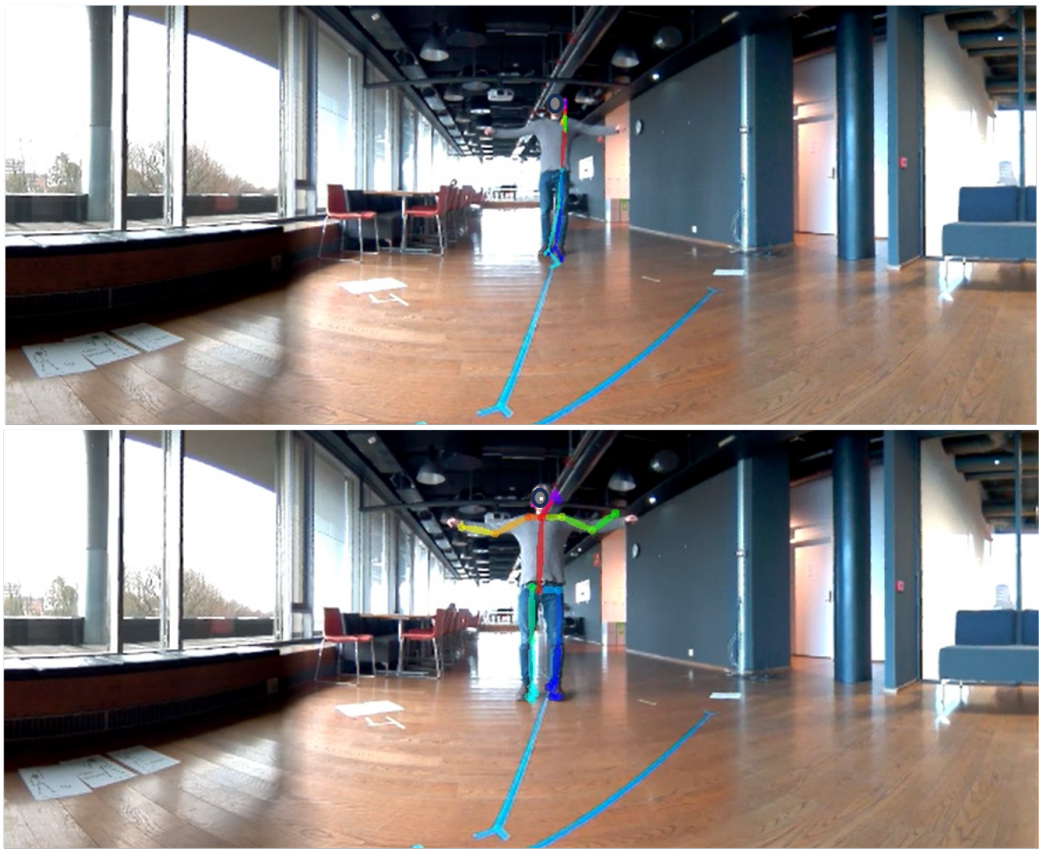


Figure 5.4: If users were more than 2.5 meters away from the robot, they would be too small in the image for OP to reliably map the joints (top). Once the user moved closer, OP performed well (bottom).

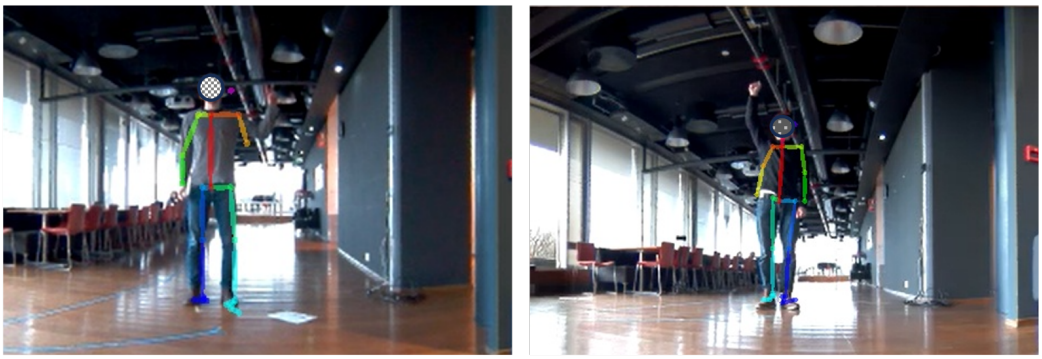


Figure 5.5: Two examples where the contrast between the users' arm and the background is poor and OP fails to identify all joints.

Figure 5.6 shows the observed counts of issues and their percentage dis-

tribution. Poor image quality is the largest category with 42% of all cases. About a third of the issues were caused by the user striking the wrong or a poor pose.

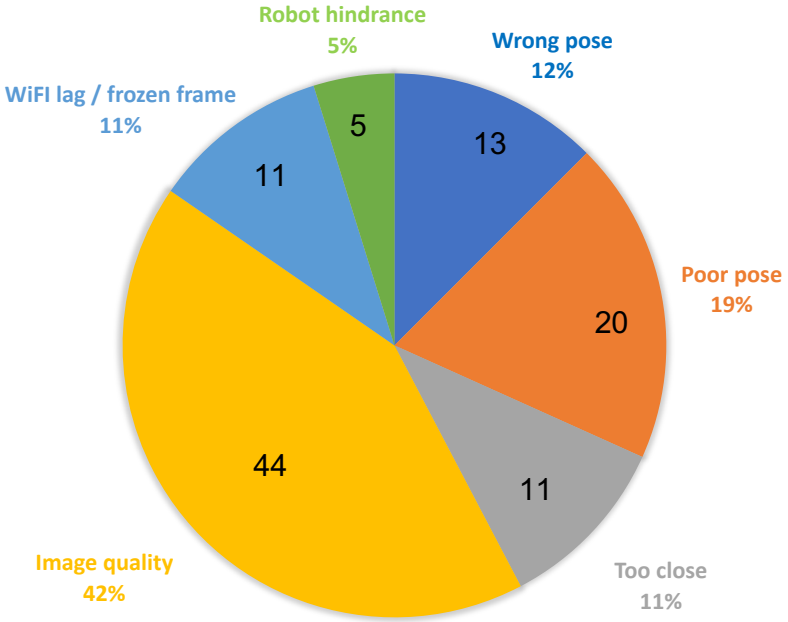


Figure 5.6: Distribution of observed issues during the experiment. Counts are indicated within the slices.

5.4.2 User experience

The users in the experiments were given four statements to rate before and six after their interaction with the robot. The statements were scaled, giving the user the option to select between scores; *strongly agree, agree, neutral, disagree,* and *strongly disagree*. Figure 5.7 sums up the number of responses in the *agree/strongly agree* and *disagree/strongly disagree* categories.

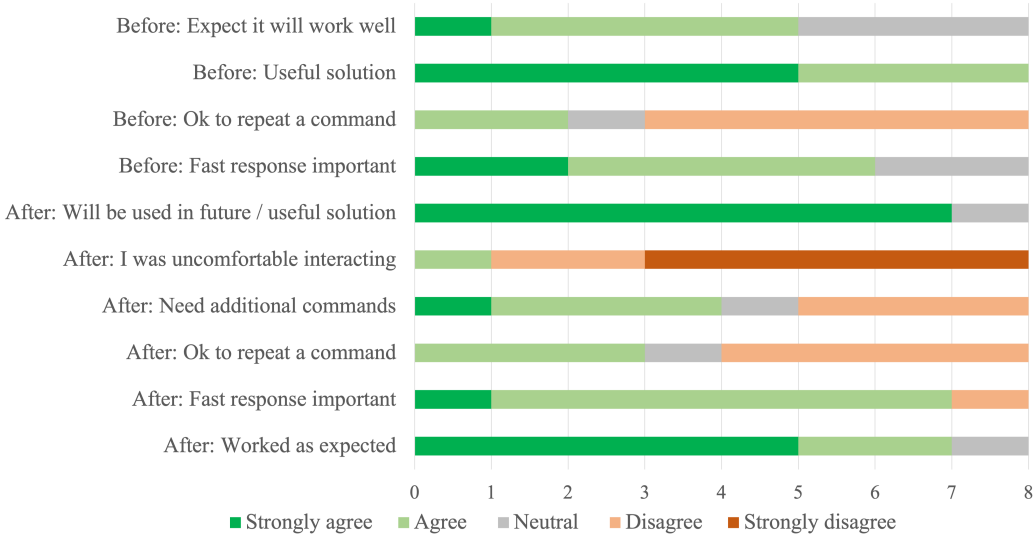


Figure 5.7: Distribution of participants' scores on statements.

Two of the statements were identical before and after the experiment in order to see if the users changed their minds after interacting with the robot.

- Statement 1 before and 2 after; *A fast response time from the robot after I have completed a command is very important to me.*
- Statement 2 before and 3 after; *Having to repeat a command because the robot didn't register my movement/command or misinterpreted it as a different command is acceptable.*

There was a slight movement in the scores in a *positive* direction, meaning *less negative* if a command has to be repeated and a *fast response* is less critical.

One statement addressed the participants opinions on the usefulness of a robot gesture control solution, with slightly different emphasis before and after, namely:

- Before (statement 3); *A gesture-command system like this seems very useful to me.*
- After (statement 6); *You see a gesture-command system like this being used in the real world, and expect to encounter it in industry in the near future (5+ years).*

The belief in the usefulness of the approach remained strong. One answer moved from *agree* to *neutral*, but all *agree* answers from before the experiment changed to *strongly agree*.

The changes in the scores were small for all three questions and should only be seen as an indication that the test persons' experience controlling the robot made them more positive. It is also interesting to note that all expected the solution to work well and reported after the test that it had. Only one of the eight found the experience uncomfortable.

User answers to open-ended questions The main takeaways from the open-ended questions are:

- The users wanted a *follow me gesture*, i.e. a better way of finetuning the direction of the robot while it was walking.
- The distance range the user could stand from the robot was restrictive. The operator had to be between 1.5 and 2.5 meters from the robot for it to react reliably to a gesture.
- There is room for improvements to the responsiveness of the system.

Summary of answers to question 1; *What functionality would you want from a system like this? What improvements could be done?*
Adding additional commands and personalizing the gesture-command relationship were mentioned; *"adding more commands for even more utility perhaps"* and *"... ability to change gesture-commands, so every user will be able to have own gestures."*. A command to change the trajectory of the robot was requested, and several mentioned improved accuracy and efficiency of the system; *"Faster response time.."* and *"General improvements such as faster and reliable detection."*

Summary of answers to question 2; *If you answered 'strongly agree' or 'agree' on question 4 above, which commands are missing?*
Most participants wanted a 'follow me' command, or an easier way to adjust which direction the robot was walking in; *"...it is hard to get it to walk in a straight line, being able to gesture small increments in the rotation would be beneficial"*. One participant wanted commands for more control of the robots movements, adding walk backwards, sideways, and an emergency stop command.

One participant wanted commands/functionality for actions which are purely for entertainment to make the robot come across as less machine-like and more friendly.

Summary of answers to question 3; *If you answered 'strongly agree' or 'agree' on question 5 above, please specify what instances made you uncomfortable.* A lack of control of the robot, when it was walking, was uncomfortable for one participant; *"Sometimes it felt like [the] robot would continue to move without stopping..."*. Others mentioned verbally that it was uncomfortable because they were worried the robot would injure itself as it did not stop immediately when the walk command ended.

Summary of answers to question 4; *Please let us know whether you have any additional observations or input.* The participants experienced the gesture-command system to work well; *"Gesture commands actually worked pretty well. I liked to have a more close interaction with a 'robot' through body language, rather than through 'programming' his actions."* and *"It worked really quite well, easy to use and interact."*. The desire for even better control / additional commands came through in this question as well; *"... being able to turn it without standing directly in front of it would be beneficial."*

5.5 Summary

In this chapter, we described the technical performance and user experience results from a live experiment where eight test persons were asked to guide a quadruped robot through a course using hand gestures. All test persons were able to complete the exercise and found the system useful. They also had clear opinions on areas where it can be improved.

The next chapter will discuss the findings and their implications for the thesis research question.

Chapter 6

Discussion

This chapter explores the results with a view to answering the research question posed in section 1.2; *What is preventing visual gesture recognition from being an efficient way of controlling mobile, legged robots in real-world use cases?*

Multiple factors decide whether a technology is adopted or *diffused* in society [67]. The *relative advantage* to alternatives, e.g. cost, or having to carry a controller or a sensor and make sure it is charged in this thesis' case, *complexity of the technology* - is it easy to implement and use, *compatibility with 'social norms'* and *observability*. It is outside the scope of this thesis to discuss all contributing factors and the focus is on *technological readiness* for use in the real world and *user acceptance* or *attitudes* to the approach.

We start by discussing user acceptance in section 6.1 and technology readiness in section 6.2. Challenges and limitations of the study are raised in section 6.3.

6.1 User acceptance

The user questionnaires tell us that the test group had positive expectations to the solution before starting the experiment. All the participants were able to complete the course with the robot, although not without encountering issues, technical, user faults, or both. Despite this, their initial attitude to the system was kept or slightly improved after the experiments, see figure 5.7. This supports claims made by other papers, that *users are generally positive to using gesture-based interactions with robots* [68, 69].

Although positively inclined, many of the participants wanted improvements to the system, including finer control of the robot's movements, a faster response time, and a greater range in distance to command the robot

from.

Response times must improve Common feedback from the users was that the response time of the system needed to improve. In figure 5.7 from chapter 5.6, the majority of the users answered *agree* to a fast response time being important, while only two answered *neutral*.

Similarly, the user's answers to whether it was acceptable to repeat a gesture if the robot did not respond correctly changed in a more positive direction after the experiment, with one of the participants changing their answer from disagreeing to agreeing to it being acceptable to have to repeat a gesture.

This indicates that after interacting with the robot, some of the participants got less strict in their requirements for a correct response and a fast response time, and that the system was good enough or had enough other positive aspects for some of them to relax their requirements. However, *low latency in the response to a gesture is important*, and a clear feedback for improvements to the system.

A long response time could also leave a user feeling that they lack control of the robot, especially if the robot would not stop immediately when the user asked it to. This could make users uncomfortable and disinclined to operate the robot using visual gestures. Our experiment did not show such an effect, but it is likely that it would have been an issue in a real-world application.

Command range and colors on clothing Feedback from the users was that they wanted the command range to be larger so that they could gesture to the robot from further away. Due to the limitations of the system, users had to be between 1.5 and 2.5 meters from the robot when performing gestures.

Several of the participants wore shirts with colors that were very similar to the background. This did not work well as OP performs poorer when the contrast between the person and the background is low. The time it took before the robot responded increased, and this impacted their experience.

The current constraints on distance between robot and operator, and that the system is sensitive to the environment it is used in, means that unless technical solutions can be found *there will be use cases involving non-controlled environments where the system will perform poorly or not at all.*

Gesture and robot action selection The designed gestures were not very intuitive in terms of the action the robot would respond with. The users

would at times mix up left and right when performing gestures, however, none complained about or mentioned the choice of gestures as a big problem. One user did say they would have liked to have the opportunity to choose for themselves which gesture mapped to which robot action. Strazdas et al. [70], included this option in their study, where the users had to improve the gesture system by choosing which gesture mapped to which action command. They found that some gestures were often chosen for certain commands while others were more varying, *suggesting a need for a customizable interface* in order to fulfill user expectations.

Many of the users suggested adding more gestures to enable finer control of the robot. Specifically, a *follow me* gesture was mentioned. Though entirely possible to implement, it would also mean more gestures for users to remember. If the gestures are not very intuitive, this increases the risk that an operator would send unintended commands to the robot. *The choice of gestures should thus be as intuitive as possible, and a tradeoff between the number of gestures and the ability to fine-tune the robot control must be found.*

Summary - are user attitudes limiting adoption?

While keeping in mind the possible bias present in a test group consisting of only master students in robotics, the reported experience was positive, and the feedback given on improvement areas is all possible to achieve. *There are thus no clear reasons why visual gestures should not be an acceptable approach to robot control based on the user acceptance aspect of our experiment.*

The observed challenges with restrictions on the distance to the robot and requirements to visual contrast between the user and the background mean that deployment in real-world use cases will have limitations unless technological progress can solve them.

The experiment did end up using non-intuitive gestures as processing limitations forced a reduction in the number of frames in a sequence that could be processed in a reasonable time, and this is an area for improvement.

6.2 Technology readiness

In the previous chapters, we identified technology-related issues which impact the usability of the gesture-based robot control system, specifically:

1. Limiting the frame sequence length to reduce processing times means gestures may become less intuitive, and the barriers to user adoption increase.

2. With multiple people in the image, deciding who to 'follow' for gesture interpretation will be a challenge.

3. Wifi transmission lag problems were observed in the user experiments.

4. If the robot controller is not fully in the robot's camera frame - e.g. the person's upper torso is clipped - OpenPose will deliver zero-values for all upper body joints, the Python script will use the previous frames' joint coordinates, and the LSTM will not be able to predict correctly.

5. If the person is too small in the picture, OpenPose will generate erroneous output.

6. Poor lighting reduces contrast and makes OpenPose struggle to identify the person.

7. Fog, heavy rain, or snowfall is likely to obstruct a person in the same way as trees, furniture, or other objects placed between the robot and the person.

Rich and intuitive gestures. The decision to go from 'rich and intuitive' gestures to very distinct end poses was driven by the frame rate achieved. The frame rate was determined by the rate of the video transmitted from the robot to the laptop and by the speed of the processing on the PC (OpenPose and the LSTM).

Tests showed that the whole system had a frame rate of 3.6 fps, or 0.27 s/f, when given high-resolution images. From table 5.4, we have that the lowest *observed* latencies were as low as 0.8 seconds. From the discussion in section 4.7 we have that the lowest achievable system latency is a number greater than 0.8s and that this happens when OpenPose only needs to process three input images before a command is issued. From this, we can infer that the *dominant contributor to system latency was OpenPose.*

Significant research is going into building Human Pose Estimation solutions [48], and performance improvements to OpenPose have been proposed. A lightweight OpenPose for real-time use has been created [71]. Frame rates up to 26 fps with some loss of accuracy were achieved running on a CPU-only machine and using the COCO model rather than Body_25 used in this thesis.

OpenPose reports that the BODY_25 model is five times slower than the COCO model on a CPU, however, 40% faster on a GPU. While the fps is greater for the COCO model and could allow for 'richer' gestures as it would process more frames when run on a CPU, it is also less accurate than the

BODY_25 model and, therefore, might not improve the overall performance of the gesture control system.

Computational efficiency is highlighted as a remaining challenge for 2D pose estimation in [48], making deployment of OpenPose and similar solutions on *edge devices*, such as a mobile robot, challenging. Until the processing power available on robots or the computational efficiency of Human Pose Estimation systems improve, *faster frame rates and the ability to have richer gestures will depend on access to larger computational resources off the robot.* This puts constraints on the usage scenarios that are possible, as it means that *a reliable wireless connection with sufficient capacity must be available.*

Deciding who to follow The solution adopted in the system - to follow the person closest to the center of the image given a sufficient confidence score from OpenPose - is challenging in real-world situations as it puts another requirement on the person controlling the robot. There may also be cases where the robot is not facing the person and it is impossible to 'get in front of' it; for example it may be standing close to and facing a fence, detecting and following people more centrally in the frame on the other side of the fence, rather than the operator. A better approach is needed for the system to be robust in real-world scenarios.

Since OpenPose does not track people from frame to frame - it works on single images - this must be solved after OpenPose has classified all people in a frame. Imposing a rule that only one person can give commands, and so avoid conflicts, is one approach but vulnerable to situations with 'accidental gestures', so not practical. Tracking all identified persons between frames by assuming a maximum displacement and then finding the one most likely to be the controller could be another solution. A similar approach is discussed in [72], where a 93% success rate in maintaining the id of pupils in a classroom setting across videos was achieved.

Communication problems causing lagging behavior As long as the robot is controlled remotely, communication is an issue. This is the case in current deployments of the robot Spot, in which Boston Dynamics have implemented default behavior for when contact is lost. The robot can either stop and shut down or return to the starting point on its own. For our purposes, we would lose the ability to steer the robot, which is not desirable.

One of the technological/system issues shown in the results was frozen input image frames due to wifi lag. Wifi lag increases the response latency of the system. Wifi uses a public spectrum and can suffer from interference if there are many other networks nearby. The range of a wifi signal is also

limited as wifi is limited in the transmission power it can use. Current generation wifi (IEEE 802.11a/g and IEEE 802.11ac) have outdoor ranges of around 100 meters, and indoor ranges well below that [73].

Alternatives exist to WiFi, in particular cellular. Spot can mount a 5G radio and connect to 5G mobile networks. This may solve the problem of distance between the robot and the computer running OpenPose and the LSTM model, but only for use cases where there is 5G coverage. Indoors cellular coverage is often a problem, as well as 5G coverage in remote areas. This is illustrated in figure 6.1.

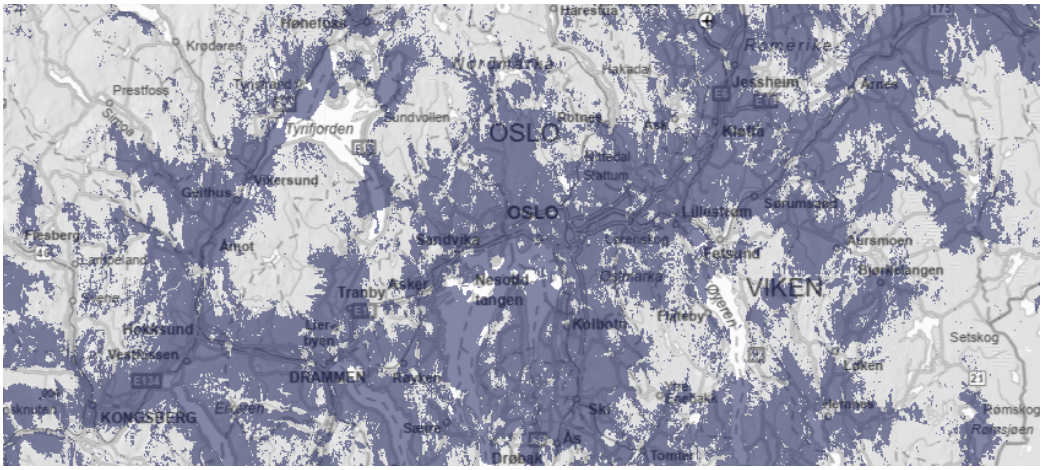


Figure 6.1: Telenor 5G coverage around Oslo in May 2023. Dark areas have outdoor coverage. Source: Telenor website.

We have argued that doing the processing on the robot will not give us improved frame rates. It is possible to mount an extra processing unit (NVIDIA Jetson Xavier NX) on Spot. This computer is less powerful than the PC which was used, however, and would not solve the problem (see section 4.1 and [74]). We are, therefore, dependent on doing the processing remotely from the robot, and this means that either the computer doing the processing must be close enough to the robot for it to connect to the robot's wifi network or that public cellular coverage is available for the location. *Situations where the PC cannot be close - e.g. no available electricity - or there is no 5G coverage - e.g. in remote areas or in some indoor locations - become problematical* .

Visibility of the user. 55% of the issues observed during the full-scale user experiment are attributable to *image quality* and *the user standing too close*. The user is either too small in the image because they were too far

away from the robot, the user is clipped because they are too close to the robot camera and therefore partially out of frame, or the contrast between the person and the background is insufficient. All of these situations would cause OpenPose to fail.

The camera used in the experiment is the best available from Boston Dynamics for use with Spot (see section 3.1).

The use of both an infrared and a visual range camera to create a combined prediction is a possible solution to poor lighting or confusing backgrounds and possibly fog or snowy conditions, as OpenPose has functionality for infrared image input. Such a combination has been shown to be resilient to a large range of low-visibility conditions [75]. However, combining two different camera inputs could make it even more challenging to maintain a high frame rate and low latency.

Using a camera with better vertical resolution and range can solve the too-small and clipped problem but would increase the processing load on OpenPose.

Greater processing resources connected to the robot might allow us to address the distance from the robot problem and background confusion issue by adding different or better cameras. It will not solve the problem where the user is partly hidden from view by objects or other people, which Zheng et al. [48] remarks as an area for further research.

Summary - technological barriers to adoption

We found that the system works well under *optimal conditions*; enough processing power, connectivity with sufficient capacity, good lighting, and good contrast between users and backgrounds, and when the user was alone or stayed front and center to the robot and between 1.5 and 2.5 meters from it. When some or more of these conditions are not satisfied, technical shortcomings become apparent. Of the identified issues, deciding who to follow has been addressed in other research, and solutions that are good enough to work in practice can probably be found. Adding processing power can solve the latency problem and allow for richer gestures. Adding processing and additional cameras may help mitigate the range and background contrast problems, but all solutions requiring more processing power depend on *reliable connectivity*, which is not a given, and a key limiting factor in many real-world scenarios. Current generation human pose estimation systems, exemplified by OpenPose, also struggle when people are partly hidden from view, and this is an area of ongoing research.

6.3 Limitations and Challenges

The research question asks about the application of a gesture control system in the real world. In order to draw conclusions from the tests, we need to conclude to what extent the tests *are* representative of real-world use cases. Three challenges are apparent; the lack of diversity in the test group, the venue used for the experiment, and the controlled nature of the experiment.

The test group The test participants were all robotics master students, which can have contributed to their positive mindset going into and out of the experiment, as they have an interest and maybe experience in working with and interacting with robots. It is not certain that a wider, more diverse group would have responded to the questionnaires the same way. They were also all young and fit and, in that respect, not representative of the general population. It is possible that the gesture performances would have been different in a more diverse group.

The experiment venue The experiment was carried out in a different and more 'visually busy' room than the MOCAP room, where training data was obtained. The course the users had to lead the robot through exposed it to different background and lighting conditions. It is still possible that experiments done outdoors in different locations and weather conditions would have impacted the quality of the images sent to OpenPose in a way we did not manage to capture indoors.

The controlled nature of the experiment The users were trained and guided with a stripe of tape on the floor that laid out the route to follow, and they were assisted with sheets of paper depicting the gestures and what commands they would send to the robot. The experiment was thus rigidly scripted. The real world is seldom like that. It is possible that a different approach with, for example, some participants receiving guidance and others not, or users being allowed to chart their own course with no hints as to what gesture to use at each juncture would have changed their perception as to how well the system worked.

Chapter 7

Conclusion

The goal of the thesis was to investigate *What is preventing visual gesture recognition from being an efficient way of controlling mobile, legged robots in real-world use cases* with a focus on user attitudes to such a system and technological barriers. This was done by developing a system for gesture control of a quadruped robot and then conducting a live experiment with multiple users in an environment different from the training location.

We found that user attitudes to gesture control were positive, indicating *few or no barriers to user acceptance of the approach.* On the other hand, we also identified *clear technology readiness issues which need to be addressed before the approach can be widely applied in real-world use cases.*

For the users, the challenges manifested as variable and, at times, long delays in command execution. The root cause is the current state of the human pose estimation system OpenPose. OpenPose needs 'near ideal' image quality with the controller clearly visible, unobstructed, and large enough in the image, with good contrast to the background, to produce reliable output. This can not be guaranteed in all real-world situations. OpenPose is also computationally expensive and has to be run on a computer off the robot. Possible mitigations to the image quality problems, such as increased image resolution and a faster frame rate, would increase the processing load further. The robot then has to be connected wirelessly at all times, and the connection was seen to cause issues. Wireless coverage is also not guaranteed in non-controlled environments.

We demonstrated that there is user acceptance for a gesture-based approach to control mobile, legged robots and that a system can be built that performs in controlled settings. We also found that the approach is not ready for wide real-world use and that improvements to human pose estimation solutions will be needed before wide deployment can happen.

Chapter 8

Future Work

More diverse user demographic and the effect of multiple interactions A limitation of the study is that all the participants in the user-case experiment were robotics students, which could have led to bias in the user attitude measurements. Other demographics, such as factory workers, might feel that their livelihood is threatened by the introduction of robots and have very different attitudes. Maurtua et al. [68] found that users with different industrial backgrounds held the opinion that introducing robots into the factory would have a negative effect on the number of jobs available for human workers but otherwise were positive to the use of robots.

It could be interesting to combine a larger and more diverse user group with the implementation method of Kim et al. [76], who had the users repeat their interactions with the robot resulting in a more positive attitude towards the robot after each repetition. What attitudes to gesture control of mobile robots would the more diverse group have, and how would this develop over time as they got experience working with the robot? More insight into the user acceptance question could be derived.

User acceptance in more challenging scenarios The activities the users were asked to complete in the experiment tested simple tasks one might come across in daily use. It would be interesting to test the robot and collect user feedback when a more challenging task is attempted, one that cannot be performed without the assistance of the robot, as proposed as future work by Maurtua et al. [68].

Tracking the controlling user when multiple people are present We mitigated the problem of deciding who the robot should follow by always selecting the center-most person in the frame.

In real-world scenarios, it must be expected that multiple people are visible and that imposing a 'front and center' rule will be impractical. A tracking system that works across individual image frames is therefore necessary. In Hur et al.s [72] work, individuals were tracked across frames by post-processing the output from OpenPose, and a 93% tracking accuracy was achieved. However, improvements are still needed, such as tracking over longer periods of time to re-identify people when tracking fails.

Not having to face the robot when gesturing A restriction imposed on the robot controller was that they had to face the robot when performing a gesture. Classification of actions from multiple angles has been successfully achieved in lab conditions [59] and could be a user-friendly development of the system for real-world use.

Human-Pose Estimator improvements Existing human-pose estimators are often trained on high-resolution images or videos, which can lead to poor estimation on low-resolution input, as seen in this thesis with OpenPose. A development in HPEs could be to include low-resolution along with high-resolution images by leveraging the contrastive learning scheme proposed by Chen et al. [77].

Neural Architecture Search (NAS) can be used to find efficient HPE network architectures to reduce computational costs [78], which was one of the main technological challenges in this thesis. Multi-objective NAS is an interesting further development for when multiple objectives, such as latency, energy consumption, and accuracy, must be taken into consideration and should be looked at further.

Do the processing onboard the robot Success in reducing the computational cost of doing HPE as discussed above, could make it feasible to implement the gesture recognition onboard the the robot. This would make continuous wireless connectivity less critical and improve the usability of the approach.

Bibliography

[1] Justin Carpentier and Pierre-Brice Wieber. "Recent progress in legged robots locomotion control." In: *Current Robotics Reports* 2.3 (2021), pp. 231–238.

[2] Boston Dynamics. *Spot technical specification.* URL: `https://support.bostondynamics.com/s/article/Robot-specifications` (visited on 05/02/2023).

[3] Thomas B Sheridan. "Human–robot interaction: status and challenges." In: *Human factors* 58.4 (2016), pp. 525–532.

[4] Christoph Bartneck et al. *Human-Robot Interaction. An Introduction.* Cambridge University Press, 2020. ISBN: 9781108735407.

[5] Pedro Neto et al. "Gesture-based human-robot interaction for human assistance in manufacturing." In: *The International Journal of Advanced Manufacturing Technology* 101.1 (2019), pp. 119–135.

[6] Michael T Wolf et al. "Gesture-based robot control with variable autonomy from the JPL BioSleeve." In: *2013 IEEE International Conference on Robotics and Automation.* IEEE. 2013, pp. 1160–1165.

[7] Jawad Nagi et al. "Max-pooling convolutional neural networks for vision-based hand gesture recognition." In: *2011 IEEE international conference on signal and image processing applications (ICSIPA).* IEEE. 2011, pp. 342–347.

[8] Michael Van den Bergh et al. "Real-time 3D hand gesture interaction with a robot for understanding directions from humans." In: *2011 RO-MAN.* 2011, pp. 357–362. DOI: `10.1109/ROMAN.2011.6005195`.

[9] Mikael Uimonen. "Multimodal interface for interactive cooperation with quadruped robots." English. Master's thesis. Aalto University. School of Science, 2022, p. 73. URL: `http://urn.fi/URN:NBN:fi:aalto-202212187146`.

[10] Jim Tørresen. "A Review of Future and Ethical Perspectives of Robotics and AI." In: *Frontiers in Robotics and AI* 4 (2018). ISSN: 2296-9144. DOI: 10.3389/frobt.2017.00075. URL: https://www.frontiersin.org/articles/10.3389/frobt.2017.00075.

[11] Eleanor Bird et al. *The ethics of artificial intelligence: Issues and initiatives*. DOI: 10.2861/6644.

[12] Leila Scola. *AI and the Ethics of Energy Efficiency*. URL: https://www.scu.edu/environmental-ethics/resources/ai-and-the-ethics-of-energy-efficiency/ (visited on 05/13/2023).

[13] IEA. *Norway 2022 Energy policy Review*. 2022. URL: https://www.iea.org/reports/norway-2022.

[14] Peter Flach. *Machine Learning. The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012. ISBN: 9781107422223.

[15] Adelchi Azzalani and Bruno Scarpa. *Data Analysis and Data Mining. An Introduction*. Oxford University Press, 2012, pp. 9–10. ISBN: 9780199767106.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *nature* 521.7553 (2015), pp. 436–444.

[17] OpenAI. *OpenAI website*. URL: https://openai.com/ (visited on 05/14/2023).

[18] OpenAI (2023). *GPT-4 Technical Report*. URL: https://cdn.openai.com/papers/gpt-4.pdf (visited on 05/15/2023).

[19] Martijn van Otterlo and Marco Wiering. "Reinforcement Learning and Markov Decision Processes." In: *Reinforcement Learning: State-of-the-Art*. Ed. by Marco Wiering and Martijn van Otterlo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–42. ISBN: 978-3-642-27645-3. DOI: 10.1007/978-3-642-27645-3_1. URL: https://doi.org/10.1007/978-3-642-27645-3_1.

[20] Spyros Gidaris and Nikos Komodakis. "Object detection via a multi-region and semantic segmentation-aware cnn model." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1134–1142.

[21] Reagan L Galvez et al. "Object detection using convolutional neural networks." In: *TENCON 2018-2018 IEEE Region 10 Conference*. IEEE. 2018, pp. 2023–2027. DOI: 10.1109/TENCON.2018.8650517.

[22] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." In: *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee. 2013, pp. 6645–6649. ISBN: 978-1-4799-0356-6. DOI: 10.1109/ICASSP.2013.6638947.

[23] Yajie Miao, Mohammad Gowayyed, and Florian Metze. "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding." In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE. 2015, pp. 167–174. DOI: 10.1109/ASRU.2015.7404790.

[24] Jiuxiang Gu et al. "Recent advances in convolutional neural networks." In: *Pattern recognition* 77 (2018), pp. 354–377. DOI: https://doi.org/10.1016/j.patcog.2017.10.013. URL: https://www.sciencedirect.com/science/article/pii/S0031320317304120.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[26] Hojjat Salehinejad et al. "Recent advances in recurrent neural networks." In: *arXiv preprint arXiv:1801.01078* (2017).

[27] Sepp Hochreiter. "Recurrent neural net learning and vanishing gradient." In: *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems* 6.2 (1998), pp. 107–116.

[28] IBM. *What are recurrent neural networks*. URL: https://www.ibm.com/topics/recurrent-neural-networks (visited on 04/12/2023).

[29] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[30] Cetin Kaya Koç. "Analysis of sliding window techniques for exponentiation." In: *Computers Mathematics with Applications* 30.10 (1995), pp. 17–24. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(95)00153-P.

[31] The Editors of Encyclopaedia Britannica. *Robotics*. URL: https://www.britannica.com/technology/robotics (visited on 05/25/2023).

[32] The Editors of the Cambridge Dictionary. *Robotics, English meaning*. URL: https://dictionary.cambridge.org/dictionary/english/robotics (visited on 05/25/2023).

[33] Signe Redfield. "A definition for robotics as an academic discipline." In: *Nature Machine Intelligence* 1.6 (2019), pp. 263–264.

[34] Selma Sabanovic, Marek P Michalowski, and Reid Simmons. "Robots in the wild: Observing human-robot social interaction outside the lab." In: *9th IEEE International Workshop on Advanced Motion Control, 2006.* IEEE. 2006, pp. 596–601.

[35] Yingfeng Chen et al. "Robots serve humans in public places— KeJia robot as a shopping assistant." In: *International Journal of Advanced Robotic Systems* 14.3 (May 2017). DOI: 10.1177/1729881417703569.

[36] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. "A gesture based interface for human-robot interaction." In: *Autonomous Robots* 9.2 (2000), pp. 151–173.

[37] Paul Doliotis et al. "Comparing gesture recognition accuracy using color and depth information." In: *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments.* Association for Computing Machinery, 2011. ISBN: 978-1-4503-0772-7.

[38] Hee-Deok Yang, A-Yeon Park, and Seong-Whan Lee. "Gesture spotting and recognition for human–robot interaction." In: *IEEE Transactions on robotics* 23.2 (2007), pp. 256–270.

[39] Fotini Patrona, Ioannis Mademlis, and Ioannis Pitas. "Self-Supervised Convolutional Neural Networks for Fast Gesture Recognition in Human-Robot Interaction." In: *2021 10th International Conference on Information and Automation for Sustainability (ICIAfS).* IEEE. 2021, pp. 88–93.

[40] Yugo Katsuki, Yuji Yamakawa, and Masatoshi Ishikawa. "High-speed human/robot hand interaction system." In: *Proceedings of the tenth annual acm/ieee international conference on human-robot interaction extended abstracts.* 2015, pp. 117–118.

[41] Elena Nazarova et al. "HyperPalm: DNN-based hand gesture recognition interface for intelligent communication with quadruped robot in 3D space." In: *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* IEEE. 2022, pp. 2043–2048.

[42] Alessandro Gasparetto and Lorenzo Scalera. "From the Unimate to the Delta robot: the early decades of Industrial Robotics." In: *Explorations in the History and Heritage of Machines and Mechanisms: Proceedings of the 2018 HMM IFToMM Symposium on History of Machines and*

Mechanisms. Springer. 2019, pp. 284–295. DOI: 10.1007/978-3-030-03538-9_23.

[43] International Federation of Robotics. *World Robotics 2021 Report*. URL: https://ifr.org/ifr-press-releases/news/robot-sales-rise-again#downloads (visited on 05/25/2023).

[44] Hongyi Liu and Lihui Wang. "Gesture recognition for human-robot collaboration: A review." In: *International Journal of Industrial Ergonomics* 68 (2018), pp. 355–367.

[45] Linn Danielsen Evjemo et al. "Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories." In: *Current Robotics Reports* 1 (2 2020), pp. 35–41. DOI: 10.1007/s43154-020-00006-5.

[46] Julia Berg et al. "Human-Robot-Interaction for mobile industrial robot teams." In: *Procedia CIRP* 79 (2019), pp. 614–619.

[47] Ilias El Makrini et al. "Working with walt: How a cobot was developed and inserted on an auto assembly line." In: *IEEE Robotics & Automation Magazine* 25.2 (2018), pp. 51–58.

[48] Ce Zheng et al. *Deep Learning-Based Human Pose Estimation: A Survey*. 2020. arXiv: 2012.13392 [cs.CV].

[49] Yanjie Li et al. "Tokenpose: Learning keypoint tokens for human pose estimation." In: *Proceedings of the IEEE/CVF International conference on computer vision*. 2021, pp. 11313–11322.

[50] Run Zhou Ye et al. "Effects of Image Quality on the Accuracy Human Pose Estimation and Detection of Eye Lid Opening/Closing Using Openpose and DLib." In: *Journal of Imaging* 8.12 (2022), p. 330.

[51] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. "Pifpaf: Composite fields for human pose estimation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 11977–11986.

[52] Boston Dynamics. *Spot real world applications*. URL: https://www.bostondynamics.com/products/spot (visited on 05/02/2023).

[53] Amanda Bouman et al. "Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2518–2525.

[54] Eric M Wetzel et al. "The Use of Boston Dynamics SPOT in Support of LiDAR Scanning on Active Construction Sites." In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vol. 39. IAARC Publications. 2022, pp. 86–92.

[55] Eric M Wetzel et al. "A Step towards automated tool tracking on construction sites: Boston dynamics SPOT and RFID." In: *EPiC Series in Built Environment* 3 (2022), pp. 488–496.

[56] Brian L. Due. "A Walk in the Park With Robodog: Navigating Around Pedestrians Using a Spot Robot as a "Guide Dog"." In: *Space and Culture* (Mar. 2023), pp. 1–17. DOI: 10.1177/12063312231159215.

[57] Zhe Cao et al. "Realtime multi-person 2d pose estimation using part affinity fields." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7291–7299.

[58] Woojoo Kim et al. "Ergonomic postural assessment using a new open-source human pose estimation technology (OpenPose)." In: *International Journal of Industrial Ergonomics* 84 (2021), p. 103164. ISSN: 0169-8141. DOI: https://doi.org/10.1016/j.ergon.2021.103164. URL: https://www.sciencedirect.com/science/article/pii/S0169814121000822.

[59] Farzan Majeed Noori et al. "A robust human activity recognition approach using openpose, motion features, and deep recurrent neural network." In: *Scandinavian conference on image analysis*. Springer. 2019, pp. 299–310.

[60] Chengle Fang et al. "Research on Real-Time Detection of Safety Harness Wearing of Workshop Personnel Based on YOLOv5 and OpenPose." In: *Sustainability* 14.10 (2022). ISSN: 2071-1050. DOI: 10.3390/su14105872. URL: https://www.mdpi.com/2071-1050/14/10/5872.

[61] Chuan-Bi Lin et al. "A Framework for Fall Detection Based on OpenPose Skeleton and LSTM/GRU Models." In: *Applied Sciences* 11.1 (2021). ISSN: 2076-3417. DOI: 10.3390/app11010329. URL: https://www.mdpi.com/2076-3417/11/1/329.

[62] Google / Keras. *The Keras machine learning framework*. URL: https://keras.io/about/ (visited on 05/03/2023).

[63] Kaggle. *Kaggle 2022 machine learning and Data science survey*. URL: https://www.kaggle.com/kaggle-survey-2022 (visited on 05/03/2023).

[64] The OBS Project. *The Open Broadcaster Software application*. URL: https://obsproject.com/ (visited on 05/03/2023).

[65] Ginés Hidalgo et al. *OpenPose Documentation.* URL: https://raw.githubusercontent.com/CMU-Perceptual-Computing-Lab/openpose/master/.github/media/keypoints_pose_25.png (visited on 04/27/2023).

[66] Kilian Weinberger. *Cornell class SP17 CS4780k-Nearest Neighbors.* URL: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html (visited on 05/14/2023).

[67] B.H Hall. "Innovation and Diffusion." In: *The Oxford handbook of Innovation.* Oxford University Press, 2005. ISBN: 9780199286805.

[68] Inaki Maurtua et al. "Human–robot collaboration in industrial applications: Safety, interaction and trust." In: *International Journal of Advanced Robotic Systems* 14.4 (2017). DOI: 10.1177/1729881417716010.

[69] Luis Roda-Sanchez et al. "Human-robot interaction in Industry 4.0 based on an Internet of Things real-time gesture control system." In: *Integrated Computer-Aided Engineering* 28.2 (2021), pp. 159–175. DOI: 10.3233/ICA--200637.

[70] Dominykas Strazdas et al. "Robots and Wizards: An Investigation Into Natural Human–Robot Interaction." In: *IEEE Access* 8 (2020), pp. 207635–207642. DOI: 10.1109/ACCESS.2020.3037724.

[71] Daniil Osokin. "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose." In: *arXiv preprint arXiv:1811.12004* (2018).

[72] Paul Hur and Nigel Bosch. "Tracking Individuals in Classroom Videos via Post-processing OpenPose Data." In: *LAK22: 12th International Learning Analytics and Knowledge Conference.* 2022, pp. 465–471.

[73] Stefan Aust, R Venkatesha Prasad, and Ignas GMM Niemegeers. "Outdoor long-range WLANs: A lesson for IEEE 802.11 ah." In: *IEEE Communications Surveys & Tutorials* 17.3 (2015), pp. 1761–1775.

[74] Boston Dynamics. *Spot optional onboard computer.* URL: https://www.bostondynamics.com/resources/blog/doing-more-spot (visited on 05/02/2023).

[75] Christopher Brunner et al. "Selective combination of visual and thermal imaging for resilient localization in adverse conditions: Day and night, smoke and fire." In: *Journal of Field Robotics* 30.4 (2013), pp. 641–666.

[76] Aelee Kim et al. "The effects of familiarity and robot gesture on user acceptance of information." In: *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI).* IEEE. 2013, pp. 159–160.

[77] Ting Chen et al. “A simple framework for contrastive learning of visual representations.” In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[78] Wenqiang Zhang et al. “Efficientpose: Efficient human pose estimation with neural architecture search.” In: *Computational Visual Media* 7 (2021), pp. 335–347.

Chapter 9

Appendix

9.1 OpenPose performance in intial testing on training data

The best performance was recorded when the trainer was standing still facing the camera at a distance of about 2 meters. However, at times it struggled to predict accurately when the arms were positioned in front of the body. OP would loose sight of the arm sometimes and incorrectly predict the arm to be hanging by the persons side. Likewise, when bending the knees while facing the camera, OP would struggle to see that it was the knees which were bent and would instead predict that the legs were shrinking and growing in length as the leg was raised. Other than this OP performed great in this scenario.

OP also did well when the trainer was not facing the camera. It would make quite accurate estimates of where the arms and legs hidden from the camera should be when the person stood sideways covering limbs from sight with the rest of her body. Similarly, when the person walked past the robot and camera while gesturing, OP performed well though not always perfectly, especially when limbs were hidden. However, when the robot was moving and the person was standing still facing the robot, OP did great in its predictions.

When the lights were low OP still managed well until the lights were so low that one could barely see the silhouette of the trainer in the frame. With such low lighting OP did not find the person at all.

OP gave different results when the person was standing behind obstacles. When standing behind a chair so that the trainer's legs were partially obstructed - the chair covering the person from the ankles to the hips - OP performed well. It managed to predict where the legs were and correctly ended the legs where the feet showed under the chair. When the trainer stood behind a solid box which completely covered her legs from the hips

down, OP would sometimes 'adjust' the length of the legs and assume they were the same height as the box. At other times, OP would predict the correct length of the legs and not stretch them to be the same length as the box. An observation is that OP will often, though not always, 'fill in the blanks' when a body part is obscured or missing instead of simply 'omitting' the part by zeroing the coordinates defining it. In later tests we observed that OP did omit predicting the position of the arm of a test subject when it was raised above the persons head and the contrast to the background was very low.

When the trainer was wearing the massive backpack while facing sideways to the camera, OP still gave correct predictions. Even when the persons back was to the camera, OP was generally able to make an estimate of where the person's shoulders and head were. When the trainer was holding a long object and the contrast between the clothing and the object was good, OP predicted correctly. Problems would only occur if the contrast was low.

OP did struggle when the distance between the person and the camera increased beyond 3 meters. It would struggle with predicting where the shoulders were, and where the arms were positioned when performing gestures where the arms crossed in front of the body. OP would also switch left and right from frame to frame, changing between predicted that the trainer was facing the camera or had her back towards the camera.

Wearing a camouflage uniform in the MOCAP room did present some issues for OP as the contrast between the background and the clothes was not very large. This resulted in OP incorrectly predicting that the arms were hanging by the persons sides rather than being lifted away from the body which was the actual case. Holding the long object did not affect OP when wearing the camouflage uniform as the object was black and the contrast good.

Wearing the camouflage uniform outside, in the forest, was even more challenging. OP was at times able to find the person just fine, other times it couldn't find the person at all. The same results were observed when the trainer wore the backpack. When standing behind some thin trees while wearing camouflage OP was not able to find the person at all. The contrast was too low.

9.2 Overview of gesture separability tests done with kNN

Table 9.1: Overview of all tests for separability done using the kNN

Testing different features, k=3		
Features	Gestures	Accuracy
50 joints coordinate points	G1 to G11	36%
10 angles: a1 to a10	G1 to G11	36%
6 angles: a1 to a6	G1 to G11	54%
Testing with smaller combinations of gestures, angles as features, k=3		
Features	Gestures	Accuracy
6 angles: a1 to a6	G2, G4, G7, G11	100%
6 angles: a1 to a6	G2, G4, G5, G7, G11	80%
6 angles: a1 to a6	G1, G2, G4, G5, G7, G11	83%
6 angles: a1 to a6	G2, G4, G5, G7, G8, G11	66%
6 angles: a1 to a6	G2, G3, G4, G5, G7, G11	66%
6 angles: a1 to a6	G1 to G11	54%
10 angles: a1 to a10	G2, G4, G7, G11	100%
10 angles: a1 to a10	G1 to G11	36%
Testing with cropped image data, angles and left/right hand coordinates as features, adding neutral class to gestures (totals to 12 gestures), k=3		
Features	Gestures	Accuracy
6 angles: a1 to a6	G1, G2, G4, G5, G7, G11, G12	86%
6 angles: a1 to a6	G1 to G12	92%
6 angles: a1 to a6, 4 coordinate points	G1, G2, G4, G5, G7, G11, G12	43%
6 angles: a1 to a6, 4 coordinate points	G1 to G12	25%
10 angles: a1 to a10	G1, G2, G4, G5, G7, G11, G12	86%
10 angles: a1 to a10	G1 to G12	83%
10 angles: a1 to a10, 4 coordinate points	G1, G2, G4, G5, G7, G11, G12	43%
10 angles: a1 to a10, 4 coordinate points	G1 to G12	42%
Testing with no zero padding, cropped image data, angles as features, adding neutral class to gestures (totals to 12 gestures), k=3		
Features	Gestures	Accuracy
6 angles: a1 to a6	G1 to G12	83%
10 angles: a1 to a10	G1 to G12	25%
Testing with k=12, k=24, k=36, angles as features, using gestures G1 to G12		
Features	Value of k	Accuracy
6 angles: a1 to a6	12	82%
6 angles: a1 to a6	24	80%
6 angles: a1 to a6	36	77%

9.3 Answers to user questionnaire in the experiment

9.4 Questionnaire

The questions were scaled from strongly agree to strongly disagree.

Scenario the participants had to base their answers on: You decide to take your quadruped robot on a trip. First you have to get the robot to your car. However, the robot is too heavy for you to carry so you decide to steer it using the built-in gesture-command system (instead of using the handheld controller). On the way to your car, you come across some obstacles in your path which you have to manoeuvre the robot around.

Question 1. before the experiment

A fast response time from the robot after I have completed a command is very important to me.

B01: agree

B02: agree

B03: agree

B04: strongly agree

B05: strongly agree

B06: neutral

B07: neutral

B08: agree

Question 2. before the experiment

Having to repeat a command because the robot didn't register my movement/command or misinterpreted it as a different command is acceptable.

B01: disagree

B02: disagree

B03: strongly disagree

B04: neutral

B05: agree

B06: disagree

B07: agree

B08: disagree

Question 3. before the experiment

A gesture-command system like this seems very useful to me.

B01: strongly agree

B02: strongly agree

B03: agree

B04: agree

B05: strongly agree

B06: agree

B07: strongly agree

B08: strongly agree

Question 4. before the experiment

I expect this gesture-command system to work well on this robot.

B01: neutral

B02: agree

B03: agree

B04: agree

B05: strongly agree

B06: agree

B07: neutral

B08: neutral

Question 1. after the experiment

The gesture-command system worked as I expected.

B01: neutral

B02: strongly agree

B03: agree

B04: strongly agree

B05: strongly agree

B06: strongly agree

B07: strongly agree

B08: agree

Question 2. after the experiment

A fast response time from the robot after I have completed a command is very important to me.

B01: agree

B02: agree

B03: agree

B04: agree

B05: strongly agree

B06: agree

B07: disagree

B08: agree

Question 3. after the experiment

Having to repeat a command because the robot didn't register my movement/command or misinterpreted it as a different command is acceptable

B01: agree

B02: strongly disagree

B03: disagree

B04: disagree

B05: agree

B06: neutral

B07: agree

B08: disagree

Question 4. after the experiment

There were commands you wanted to give the robot which were not available.

B01: disagree

B02: disagree

B03: strongly disagree

B04: agree

B05: agree

B06: neutral

B07: strongly agree

B08: agree

Question 5. after the experiment

I felt uncomfortable while interacting with the robot.

B01: strongly disagree

B02: strongly disagree

B03: strongly disagree

B04: agree

B05: disagree

B06: strongly disagree

B07: strongly disagree

B08: disagree

Question 6. after the experiment

You see a gesture-command system like this being used in the real world, and expect to encounter it in industry in the near future (5+ years).

B01: strongly agree

B02: neutral

B03: strongly agree

B04: strongly agree

B05: strongly agree

B06: strongly agree

B07: strongly agree

B08: strongly agree

Answers to the open-ended questions

1. *What funcionality would you want from a system like this? What improvements could be done?*

   Answer from participant B01: "Adding more commands for even more utility perhaps. Since the exisiting ones worked very well."

   Answer from participant B02: "Change/modify direction while walking to better maintain wanted trajectory"

   Answer from participant B03: "High accuracy. Redundant systems to protect yourself from the robot."

   I interpret participant B03's answer as high accuracy is a functionality they would want from a system like this. They also want the ability to emergency stop the robot should anything go wrong. We did have an emergency stop which we would initiate should the participant be in harms way, however, the participant wasn't physically holding the emergency stop button. The participant might have felt more at ease if given the button to hold for themselves.

   Answer from participant B04: "Faster response time. Ability to change gesture-commands, so every user will be able to have own gestures. (For example left arm up = to go or something like that)."

   This participant wishes to be able to personalise it more. By deciding themselves what body gesture equals what action from the robot.

   Answer from participant B06: "General improvements such as: faster and reliable detection. Like is, but more efficient."

   Answer from participant B08: "The range of the sensor (both close and far away) could be improved."

   This participant was especially tall compared to the rest. If the participant stood too close the robot, then the top half of their body would be out of frame and the robot wouldn't be able to see them - this applied to all participants. Standing too far away the human recognition software OP would struggle to locate the arms correctly due to the low image resolution. Three of the gestures involved the participant raising

their arms above their head. If the participant is standing too close then their arms would be out of frame and the system would not see or recognise that their arms were up. So the participant had to stand at least 1 to 1.5 meters away from the robot. Since this participant was particularly tall, they had to stand further away from the robot to be in frame. However if they stepped too far away then the human recognition software/model would struggle. This participant was forced to stand quite far away to be in frame and therefore had a smaller distance range in which he could interact with the robot than a shorter participant would have.

Summary of answers for question 1
The participants mentioned adding more commands as an improvement to the system. Having the option to change the trajectory of the robot while walking. Also having the opertunity to personalise and change around which gesture activates which command. Another common answer was to improve the accuracy and efficiency of the system and increase the distance range in which they could give commands from.

2. *If you answered 'strongly agree' or 'agree' on question 4. above, which commands are missing?*

   Answer from participan B04: "Were actually thinking about something funny like a 'do a circle'; to make it feel more friendly?"

   Answer from participant B05: "Instant shutdown command. Walk sideways (left/right). Back up/reverse."

   Answer from B06: "Walk a path, it is hard to get it to walk in a straight line, being able to gesture small increments in rotation would be beneficial."

   Answer from participant B07: "Look and follow mode"

   Answer from participant B08: "I would like it to follow me and not just walk in a straight line."

   Summary of answers for question 2
   Most participants wanted a 'follow me' command, or an easier way to adjust which direction the robot was walking in. One participant

wanted commands for more controll of the robots movements, adding walk backwards, sideways, and an emergency stop command. One participant wanted commands/functionality for actions which are purely for entertainment to make the robot come across as less machine-like and more friendly.

3. *If you answered 'strongly agree' or 'agree' on question 5. above, please specify what instances made you feel uncomfortable.*
   B04: "Sometimes it felt like robot would continue to move without stoping (might be because of the response time)."

   Summary of answers for question 3
   A lack of control of the robot when it was walking was uncomfrotable for the participant. They also mentioned verbally that it was uncomfortable because they were worried the robot would injure itself as it did not stop immediately when the walk command ended.

4. *Please let us know whether you have any additional observaions or input.*

   B01: "The robot's response felt very natural and quick"

   B02: "It worked really quite well, easy to use and interact."

   B04: "Gesture-commands actually worked pretty well. I liked to have a more close interaction with a 'robot' through body language, rather than through 'programing' his actions."

   I interpret their answer as: "I liked having a more close interaction with the robot through body language, rather than through 'programing' its actions."
   Participant B04 thought the gesture-command interaction system/application worked really well, and enjoyed controlling the robot with their body-language instead of using more convential options such as through a handheld controller or a physical operating station. Using body signals made the interaction more intimate of sorts and less machine-like perhaps?

   B06: "As in 2., being able to turn it without standing directly in front of it would be beneficial."

B08: "I want one of these."
Summary of answers for question 4
The participants experienced the gesture-command system to work really well, but still see areas for improvement in controlling the robot.

9.5 Live experiment user consent form

Do you want to participate in the researchproject "Human Robot Interaction with a Quadruped Robot"?

This is an inquiry to you about participating in a research projectwhere the purpose is to test how well a system works in practice. The system consists of controlling a four-legged robot (Spot from Boston Dynamics) using bodysignals. In this document, we provide you with information about the aims of the project and what participation will mean for you.

Purpose

In this project we, will examine how well this interaction system works in practice today and whether this is a solution that can be expected to be encountered in industry. You are given a series of pre-selected body signals that you must perform in order to control the robot. We will investigate how well the system's classification algorithm is able to distinguish between the different body signals, how responsive the robot and the system are perceived, and find any weaknesses in the system.

We use OpenPose for human recognition in video footage (of you) taken by the on-board cameras on the robot. The data from OpenPose is used as input for a LSTM classification algorithm, which distinguishes the different body signals.

Vi aim to have about 10 participants. As a participant you will first be given more information on the setup of the experiment. You will then be given a few questions to answer about your expectations for the system. Then you will be shown each of the body signals which you will try with the robot. After that, we will walk through a course which you will take robot through using the body signal system. Finally, you will be asked some questions about the experiment.

Who is responsible for the research project?

This project is a master thesis carried out at the Department of Informatics (IFI) at the University of Oslo in collaboration with the Norwegian Defense

Research Institute (FFI). IFI is responsible for the project. The robot Spot is made available by FFI.

Why are you being asked to participate?

We ask you to participate because we need as many participants as possible. Participants do not need any specific background, however, must have full functionality in their arms and legs. The aim is a minimum of 10 participants, but more participants than this is positive for the project as we will have more data to use.

If you would like to participate, you must sign the las page of this form, and then we will contact you.

If you do not want to participate, we will not contact you.

What does participating mean for you?

As a participant, you will be contacted about the meeting place and time. You will be asked to control the robot by performing different body movements as instructed by the interviewer. A video will be taken of you during the experiment. Estimated duration is 45 minutes in total.

Participation is voluntary

Participation in the project is voluntary. If you choose to participate, then you can withdraw your consent at any time without giving a reason. All your personal data will then be deleted. There will be no negative consequences for you if you do not want to participate or later choose to withdraw.

Your privacy – how we store and use your information

We will only use the information about you for the purposes we have described in this document. We treat the information confidentially and in accordance with the privacy reuglations.

- Your data will not be shared with others. Only master student Tale Sandberg and supervisors Tønnes Nygaard and Benedikte Wallace have access to your data.
- It is the data processor Tale Sandberg who collects, processes, and saves the data.
- We make sure that no one can get hold of the data we collect about you. All data is stored on a secure computer. Your name and contact

details are replaced with a code that is stored on a separate name list, which is stored in a different place from the rest of the data.

- We make sure that no one can recognize you when writing the master thesis. You will be anonymized if images of you are used in the master thesis.
- We follow the law on data privacy, and have applied for permission to process personal data at Sikt.

What happens to your personal data when the research project ends?

The project is scheduled to end in June 2023.

We will delete your data when the master's thesis is finished in June 2023. If images of you from the video recording are used in the master's thesis article, then you will be anonymized.

What gives us the right to process personal data about you?

We process information about you based on your consent.

On behalf of the University of Oslo, the Personal Protection Service has assessed that the processing of personal data in this project is in accordance with the privacy regulations.

Your rights

As long as you can be identified in the data material, you have the right to:

- access to the information we process about you, and to be given a copy of the information
- to have information about you corrected that is incorrect or misleading
- to have personal data about you deleted
- to send a complaint to the Norwegian Data Protection Authority about the processing of your personal data

If you have questions about the study, or want to know more about or exercise your rights, please contact:

- University of Oslo, master student Tale Sandberg on email talehs@ifi.uio.no, tlf: +47 XXX XX XXX, or University of Oslo Tønnes Nygaard on email tonnesfn@ifi.uio.no, tlf. +47 XXX XX XXX

- Our data protection representative: Maren Magnus Voll by e-mail: personvernombud@uio.no, tlf. +47 XXX XX XXX

(**Note** the tlf numbers were provided to the test participants but have been censored here)

If you have questions related to Personal Protection Services' assessment of the project, you can contact:

- Privacy services by email (personverntjenester@sikt.no) or by phone: 53 21 15 00.

Best regards
Tønnes Nygaard Institutt for Informatikk Universitetet i Oslo
Tale Sandberg Institutt for Informatikk Universitetet i Oslo

Declaration of consent

I have received and understood information about the project Human Robot Interaction with a Quadruped Robot, and have been given the opportunity to ask questions. I agree to:

- to participate in video recording of me
- that anonymized photos of me can be published in the master thesis article

I consent to my information being processed until the project is finished in June 2023

(Signed by project participant, date)

_____ _____

(Email) (Phone)