TU WIEN Informatics

Development of A Vision-Based Foot Massage Robot

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

David Ammer, BSc
Matrikelnummer 01619041

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Horst Eidenberger

Wien, 13. Jänner 2023

David Ammer	Horst Eidenberger

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Development of A Vision-Based Foot Massage Robot

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

David Ammer, BSc
Registration Number 01619041

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Horst Eidenberger

Vienna, 13th January, 2023

David Ammer	Horst Eidenberger

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

David Ammer, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. Jänner 2023

David Ammer

Danksagung

An dieser Stelle möchte ich mich bei meiner Familie und meinen Freunden bedanken, die mich unterstützt haben. Ebenso bei meinen Studienkollegen, mit denen schwierige Situationen humorvoll aber doch mit Stil gelöst werden konnten. Einen besonderen Dank möchte ich an meinen Bruder und meine Freundin richten.

Kurzfassung

Bisher wurden viele unterschiedliche Roboter gebaut und entwickelt um den Menschen zu helfen, die Lebensqualität zu verbessern oder die Arbeit zu erleichtern. Heutzutage leben wir immer mehr in einer Leistungsgesellschaft, wodurch viele Menschen mehr Stress und Druck im Leben verspüren. Eine gängige Methode um den Stresspegel zu senken ist zum Beispiel eine Fußmassage.

Diese Diplomarbeit präsentiert einen Prototyp von einem bildbasierten Fußmassageroboter. Dazu wurden ein Hardware- und ein Softwaresetup erstellt. Ein Teil des Hardwaresetups umfasst die Modifikation des Roboterarms. Dies beinhaltet das Ersetzen des Greifers, der ursprünglich als Endeffektor diente, mit einem Deoroller. Dieser Deoroller ist zusätzlich mit fünf Tastern ausgestattet, die den physischen Kontakt zwischen dem Roboter und einer Fußsohle erkennen sollen. Außerdem wurden eine Plattform und eine Fußablage angefertigt.

Eines der Hauptziele des Softwaresetups ist die Identifizierung und Erkennung der Fußsohle und des Roboters selbst. Dazu wurde ein *Mask Region-Based Convolutional Neural Network* trainiert. Dieses Netzwerk markiert die Pixel von der Fußsohle und dem Roboter auf dem Eingabebild. In der weiteren Abfolge wurde eine *Sohlenmaskenvorlage* erstellt, auf welcher die Massagepunkte einmal manuell platziert wurden. Die Punkte dieser Vorlage werden während der Massage auf die Maske des zu massierenden Fußes projiziert. Zusätzlich werden diese Punkte in eine Simulation, in welcher der Roboter abgebildet ist, geladen. Mittels inverser Kinematik werden die Rotationswinkel der einzelnen Servomotoren in der Simulation berechnet, sodass der Endeffektor den Massagepunkt erreicht. Danach werden diese Winkel an den Roboterkontroller, in diesem Fall ein Arduino, gesendet um den Roboter zu bewegen. Wird ein physischer Kontakt erkannt, wird die Massagetechnik *Friktionieren* ausgeführt. Des Weiteren können einige Einstellungen der Massagebewegungen, wie zum Beispiel die Geschwindigkeit, verändert werden. Dazu wurden drei neuronale Netzwerke (Convolutional Neural Networks) trainiert, die zur Audioerkennung für Befehle eingesetzt werden.

Schließlich wurde der Prototyp in einer Nutzerstudie mit einer Gruppe von *13* Teilnehmer*innen getestet. Bei der Auswertung wurde festgestellt, dass ein Roboterarm eine *State-of-the-Art* Fußmassage durchführen kann. Der Stresspegel ist niedriger als vor der Massage und die Teilnehmer*innen sind entspannter. Jedoch massiert der Ro-

boter den rechten Fuß genauer als den Linken und reagiert nicht immer korrekt auf die Sprachbefehle.

Abstract

So far, many different robots have been built and developed to help people, improve the quality of life or make work easier. Nowadays we live in a society where pressure arises due to work and other factors, causing people to develop stress in their lives. A massage can have a relaxing effect and help alleviate stress, such as a foot massage.

This thesis presents a prototype of a vision-based foot massage robot. For this purpose, a hardware and a software setup was created. One part of the hardware setup was the modification of the robotic arm. This involved replacing the claw, which originally served as an end-effector, with a deodorant roller. This deodorant roller is additionally equipped with five buttons, which are supposed to detect the physical contact between the robot and the sole. Furthermore, a platform and a footrest was built.

One of the main goals of the software setup is to identify and recognize the sole of the foot as well as the robot. To achieve this goal, a *Mask Region-Based Convolutional Neural Network* was trained. This network marks the pixels from the robot and the sole of the foot on the input image. The massage points are manually generated once on a *sole mask template* and projected onto the mask of the foot during the massage. In addition, these points are loaded into a simulation, where the robotic arm is also mapped into. The rotation angles of each servomotor are calculated in the simulation with inverse kinematics, so that the end-effector reaches the massage point. Then these angles are sent to the robot controller, in this case an Arduino, to move the robot. If physical contact is detected, the massage technique *frictioning* will be performed. Furthermore, some settings of the massage movements, such as the speed, can be changed. For this purpose, three *Convolutional Neural Networks* were trained, which are used for audio command recognition.

Finally, the prototype was tested in a user study with a group of *13* participants. Through the evaluation it was discovered that a robotic arm can perform a state-of-the-art foot massage. The stress level of the participants is lower than before the massage and they reported to be more relaxed. However the robot massages the right foot more accurate than the left one and the robot does not always respond well to the voice commands.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

Robots have been indispensable for decades. They facilitate our work and also daily life. Further, robots appear in various shapes and colors. Some examples include robotic hands for grasping objects, a human-like robot for social contact in a nursing home or small moving robots for transporting goods. Amazon uses many of them to transport packages from one workplace to another. Moreover, their idea to have a fully autonomous warehouse is becoming real [Sta22]. For this purpose, all robots need a system to control their actions. This is often a combination of different sensors to receive various data (position, pressure, distance, etc.) and a "brain" to process this information. Therefore, robot kinematics or machine learning is often used [CHM20]. Due to the increasing amount of available data and complexity of various algorithms, machine learning has become an important area of research. According to Naqa and Murphy [ENM15], it is designed to emulate the human brain and intelligence by learning from its surrounding environment. Various fields of application are computer vision, pattern recognition, biomedical engineering, etc., in order to select relevant results of searches, detect different objects in images or convert speech to text. According to Shinde and Shah [SS18], deep learning is a subset of machine learning. It is used for more complex tasks. An example is understanding images as to annote and tag them. This is useful for image indexing, image retrieval and image search engines [GWB17]. Further, deep learning is outperforming approaches from machine learning for image classification [KSH17] and speech recognition [HDY$^+$12]. Another area of application is the healthcare sector. According to Razzak et al. [RNZ18], consumers expect the highest level of care. Therefore many medical professionals are trained to interpret available information such as images of maladies. As mentioned before, the increasing amount of data makes it more difficult to develop an understanding of it. Due to the complexity of medical pictures, it is quite different for experts to interpret them. In addition, there exists extensive variation in interpretation

between specialists. Furthermore, they may miss important information due to their heavy workload and the resulting tiredness. Therefore, state-of-the-art deep learning architectures are used for medical image segmentation and classification [RNZ18].

According to Varma et al. [VJMJ21], younger people deal with more problems concerning their overall physical and mental health during the COVID-19 pandemic. They perceived stress, anxiety, depression and financial insecurity. Further, it seems to be that stress can cause cancer, heart disease, high blood pressure and suicide [Kee03]. Different methods such as exercising, eating healthy, etc. can be used to improve mental health. Another method is receiving a massage. According to Vickers and Zollman [VZ99], a massage is used to promote relaxation, treat painful muscular complaints, reduce anxiety and stress. To achieve these results, techniques such as friction massaging or kneading are used. Friction massage is a deep massage in which the thumbs or fingertips make circular motions. Different parts like the back, head, neck, stomach, shoulders, etc. can be focused. According to Hayes and Cox [HJ99], a foot massage can reduce heart rate, blood pressure and increases relaxation. This observation is the result of a five minute foot massage.

In this thesis a combination of cameras, deep learning, a robot and foot massage techniques are used. The aim of this work is to develop a massage robot that performs a state-of-the-art foot massage based on audio commands and pictures from the cameras. The following questions are in focus to be answered in this thesis:

1. Can a robot perform a state-of-the-art foot massage?
2. Is it possible to adjust the massage in a convenient way based on the audio feedback from the user?
3. Is it possible to reduce the stress level and to improve the well-being of the user within a twenty minute foot massage from a robot?

Several problems have to be solved to answer these questions. First, the setup of the hardware has to be constructed. This includes building a footrest, modifying the head of the robot, mounting the robot on a wooden platform, connecting the footrest to the platform, and placing the cameras. Afterwards, a state-of-the-art method for recognizing feet and audio commands has to be researched and compared. Then the most promising classifier has to be chosen and trained. For the training, a dataset has to be found or created. After that, the cameras have to be calibrated and a distance estimation function has to be discovered. Then a method to calculate inverse kinematics for the robot has to be explored. Later, an application has to be implemented that, will recognizes the foot, controls the robot, responds to audio commands and performs state-of-the-art foot massages. After that, a case study has to be designed and evaluated. In the last step, the evaluation has to be analyzed and a final conclusion has to be made.

1.2 Contribution

This thesis provides a prototype consisting of a robot and a software solution and uses it for a case study to investigate the effect of a robotic foot massage. Several steps were necessary for the contribution. First, literature research was conducted to gather background information. Then, the setup of the hardware was constructed. Furthermore, a pre-trained model for object detection and segmentation was trained and fine-tuned with a new dataset. It contained different images of feet with and without socks. Once the model was ready for use, the cameras were calibrated and a distance estimation function was discovered. Then the inverse kinematics for the provided robot were calculated and implemented. To simulate a state-of-the-art foot massage, the points to be treated on the foot, were generated. At the end of the implementation, an audio model was provided to recognize commands and manipulate the movement of the robot in a convenient way. Finally, a case study was designed, focusing on the questions that this thesis aimed to answer. In addition, the results were analyzed and presented.

1.3 Thesis Structure

This thesis focuses on developing a hardware and software prototype that includes a footrest, feet and audio command recognition, inverse kinematics calculation, control of a robot, performance of state-of-the-art foot massage techniques with the robot, evaluating the effect of a robotic foot massage and manipulating the motion of the robot via audio commands. To achieve this goal, the structure of the thesis provides for the following steps. In Chapter 2, the theoretical background is explained to provide the reader the necessary background knowledge. This includes machine learning methods, camera calibration, robot kinematics and foot massage techniques. A deep insight into the design is shown in Chapter 3 and implementation of the prototype is provided in Chapter 4. The case study, its results and the evaluation of the prototype are explained in Chapter 5. Further, limitations and future research are presented in Chapter 6.2. At the end, a summary of the conclusions is provided in Chapter 6.1.

CHAPTER 2

Theoretical Background

In this chapter, an overview of the used methods in this thesis is presented and related background information is provided. First, a brief insight into deep learning techniques that are used for classification, is given in Section 2.1. Then, Section 2.2 presents the basics of calibrating and undistorting a camera. In Section 2.3 fiducial marker is explained and how to detect it. Further, a brief insight into robotics including robot kinematics, is given in Section 2.4. At the end, Section 2.5 provides information about the effects and techniques of a foot massage.

2.1 Deep Neural Networks

This section describes relevant background information on deep learning techniques used in this thesis. First, the basics of machine learning are described. Then, artificial neural networks and convolutional neural networks are explained. At the end, mask region-based convolutional neural networks is presented.

2.1.1 Basics of Machine Learning

According to Mitchell [Mit97, p. 2], the definition of learning is defined as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.". In the case of this project, the three features are defined as:

- Experience E: a dataset with images of labeled feet and robots.
- Task T: recognize and label the mask of a foot and a robot.
- Performance P: percentage of the overlap between the ground truth and the prediction, also called Intersection over Union (explained in Section 2.1.9).

Machine learning can be categorized into two main classes: *supervised* and *unsupervised* learning. The main difference between them is the labeled dataset. Supervised learning is often used for classification [Zha10]. Further, the dataset is divided into a training set and a validation set. According to Quin [Qin20], the training set is used for adjusting the parameters and training the model. The validation set is used to test the model during the training. Therefore, it is possible to improve the model while training. Supervised learning can be divided into two main categories: *regression* and *classification* algorithms. Regression algorithms are used to predict a continuous quantity and classification algorithm predict a discrete or categorical output, which usually represents a class [CHP21]. An example of regression would be a prediction about the temperature in the next few days and for classification of whether the weather will be hot or cold. To measure the performance of a classification model, accuracy is used. Accuracy refers to the number of correct predictions in percentage. The opposite is the error rate, which refers to the amount of incorrect outcomes, which is also expressed as a percentage [Jap06]. To obtain these two evaluation values, the model must process a set of data, that was not used for training. This dataset is called test set. Unsupervised learning uses a dataset without a given label. It discovers similarities in data groupings or hidden patterns. An example is clustering [Qin20]. Unsupervised learning will not be mentioned in further detail, as this work focuses on supervised machine learning.

Typical problems when training a model are *underfitting* and *overfitting*. Overfitting means that the model performs well only in the training set and not in the test set. Therefore, the model is not flexible and adaptive. It learns all the noises and too many details from the training set. On the other hand, underfitting is the counterpart of overfitting. Generalization and modeling the data is not possible. The model is not able to capture patterns or trends in the data as well as having a bad performance on the training data [ZZJ19].

2.1.2 Artificial Neural Networks

Artificial neural networks (ANNs) are inspired by the human brain. It is a machine learning algorithm designed to emulate the learning behavior of a person [JMM96]. Two parts of the human brain are *biological neurons* and *synapses*. A biological neuron processes incoming information and synapses form a functional unit between two neurons [JMM96]. An artificial neuron is a simplified model of the biological one. In addition, it is also a computational unit. Figure 2.1 shows the design of an artificial neuron.

- x_n is the input value where n goes from 0 to m.
- w_n is the *weight* value where n goes from 0 to m.
- θ is the bias.
- F is a *transfer* function.
- y is the output value.

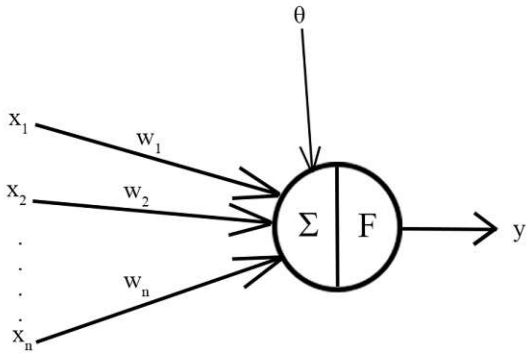

Figure 2.1: Artificial neuron design.

In Figure 2.1 it is shown that the body receives the information from the inputs $(x_1, x_2...x_n)$. To improve the computational efficiency of learning algorithms, this information is normalized. In addition, each input has a weight $(w_1, w_2...w_n)$ which is multiplied together. The weight symbolizes the relevance of the input signal with respect to the functionality of the artificial neuron [DSSF+17]. Then the body sums up the weighted inputs and its bias, afterwards processing the sum with a transfer (activation) function. This function can be any mathematical function like a *Step* function, a *Linear* function or a *Non-linear* function (*Sigmoid*, *rectified linear unit*). Furthermore, the properties of an artificial neuron are defined [KBK11]. The simplicity of an artificial neuron model can be seen in Equation 2.1.

$$y = F(\sum_{n=1}^{m} w_n * x_n + \theta) \tag{2.1}$$

Most of the time non-linear activation functions are used because a neural network with many hidden layers would become a large linear regression model. Consequently, it would be useless to train with real-world data, in order to learn complex patterns. The step function is a binary function. In the binary function, a threshold value is defined to decide whether a neuron should be activated or not. Therefore, only two output values (for example *0* and *1*) are possible [KBK11]. When a neuron is deactivated, its output is not passed on to the next layer. The mathematical description is shown in Equation 2.2.

$$y = \begin{cases} 1 \text{ if } w_n x_n \geq \text{threshold} \\ 0 \text{ if } w_n x_n < \text{threshold} \end{cases} \tag{2.2}$$

A layer is a combination of neurons, with its results being passed on to the next layer. Every single neuron of a layer is connected to every neuron of the next layer. When two or more artificial neurons are combined, they form an artificial neural network. According to Da Silva et al. [DSSF+17], ANNs can be divided into three parts:

- Input layer: This layer receives normalized information from the external environment.
- Hidden layers: These layers process the most and are responsible for extracting patterns. The name *hidden* represents the property that the outputs of these layers are only available within the network.
- Output layer: This layer presents the final results.

It is important to mention that the output layer presents a prediction as output. Also, an artificial neuron is called *perceptron*, if a binary step function (activation) is used, which is shown in Equation 2.2. It is most commonly used in the last layer to solve classification problems. For the output layer, a linear transfer function is often used, which calculates a linear transformation over the sum of the weighted inputs and the bias [KBK11]. The hidden layers usually use a non-linear function such as *Sigmoid* or *Rectified Linear Unit* (ReLU) [1]. Linear functions can only learn linearly separable problems. Therefore non-linear functions are often used to solve non-linear problems. According to Sharma et al. [SSA17], ReLU is used regularly, because in most cases it has a better performance than other activation functions. The mathematical representation is shown in Equation 2.3 where $z = w_n * x_n + \theta$.

$$f(z) = \max(0, z) \tag{2.3}$$

Moreover, it does not activate all the artificial neurons at the same time. The neurons will be deactivated if the output of the linear transformation is less than *0*. ReLU has two linear pieces, for this reason it is considered a piecewise linear function. Further, the activation function ReLU can be seen in Figure 2.2.

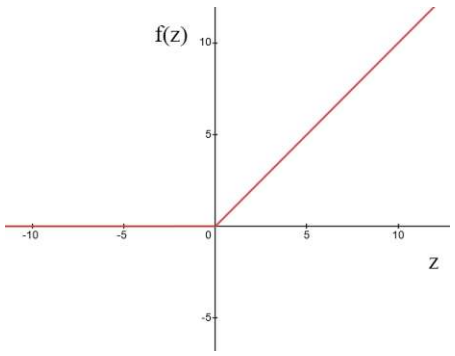

Figure 2.2: Rectified linear unit function.

Another non-linear activation function is the Sigmoid function, which is often used for the output layer. It is an S-shaped curve and belongs to the family of squash functions.

[1] ReLU appears later in the evolution of neural networks

It maps the interval (-∞ , ∞) to (*0,1*). Therefore, the Sigmoid function is useful to convert real numbers into properties [DQZ18]. Equation 2.4 shows the mathematical representation.

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

The topology or architecture of an artificial neural network is the interconnection of individual neurons. Numerous possible topologies can be created, which can basically be divided into two classes: *feed-forward* topology and *recurrent* topology. Figure 2.3 shows those two basic topologies of an artificial neural network [KBK11].

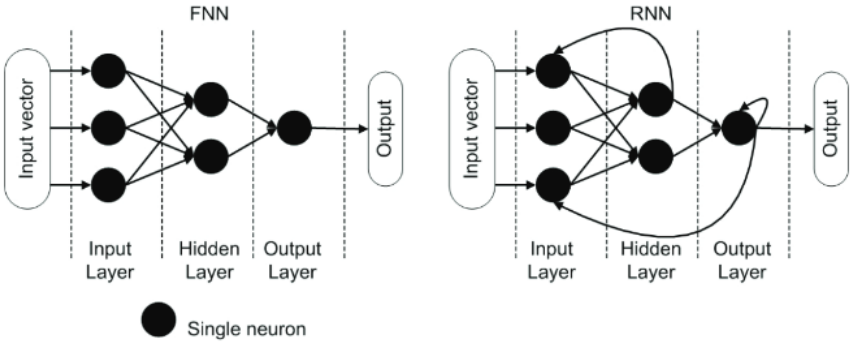

Figure 2.3: Graphical representation of a feed-forward (left) and recurrent (right) topology [KBK11].

One can see that a feed-forward topology (FNN) is an acyclic graph. The information always flows in one direction without back-loops. The *input* marks its starting point and the *output* its destination. Further, Figure 2.3 shows a simple feed-forward neural network with one hidden layer. According to Lou et al. [LPW+17], the *width* and *depth* of a network is often specified by the architecture. The amount of hidden and output layers determines the *depth* of a network, while the maximal number of nodes in a layer defines the *width* of a network. A network with more than one hidden layer is classified as a *deep neural network* [LRM20].

In a recurrent topology (RNN) or feedback architecture, the output of the neurons is used as feedback. The information flows not only in one direction, but in both. From the *input* to the *output* and in the opposite direction as well [KBK11]. In addition, the feedback is used for other neurons for dynamic information processing [DSSF+17]. With this ability, an internal state of the network is possible. Moreover, it is possible to show a dynamic temporal behavior, as the inputs of each neuron are modified within the feedback paths. This leads the network into a new state [JMM96]. A recurrent topology is shown in Figure 2.3.

Every network has to be trained. To gain an understanding of how to adapt a network model, a *loss* function like *cross-entropy* is used. Therefore, the output of a model has

to be evaluated and compared to the label, to measure accuracy. The main goal is to minimize the loss between the model prediction and the ground truth output. Moreover, the best model has a cross-entropy loss of *0*. To achieve such a value, an optimization algorithm like *gradient descent* is used. This is one of the most common methods for neural network optimization [Rud16]. With this algorithm, the gradient of a loss function is calculated and moved in the opposite direction of the gradient, in order to search for values that minimize the function at a local or global minimum. This means that after each epoch the weights and biases are updated in the opposite direction of the gradient of the loss function. This process is also called training.

In one *epoch* the neural network is trained with all the training data for one cycle. Further, *backpropagation* is a method of reducing the cost function by modifying the weights. Due to that, the gradient is computed by going backwards through the network. Backpropagation also calculates the error with the partial derivative of the gradient [LBH15]. The counterpart is *forward propagation*, where the predicted value is generated. In addition, the learning rate is a tuning parameter in an optimization algorithm. It often has a range between *0.0* and *1.0* and scales the magnitude of the weight during training. It is also used as a step size and indicates how slowly or quickly the weight values of the model are updated with respect to the loss gradient descent. On the one hand, a small learning rate requires many updates before the minimum point is reached and causes a training algorithm to converge slowly. On the other hand, a learning rate with a too large value causes the training algorithm to diverge and skip the optimal solution [Ben12]. The loss function, learning rate and optimization algorithm are also known as hyperparameters.

A variation of gradient descent in which the parameters are updated incrementally for every predefined number of examples, is called *stochastic gradient descent* (SGD). This predefined number of examples is known as *batch size* [Mit97]. Moreover, the batch size is also a hyperparameter [YS20]. In the next subsection, a class of ANNs will be explained.

2.1.3 Convolutional Neural Networks

In this work, two main goals are the detection of the foot and the recognition of audio commands. The most common solution was used. *Convolutional neural networks* (CNNs) are often used for facial recognition, object detection, semantic segmentation and also for data that is not images such as audio. Moreover, they are a class of artificial neural networks [YNDT18]. One of the most advantageous aspects of CNNs is that the number of parameters in ANNs is reduced. CNN is shift- and space invariant. Therefore, it is not necessary to know the position of a face in an image, for example. The only goal is to detect the face in the picture [AMAZ17]. CNN has three main types of layers [YNDT18]:

- Convolutional layers
- Pooling layers
- Fully-connected layer

A convolutional layer consists of a combination of nonlinear and linear operations. Linear operations are convolution operations and nonlinear operations are activation functions. Further, this layer performs feature extraction. The linear operation *convolution* uses a (filter) kernel, also known as a filter bank, which is applied across the input. A kernel is a small array of numbers that can be multi-dimensional and the size of the kernel is smaller than the input. These numbers are weights and are learned during training. The input is also an array of numbers called *tensor*. To obtain the output tensor, an element-wise product between each element of the input (patch of the image) and the kernel has to be calculated. After that, the values are summed up and the result is passed to the corresponding position of the output tensor, also called *feature map* [YNDT18]. This calculation step is known as *convolution*. This process is repeated with the next patch of the image until all values of the image are convolved and calculated. The patch of the image has the same size as the kernel. A visual representation of the calculation is shown in Figure 2.4.

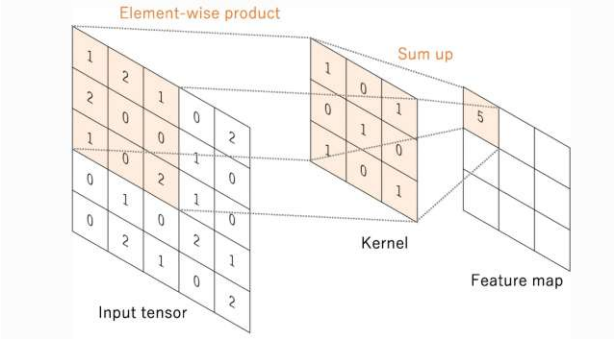

Figure 2.4: A convolution operation [YNDT18].

CNNs have the following hyperparameters that have to be set before the training: *kernel size*, *number of kernels*, *padding* and *stride*. The parameter stride determines the distance between the kernel positions. Normally *1* is used for the stride value otherwise the output is goint to be smaller than normal. In this way, downsampling of the feature maps can be achieved [YNDT18]. Since it is not possible to center the outermost elements, the resulting feature map will be smaller after each convolutional layer. Therefore, zero padding is usually used. This technique adds zeroes to every side of the input tensor. After that, it is possible to center the kernel on the outermost elements [YNDT18]. Figure 2.5 shows a convolution operation with zero padding.

The kernels are parameters. In the convolutional layer, they are learned during training. Weight sharing is a key feature of a convolution operation. This means that the kernels are shared for all image positions [YNDT18]. Fewer weights have to be learned and reducing the number of parameters increases the efficiency of the model. Furthermore, this leads to translation invariance. The features can be detected independently of their location in the image. Moreover, it is not possible to use different filter banks in one feature map. The weights of the kernel do not change during convolution. Another

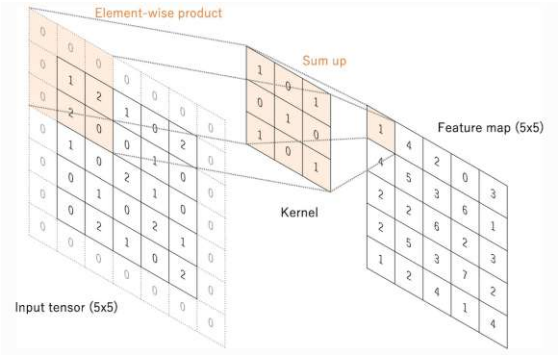

Figure 2.5: A convolution operation with zero padding [YNDT18].

property is that CNNs can be hierarchical. This happens when one convolutional layer follows another convolutional layer. An example would be edge detection on a car. The first layer would detect all edges and the next layer would combine these edges to detect other features such as the shape of the wheel and windows.

The *pooling* layer is also known for downsampling. This operation reduces the number of parameters that can be learned. Therefore, downsampling operations are used to reduce the in-plane dimensionality of the feature maps. In addition, it extracts features that are rotation and translation invariant. The pooling layer uses a filter without any weights across the entire input. This is similar to the convolution layer but uses an aggregation function instead of a convolutional one. Further, the pooling layer has the hyperparameters *filter size*, *stride* and *padding*. Pooling operations can be divided into two types: *Max Pooling* and *Average Pooling* [YNDT18].

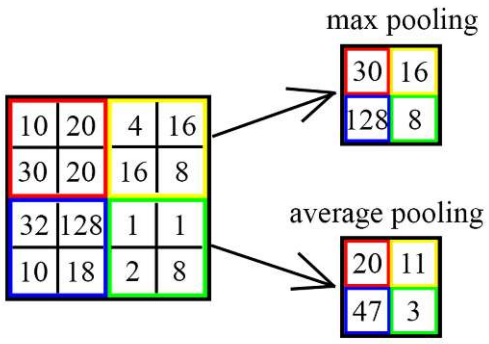

Figure 2.6: Visual representation of max and average pooling.

Figure 2.6 shows the pooling operations. The hyperparameters are stride with a value of *2*, a filter size of *2*x*2* and no padding. Max pooling selects the maximum value of a patch from the feature map, while average pooling takes the average of the patch. In both cases, the depth of feature maps remains unchanged [YNDT18]. After the pooling

operation, the result has to be multiplied by a trainable weight. Then a trainable bias is added and finally a non-linear transfer function (ReLU) is used [SMB10].

In the fully-connected layer, the output of the previous convolution or pooling layer is transformed into a one-dimensional array of numbers. This operation is called *flatten*. The classification task is performed on this layer as well. Each node of a fully connected layer is connected to every output of one or more fully connected layers by a learnable weight. The last fully-connected layer has the same number of output nodes as the number of classes to be predicted. Further, a *softmax* function is used that normalizes the output between *0* and *1* such that the sum of the value is *1* [YNDT18]. Compared to a fully connected network, CNNs are more efficient because they have fewer connections and are easier to train. The backpropagation needs to be modified to use shared-weight. An example of a CNN is the VGG-19 model, which is used for image classification [SZ14]. It is shown in Figure 2.7.

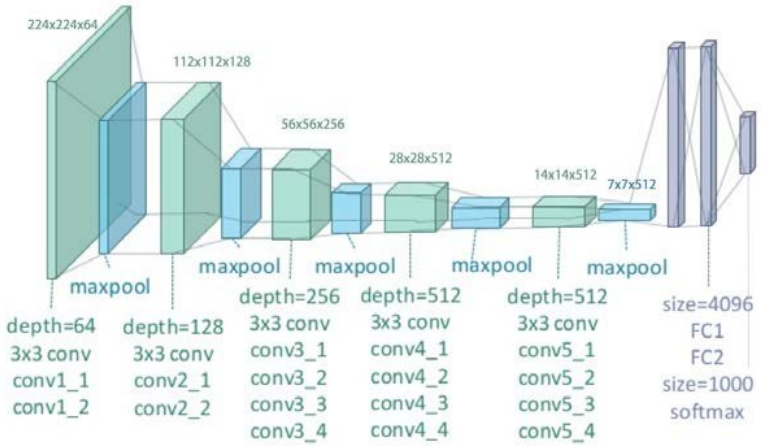

Figure 2.7: Visual representation of the VGG-19 architecture [ZYM18].

2.1.4 Residual Neural Networks

Observations were made that the depth of a CNN is of importance for image recognition tasks [SLJ+15] [SZ14]. The accuracy of a model can be increased with the number of layers stacked. Unfortunately, a problem occurred when a deep model was changed to have even more depth. By increasing the depth, the accuracy degrades rapidly and the training error increases. However, overfitting was not the reason for these troubles [HS15] [SGS15]. The problem is called *degration* [HZRS16]. This can be seen as an indicator that it is not always easy to optimize different systems. To solve this problem, a *Residual Neural Network* (ResNet) was developed in which stacked layers fit a residual mapping. For this purpose, *residual blocks* are required.

Figure 2.8 shows a residual block used in ResNet. It consists of a *shortcut connection* and stacked layers. ResNet can be trained by stochastic gradient descent with backpropagation like CNNs. The advantage of ResNet is that it leads to faster convergence at earlier stages

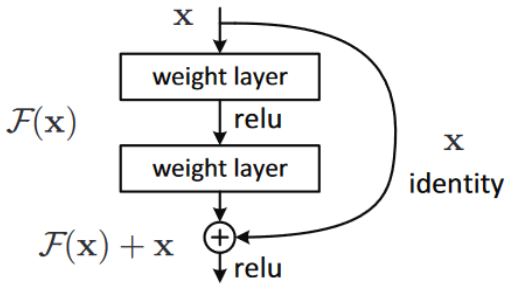

Figure 2.8: A building block for residual learning [HZRS16].

and better accuracy at a bigger depth parameter [HZRS16]. The idea is that shortcut connections skip some layers by connecting activations of one layer to further layers. This means that the output of the previous layer is added to the layer ahead, also known as *identity mapping*. Identity mapping does not have any parameters. The approach is that optimizing the residual mapping is easier than optimizing the original mapping. Therefore, $H(x)$ represents the underlying mapping and the input of the first layer of the underlying mapping as x. The residual function is defined as $F(x) := H(x) - x$ and the original mapping $H(x) := F(x) + x$ [HZRS16].

$$y = F(y, \{W_i\}) + x \tag{2.5}$$

Equation 2.5 shows the definition of a building block, where x is the input and y is the output of the stacked layers. The residual mapping is defined as $F(x, \{W_i\})$. The example in Figure 2.8 shows a block with two layers. Due to this reason, the function is defined as $F = W_2\sigma(W_1x)$, where σ refers to the ReLU activation function. To simplify the notation, the biases are left out. An element-wise addition for $F + x$ is calculated by the shortcut connection. For this calculation, F and x need to be the same size. If this is not the case, a linear projection W_s like zero-padding can be performed. Equation 2.6 shows the adaption of the linear projection to perform in the same dimension [HZRS16].

$$y = F(y, \{W_i\}) + W_sx \tag{2.6}$$

2.1.5 Feature Pyramid Network

Detecting objects at different scales is a major challenge in image recognition. A solution for this would be pyramid representations, but they are memory-intensive and time-consuming. Another solution is a *Feature Pyramid Network* (FPN). It consists of a *bottom-up* and a *top-down* pathway [LDG+17]. The architecture of FPN is shown in Figure 2.9 and 2.10.

To construct the bottom-up pathway, ResNet or VGG is used. It is the feedforward computation of the backbone ConvNet and is used for feature extraction. It creates a feature hierarchy consisting of feature maps. The spatial dimension is reduced by *0.5*.

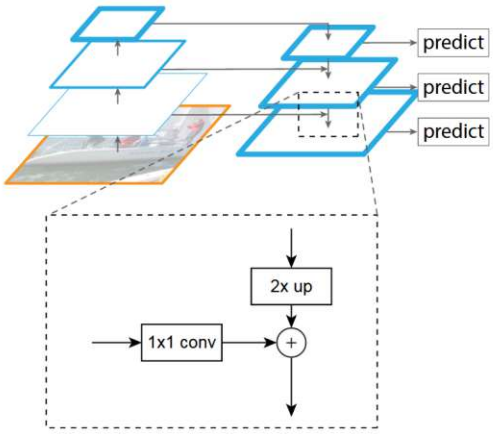

Figure 2.9: Feature pyramid network [LDG+17] .

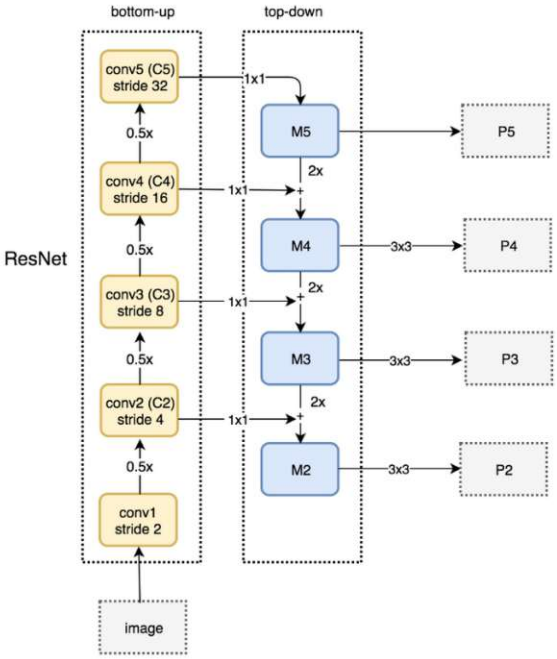

Figure 2.10: A detailed architecture of FPN[Hui18b].

When layers produce an output with the same size, they are in the same network stage. The feature pyramid has one level for each stage. The feature map is the output of the last layer [LDG+17].

The top-down pathway generates the final set of feature maps. Therefore, the spatial resolution is upsampled by a factor of *2*. Then, the feature maps of the same size

from the bottom-up and top-down pathways are merged with an element-wise addition. Furthermore, the channel dimension is reduced by a *1x1* convolutional layer. These steps are iterated until the last feature map is calculated [LDG+17]. The last layer is skipped because the spatial dimension is too large. To reduce the aliasing effect of upsampling, a *3x3* convolution is added to each merged map, to generate a prediction or the final set of feature maps (P_2, P_3, P_4,P_5).

2.1.6 Region Proposal Network

One of the most time-consuming tasks in real-time object detection has been the *region proposal* algorithms for estimating the location of the objects. An example is the *Selective Search* algorithm [UVDSGS13]. It takes *2* seconds per image in a CPU implementation. To solve this problem, Ren et al. [RHGS15] introduced *Region Proposal Network* (RPN). The main idea of RPN is to share convolutions during the test time to reduce the computing proposals. This method reduces the required calculation time to only *10*ms per image [RHGS15]. It is a fully convolutional network that generates bounding boxes with an objectness score at each position. It can also be integrated into any object detection network.

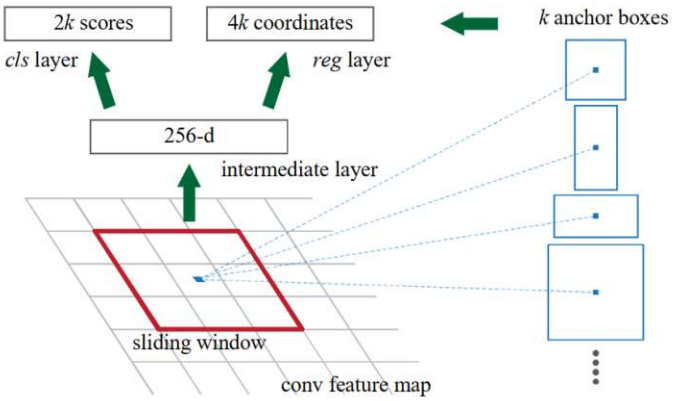

Figure 2.11: Region proposal network [RHGS15].

In Figure 2.11 an illustration of the network at a single position is shown. First, RPN generates *anchor boxes* for every *anchor point*. An anchor point is a point in the feature map generated by the last layer of a backbone network (for example CNN). An example is a VGG16 model, where the input could be an RGB-Image with dimensions of *224*x*224*x*3* (width, height and channels). The generated feature map of the last layer would have a size of *1/16* of the original image. In this example, the map has a size of *14*x*14*x*512*, where *512* refers to the number of feature maps. An *n*x*n* spatial window of the feature map is taken as input to RPN. For the input in Figure 2.11, a default kernel is shown as a *3*x*3* spatial window, which is also a *3*x*3* convolutional layer. The anchor point is always in the center of a window. Due to this reason, the scale and aspect ratio have to be taken into account. By default, the value *3* is used for scales and aspect

ratios to generate $k = 9$ anchors. To use the above example, a feature map, with the dimension of $14 \times 14 \times 9$ anchors, is used. The maximum amount of possible proposals are represented by k for each location [RHGS15]. Further, in Figure 2.11 the layer *256*-d represents a lower dimensional feature, where the sliding windows are mapped to it. The *box-regression layer* (reg) has $4k$ outputs used to detect the position of k boxes and the *box-classification layer* (cls) has $2k$ scores to classify the box. Both are fully connected layers. For training, each anchor has a binary class label whether it is an object or not. Only anchors with the highest Intersection-over-Union (explained in Section 2.1.8) or an overlap higher than *0.7*, receive a label [RHGS15]. The position difference is used as input for the cross-entropy loss, and the robust loss function (smooth L_1) to learn offsets. To obtain a *Region-of-Interest* (RoI), the anchor boxes with a high confidence score are modified with the learned offsets (also called proposal generation) and then forwarded to a RoI pooling layer (explained in Section 2.1.8).

2.1.7 Faster Regional Convolutional Neural Network

In the previous Chapter 2.1.6 RPN was introduced, where almost cost-free region proposals can be made. RPN was first mentioned by Ren et al. [RHGS15]. To improve *Fast Region-Based Convolutional Neural Network* (Fast R-CNN) [Gir15], which was already an improvement of *Region-Based Convolutional Neural Network* (R-CNN) [GDDM14], the new *Faster Region-Based Convolutional Neural Network* (Faster R-CNN) with RPN was invented.

To provide insight into R-CNN and Fast R-CNN, both are used for object detection and semantic segmentation. In semantic segmentation, each pixel is associated with a category or a label. Figure 2.12 shows the pipeline of R-CNN.

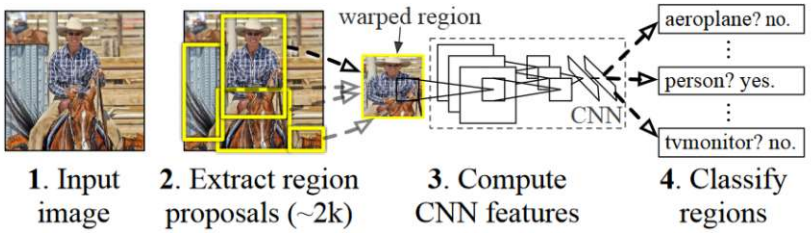

Figure 2.12: Region-based convolutional neural network [GDDM14].

R-CNN has an image as input. Then regions or boxes are generated, where an object could be located. For this purpose, a selective search [UVDSGS13] algorithm is used [GDDM14]. This algorithm works with texture, size, shape and color. These properties are used to create sub-segmentations that could belong to an object [UVDSGS13]. About *2000* bottom-up region proposals (bounding boxes) are generated with selective search. Then, the patch of the image within a bounding box is subtracted and warped to a 227×227 pixel size image, to be compatible with the CNN. After that, a *4096*-dimensional feature vector is extracted from each warped bounding box [GDDM14]. Finally, each region is classified using a class-specific linear *Support Vector Machine* (SVM) [HDO+98].

A support vector machine algorithm attempts to distinguish categories with a hyperplane. This hyperplane is in an N-dimensional space to classify the data points, where N is the number of features.

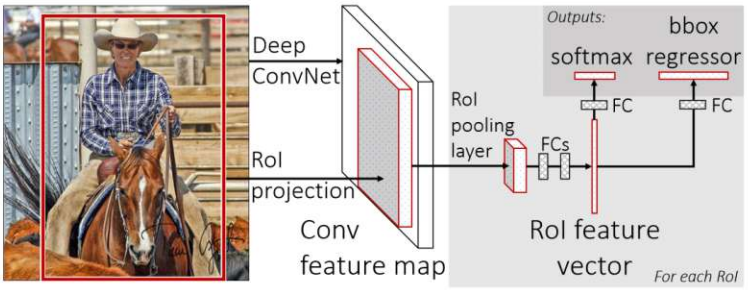

Figure 2.13: Architecture of fast region-based convolutional neural network [Gir15].

Fast R-CNN is different from R-CNN. Figure 2.13 shows the architecture of a Fast R-CNN. The idea of Fast R-CNN is, that the image is the input for the CNN (for example a VGG-16 model) and not the region proposals. As a result, a convolutional feature map is created. Further, a set of object proposals is still required for the Region of Interest (RoI) pooling layer. It is calculated for example by selective search. After the feature map is generated, the RoI pooling layer extracts feature vectors from the feature map. These vectors are fixed in length and are generated for each object proposal. Therefore, it is possible to obtain vectors of the same size from rectangles with different sizes. The RoI pooling layer maps the region proposal to the corresponding feature map and divides the section into fixed sizes. Then, the largest value in each section is copied to the output buffer. The RoI feature vector is then used for the softmax and bbox regressor layer. Thus, the network has two outputs. The softmax layer predicts one category or one class of the proposed region and the bbox regressor predicts one bounding-box regression offset per class [Gir15].

The faster solution to Fast R-CNN is Faster R-CNN [RHGS15], which is an extension of Fast R-CNN. The main difference is, that instead of selective search algorithms it uses a region proposed network (RPN), which is explained in Section 2.1.6. These algorithms are slow and time-consuming. Figure 2.14 shows the architecture of Faster R-CNN.

Faster R-CNN consists of two modules: RPN and Fast-RCNN [RHGS15]. It takes the full image as input to a convolutional network, that calculates a feature map. This map is forwarded to the RPN, which predicts region proposals. Then, the region proposals are reshaped with the RoI pooling layer. At the end, the reshaped proposed regions are classified and the offset of the bounding boxes is predicted.

2.1.8 Mask Region-Based Convolutional Neural Networks

Mask Region-Based Convolutional Neural Networks (Mask R-CNNs) are extensions of Faster R-CNN. The goal of Mask R-CNN is to perform *instance segmentation* [HGDG17].

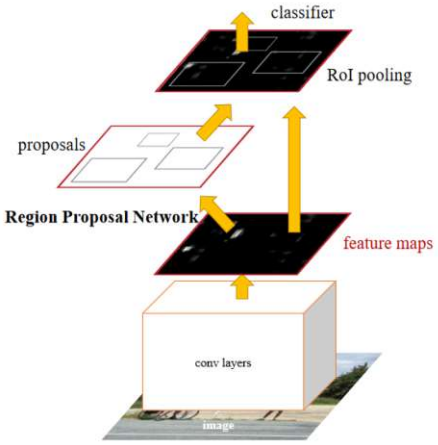

Figure 2.14: Architecture of faster region-based convolutional neural network [RHGS15].

There are two different segmentation tasks: *semantic segmentation* and *instance segmentation*. In semantic segmentation, pixels are grouped and labeled with a class such as pedestrians or a car. There is no difference between multiple objects, thus objects of the same class form a single entity and are not distinguished from each other. Instance segmentation differs between multiple objects like pedestrians. Figure 2.15 shows the architecture of Mask R-CNN that performs instance segmentation.

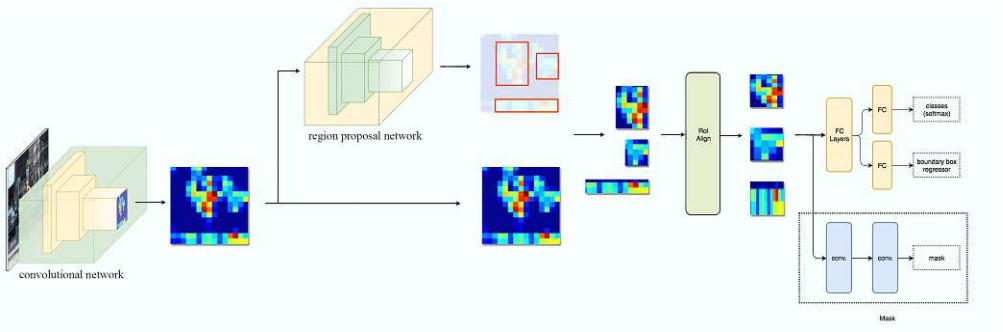

Figure 2.15: Architecture of mask region-based convolutional neural network [Hui18a].

Faster R-CNN and Mask R-CNN differ in two main parts. The first part is that Mask-RNN does not use a RoI Pool-Layer. Instead, a RoI-Align layer is used. In the RoI Pool-Layer, the boundary of the feature vector is aligned with the boundaries of the input feature map. Hence, the cells may have different sizes, which would lead to misalignments between the extracted features and the RoI [RHGS15]. As a result, pixels would not be classified or classified incorrectly and the mask would therefore be inaccurate. The RoI-Align layer solves this problem. It aligns the extracted feature with the input more properly and eliminates quantization. It also calculates the values for the feature map by interpolation. Figure 2.16 shows a RoI-Align example [RHGS15].

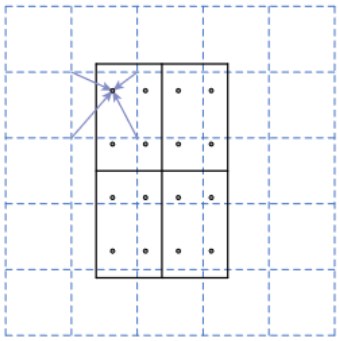

Figure 2.16: RoI-Align process [RHGS15].

The second part is that Mask R-CNN has two additional convolution layers after the RoI-Align layer. These two layers create the mask. To explain the architecture in detail, the input is an image for the backbone, which is a CNN (for example ResNet50 or ResNet101). This backbone detects, for example, edges and corners as low-level features and as higher-level features, for example, pedestrians or flowers. Also, an FPN is used to share features at different levels. This has to be mentioned because in this work a Mask R-CNN is used with a Resnet50 and FPN backbone. After the feature map is generated, the RPN is used to search for regions, that contain objects. Moreover, RPN reuses the extracted features as it uses the backbone feature map. As an output, candidate object bounding boxes are given. Then, the RoI-Aling layer aligns the bounding boxes to the feature map without quantizations and calculates the values by interpolation. Finally, the last convolution layers generate the mask, class and offset of the bounding boxes [RHGS15].

2.1.9 Intersection over Union

Intersection over Union (IoU), also called *Jaccard Index*, measures the overlap between the ground truth and the prediction. To calculate IoU, the area of overlap, of either the bounding boxes or the mask segmentation, is divided by their area of union [Cos21]. Figure 2.17 shows a visual representation. In addition, the IoU score is normalized, meaning that the value is between *0.0* and *1.0* [RTG+19]. For example, if the predicted bounding box overlaps exactly with the ground truth, the IoU score is *1.0*. This is the most optimal and means that both bounding boxes fully overlap. On the other hand, a score of *0.0* means no overlap. In general, an IoU score, which has a value greater than *0.5*, is considered a good prediction. Three examples are shown in Figure 2.17. Further, two different Intersection over Union metrics were used:

- bbox - Bounding Box
- segm - Mask Segmentation

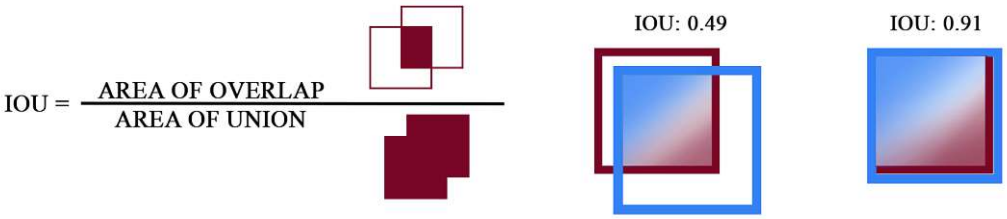

Figure 2.17: Visual representation of Intersection over Union and examples.

For the bounding box metric, the COCO evaluator measures the overlap between the predicted bounding box and the ground truth. The ground truth boxes are generated boxes of the hand-labeled mask, wherein the image of the object is. Furthermore, the metric mask segmentation measures the overlap between the hand-labeled mask representing the ground truth and the predicted mask [iC]. An example is shown in Figure 2.18.

Figure 2.18: Metric - segm: the ground truth is shown on the left side and the prediction on the right side.

Additionally, a predicted bounding box is considered as *True Positive*, if the IoU score is greater than *0.5*. Since a single reached threshold can induce a bias in the evaluation metric, the COCO evaluator averages the mean average precision of *80* classes over ten *True Positive* thresholds. Those thresholds are from *0.5* to *0.95* with a step size of *0.05*. Furthermore, different metrics for the area measured in pixels, are presented [iC]:

- *small* = area $<$ *32x32*
- *medium* = *32x32* $<$ area $<$ *96x96*
- *large* = area $>$ *96x96*

Different techniques for detecting objects with an image as input have been explained and presented. In this thesis, cameras are used to capture images of the robot and foot. The next section will give an overview of the camera calibration.

2.2 Camera Calibration

In this work, one of the main components is the camera. This device captures a three-dimensional image and converts it into a two-dimensional one. There are some distortion problems to solve before an image can be used for this work. In this section, insights into camera distortion effects are given. Further, camera parameters and methods to undistort an image are presented.

2.2.1 Distortion Effects

The pinhole camera has some advantages over a camera with lens optics. The advantages are a wide angular field and no problems with linear distortions, the disadvantage would be its resolution [You89]. In this work, resolution matters and therefore the images had to be undistorted. There are two main distortion effects: *radial* and *tangential lens distortion*.

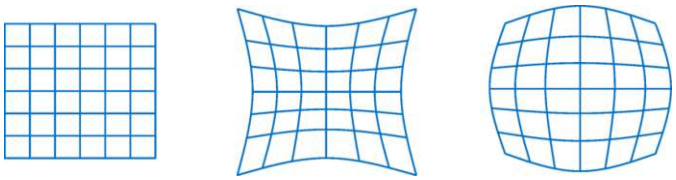

Figure 2.19: Radial distortion effects. Left: normal grid; Middle: pincushion distortion (negative radial distortion); Right: barrel distortion (positiv radial distortion) [DPWJ22].

Figure 2.20 shows the radial distortions where a straight line in the real world is not straight in an image. Two examples are *barrel distortion* and *pincushion distortion*. These distortions occur when the light is unequally bent. For this reason, the light rays bend in the optical center less than near the edges of a lens. The distortion increases with the size of the lens. Moreover, if the size of the lens decreases, the distortion increases [CV14].

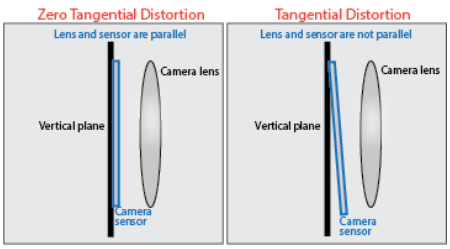

Figure 2.20: Tangential distortion [MD18].

Another distortion effect is tangential lens distortion. This distortion occurs when the image plane and the lens are not parallel, the reason being that the lens is not centered. As a result, the image plane is skewed and objects would look farther away than they really are.

2.2.2 Camera Parameters

There are two main categories of camera parameters: *extrinsic* and *intrinsic*. The extrinsic parameters form a *4x4* transformation matrix. It is used to transform points from world space to camera space. Consequently, if the physical position or orientation of a camera has changed, the extrinsic matrix will also change. This leads to the extrinsic matrix is being ignored in this work. The usual coordinate systems for computer vision are: *World Space –> Camera Space –> Pixel Space* [HS97].

The intrinsic parameters form a *3x3* matrix and convert points from camera space to pixel space. It consists of the focal length (f_x, f_y) and the optical center c_x, c_y. The focal length measures the distance between the pinhole and the image plane in pixels. Also, the intersection between the image plane and the principal axis of the camera is represented as the optical center. The intrinsic matrix is used to undistort the images and can be defined as $\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$. The mathematical representation of the radial and tangential distortions are shown in Equation 2.7 [HS97].

$$\begin{aligned} x_{radial} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{radial} &= y(1 + k_1 * r^2 + k_2 r^4 + k_3 r^6) \\ x_{tang} &= x + 2p_1 xy + p_2(r^2 + 2x^2) \\ y_{tang} &= y + p_1(r^2 + 2y^2) + 2p_2 xy \end{aligned} \tag{2.7}$$

The tangential distortion coefficients are represented as p_1, p_2, the radial distortion coefficients as k_1, k_2, k_3 and $r^2 = x^2 + y^2$. (x, y) is a point in a distorted image, which is normalized by dividing it by the focal length and converting it to the optical center. To undistort an image, the distortion coefficient must first be calculated. Then the radial and tangential components from Equation 2.7 are combined. Finally, an interpolation method is used, since the algorithm for undistortion can produce non-integer values [Mat].

2.3 Fiducial Marker

In this section, the basics of fiducial markers and the scope of applications are presented. Furthermore, a short description of the detection method is explained and examples of different variations are shown.

2.3.1 Basics

Object detection combined with real-time pose estimation is a time-consuming task. For this purpose, complex neural networks are often used. Another solution is the detection of *fiducial markers*. According to Fiala [Fia05], fiducial markers are used in a scene to locate point correspondences between an object or model and images. They are often used for drones to land at a specific location or for augmented reality. Different marker variants like *aruco marker* [KYW18], *ARToolKit* [KB99] or a *QR-Code* [Soo08] are shown in Figure 2.21.

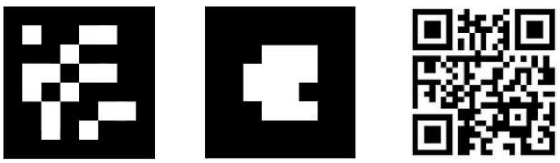

Figure 2.21: Examples of fiducial markers. 1: aruco marker; 2: ARToolKit; 3: QR-Code.

These markers can be used as a reference point to estimate the distance or scale of an object. In addition, the orientation of an object, like a robotic arm, can be detected. It is also possible to identify objects with unique markers. These markers have a pattern, that is distinct enough to distinguish them from their surroundings. It should also be possible to distinguish them from each other. The pattern is robust against lightning, blurring and partial occlusion [Fia09].

2.3.2 Detection

To detect an aruco marker, the border of the marker has to be detected first, for example. Therefore, quadrilateral contours have to be found in an image. An edge-based method such as the *Canny Detector* [DG01] is used and then these border pixels are linked into segments and grouped into *quads* to create a homography mapping for the interior of the marker [Fia05]. Another method for detecting the border is to fulfill image binarization and then discard all polygonal shapes, that are not squares [LES20]. After that, the candidate markers are analyzed by detecting black and white pixels and segmenting the marker image into cells. Further, the pixels are counted and every marker is assigned to a certain map based on the ratio of white and black pixels [LES20]. An example of a fiducial marker detection is shown in Figure 2.22.

2.4 Robot Kinematics

In this section, an overview of robot kinematics is given. Some basics about robotics are explained and methods to calculate the position of a robotic arm are presented. This section focuses on a robotic arm since this type of robot was used for this work.

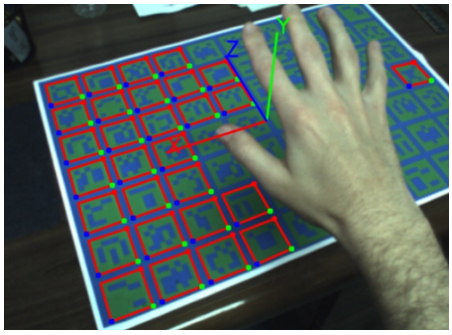

Figure 2.22: Example of aruco marker detection [GJMSMCMJ14].

2.4.1 Basics

Robots appear in various shapes and colors. Examples are a small moving robot or a robotic hand. The focus is on the robotic arm, which was used for this work. In Figure 2.23 a comparison between a robot arm and a human arm is shown.

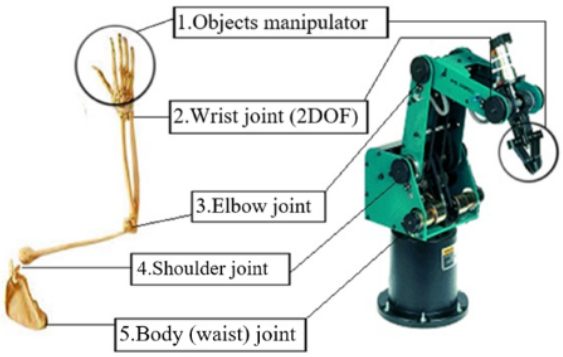

Figure 2.23: Human arm and a *5*-DOF robotic arm [ML18].

Degrees of Freedom (DOF) is a parameter that describes the motion capability of a robot. It represents the number of independent displacements. To calculate DOF for robot arms, the structure of a robot manipulator must be understood. The structure of a robot arm consists of *links* and *joints* [SSVO09]. A link is a rigid body, that does not deform when a force is applied. A joint is connected between at least two links. Hence, it is possible to apply motion to the connected links. A chain of joints and links forms a robotic arm. The shoulder and elbow of a human arm are joints and the bone between these joints is a link. According to Siciliano et al. [SSVO09], an arm (mobility), a wrist (dexterity) and an end-effector (performs task) form a manipulator. Joints can be classified into many different types like *prismatic* and *revolute*. In Figure 2.24 a kinematic diagram is shown where *R-joint* is a revolute joint and *P-joint* a prismatic one. Prismatic joints apply translational motion between the connected links, while revolute joints apply rotational motion. This work focuses on the *revolute* joint. Both types provide a single DOF. An

object in space has *6*-DOF. They are divided into *3*-DOF for positioning and the other *3* for orienting the object. It is possible that a robotic arm has more than *6*-DOF but then it is called *redundant* [SSVO09].

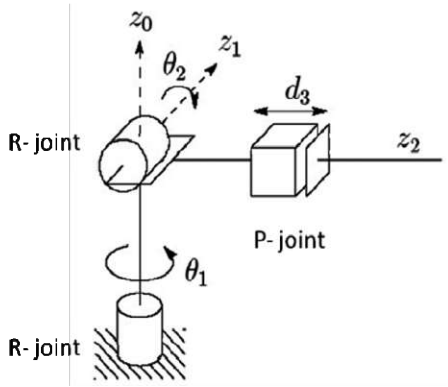

Figure 2.24: Kinematic diagram with revolute and prismatic joints [AAE10].

In order to program and move a robot arm without considering forces, kinematics is required. Kinematics can be divided into two main categories: *forward* and *inverse* kinematics. This work focuses on inverse kinematics [KB06].

2.4.2 Forward Kinematic

According to Kucuk and Nigul [KB06], the forward kinematics is straightforward with no complexity in deriving the equations. Moreover, there is always a solution for the manipulator. Forward kinematics focuses on calculating the position and orientation of the end-effector. This, the joint angles are the input for the calculation of the cartesian orientation and position. One method for solving this problem is the *Denavit-Hartenberg* method [Cor07]. For this purpose link length, link twist, link offset and joint angles are used. In addition, each joint has a coordinate frame and homogeneous transformation matrices are calculated.

2.4.3 Inverse Kinematic

The main difference between forward and inverse kinematics is, that forward kinematics calculates the position with the joint angles as input, while inverse kinematics calculates the joint angles with the position of the end-effector as input. To solve the inverse kinematics problem, the *Jacobian matrix* can be used. This is an iterative method like the *Gradient Descent Method* [GS93], which can be computationally expensive. The steps to calculate the target positions are divided into three. First, the joint configuration is determined, then the rotational changes are computed and finally, the Jacobian matrix is calculated.

$$X = f(\theta) \tag{2.8}$$

$$\theta = f^{-1}(X) \tag{2.9}$$

According to Meredith and Maddock [MM04], the forward kinematic, presented in Equation 2.8, is derived to calculate the Jacobian matrix. Therefore θ is the set of orientation values and X is the global position of a link. Equation 2.9 represents the inverse kinematics.

$$dX = J(\theta)d\theta \tag{2.10}$$

where

$$J_{ij} = \frac{\partial f_j}{\partial x_i} \tag{2.11}$$

$$d\theta = J^{-1}dX \tag{2.12}$$

Equation 2.10 represents the partial derivatives of the forward kinematics. Equation 2.12 shows the form for inverse kinematics, which requires the inverse of the Jacobian matrix. Meredith and Maddock [MM04] used an incremental change of each joint. As long as the end-effector is not located at the target position, this is the starting. In each iteration, the distance between the current position of the end-effector (X) and the target position (X_d) is updated. The partial derivatives of the joint angles (θ) are also updated for each iteration. The Jacobian matrix is represented as J [MM04]. A simplified algorithm of Meredith and Maddock [MM04] is shown below:

1. The position of the end-effector is taken for calculating the distance to the target position $dX = X_g - X$.
2. Equation 2.11 with the joint angles is used to calculate the Jacobian matrix and to invert it.
3. New joint orientations are calculated and set as current values $\theta = \theta + J^{-1}dX$.
4. The position of the end-effector is checked with forward kinematics, if it is close enough to the target position. This is considered to be the case, when the solution is below a predefined threshold, leading to the algorithm getting terminated. If this is not the case, the calculation restarts at step 1.

2.5 Foot Massage

In the previous sections different object recognition techniques, camera calibration and kinematics were presented. This section focuses on the massage techniques used for this thesis. A massage is used to promote relaxation and treat painful muscular complaints [VZ99]. Different parts like the back, shoulders, feet, etc. can be focused. Further, a five-minute foot massage can reduce heart rate and blood pressure [HJ99]. To achieve these

results, different techniques like *kneading*, *frictioning*, *stroking*, etc. can be used. This work focuses on foot massages using the following techniques: stroking and frictioning. These two techniques were chosen because the architecture of the robot used in this work was capable of performing them.

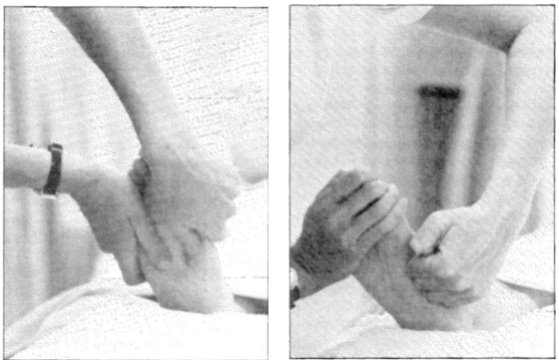

Figure 2.25: Frictioning and stroking [Joa83].

The friction method is a deep massage. During friction massage, the thumb or a fingertip is moved over the sole in a circular motion [Joa83]. It is also called *cross-friction massage*. The main purpose of this technique is to maintain the mobility of soft tissue structures [Cyr75]. In Figure 2.25 frictioning and stroking is presented. Stroking is a technique in which the muscles are stretched. It involves one hand that performs an up-and-down motion on a sole [Joa83]. It is also known as *Effleurage*, where one hand strokes along the length of a muscle [VZ99]. These are the most important techniques for this thesis. In the next chapter the design of the prototype will be presented.

CHAPTER 3

Design

In this chapter, the requirements and design of the prototype are discussed. First, Section 3.1 lists all requirements with a short description. Then the hardware and software architecture is shown in Section 3.2 and 3.3. At the end of this chapter, the methodology of this work is presented in Section 3.4.

3.1 Requirements

The main goal of this thesis is to build and develop a robotic arm that performs a foot massage. To achieve this goal, the prototype is divided into several requirements. These requirements are defined as follows:

- **Hardware Setup:** Foot recognition is a main part of this thesis. Therefore, cameras are required and placed in the correct locations. To prevent the robot from breaking out, it has to be mounted on a stable surface, a plate for example. Also, the robot requires to reach the sole of the foot, due to this a footrest needs to be built. An optimal hardware setup would be transportable.
- **End-effector Modification:** Usually, a foot massage is performed by hand. However, in this thesis a robot which has a claw as an end-effector, is used. In order to provide a more comfortable massage, the end-effector has to be modified. Further, additional hardware has to be implemented to detect contact between the foot and the robot.
- **Foot Recognition:** Another requirement is foot recognition. For this purpose, a computer vision framework such as OpenPose should be used. Since OpenPose does not work for the bottom of a foot, a Mask R-CNN (explained in Section 2.1.8) has to be trained.

- **Distance Estimation:** A distance estimation function could be used to establish contact between the foot and the robot. As mentioned before, additional hardware has to be implemented, to make sure that the robot has physical contact.
- **Robot Movement:** A function has to be implemented, that is responsible for moving the robot from one position to another. For this purpose, forward or inverse kinematics (explained in Section 2.4) could be used.
- **Massage Techniques:** There are different massage techniques for the sole of a foot. Since the robot used in this thesis is limited, a simplified frictioning and optionally a simple stroking function could be implemented.
- **Audio Recognition:** To ease control of the robot and to improve comfort of the user, audio commands which remove the need to interact via mouse and keyboard, should be implemented. Commands such as *GO* and *STOP* for example.

These are the requirements to fulfill. In the next two sections of this chapter, an overview of the hardware and software architecture is presented.

3.2 Architecture Hardware

This thesis is overall divided into two main parts: hardware and software. For this reason, software and hardware are of equal importance. The hardware part includes the physical contact between the foot and the robot, as well as modifying the end-effector, constructing a footrest, building a transportable setup and establishing communication between the components. Figure 3.2 shows the communication of the hardware architecture used in this thesis.

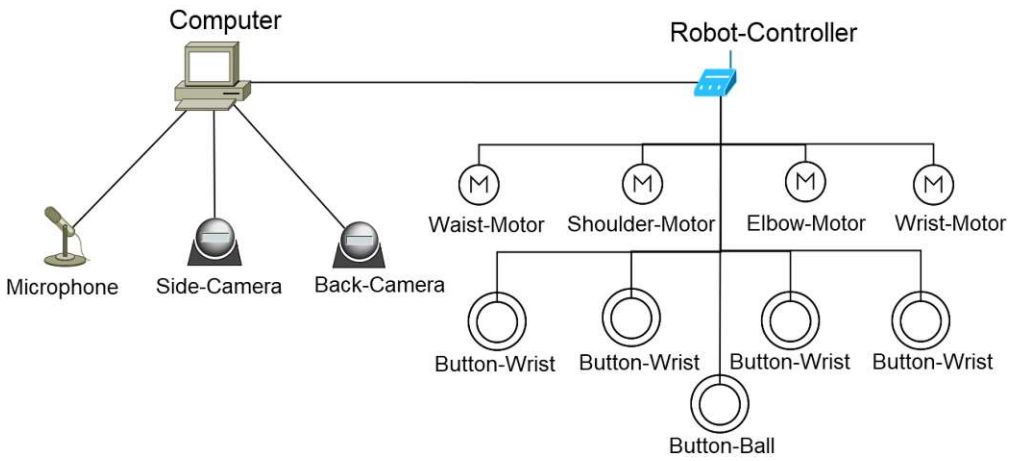

Figure 3.1: Communication model of the hardware architecture. The robot controller is an Arduino.

As it can be seen in Figure 3.2, the computer is connected to four devices: two cameras, a microphone and an Arduino. The cameras are used for capturing frames. These frames are then processed to detect and recognize the foot and the robot. Another component is the microphone. It is used to record the audio commands given by the user. This device is optional since the webcams have a built-in microphone. The advantage of having an external one is that the environmental noise is reduced and the quality of the audio samples is higher. To move the robot into the correct positions, the servomotors have to be controlled. In this case an Arduino, which is a small computer, is used. To detect physical contact between the foot and the robot, buttons are utilised and connected to the Arduino. A detailed explanation of the construction and implementation of the hardware is given in Chapter 4.

3.3 Architecture Software

In this section, an overview of the software architecture is presented. There are two main components, which are divided into the main program for the PC and the main program for the Robot-Controller (Arduino). Figure 3.2 shows the software modules.

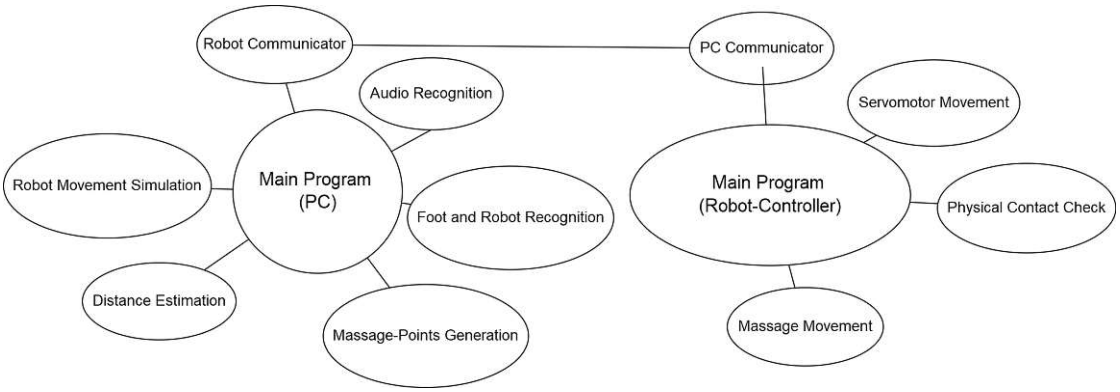

Figure 3.2: Communication model for the software architecture.

First, the modules of the PC are introduced with a short description. After that, an overview of the Arduino modules is given. A serial cable was used to connect these two devices. The software architecture of the PC is divided as follows:

- **Audio Recognition:** This module consists of a CNN module, that uses time-domain waveforms as input [DDQ+17]. For this purpose, the input stream from the microphone is converted into a waveform and then audio commands are predicted. The commands are used to control and modify the movement of the robot such as *GO*, in order to begin the massage.
- **Foot and Robot Recognition:** To recognize the foot and the robot, a Mask R-CNN is trained and used. Frames are extracted from the camera streams as

input, and afterwards analyzed to receive a mask and a bounding box from the related objects as output.

- **Massage-Points Generation:** Massage points are the movement targets of the robot. Thus, the points have to be generated and mapped on the sole of the foot. Predefined points from a foot sole template are mapped on to the actual foot. Moreover, a distinction is made between the left and right foot because the longitudinal arch on the foot is not on the same side. If this is not taken into account, the robot could miss the sole of the foot.

- **Distance Estimation:** The distance estimation function is required to measure the distance between the actual massage point and the robot. Consequently, an aruco marker is used for the back camera and a checkered board for the side camera. With two known points and the actual distance in world space, it is possible to map pixels to mm.

- **Robot Movement Simulation:** This module represents a 3D simulation of the robot used in this thesis. The distance estimation function is used to map the current massage point into the 3D simulation. The end-effector moves in the simulation with inverse kinematics to the mapped massage point. The angles of each joint are the outputs for the robot communicator.

- **Robot Communicator:** The communicator is responsible for sending and receiving messages between the PC and the Arduino (robot). There are several commands for communication such as if the robot is ready and connected or the calculated angles for the servomotors.

After resolving the recognition tasks and the motion simulation of the robot, the calculated angles and movement type are sent to the Arduino. This computing unit is mainly used to control the servomotors with the received angles. The modules of the Arduino are divided as follows:

- **PC Communicator:** In this module the Arduino translates received messages from the PC and extracts information. In addition, new messages are sent back as a response.

- **Servomotor Movement:** Each servomotor has to be controlled separately. Due to this reason, the angles have to be mapped into pulses. Further, every motor has an individual pulse range that represents the minimal and maximal possible rotation. This range is coordinated with each joint of the 3D simulation of the robot. The smoothing out and reducing the jerky movement of the robot while massaging, happens in this module as well.

- **Massage Movement:** Frictioning and stroking are two different massage techniques. With frictioning, the robot applies more pressure when physical contact

is achieved. This additional pressure is not applied for stroking. In this module, these two massage techniques are performed.

- **Physical Contact Check:** Physical contact is important for the massage. The mapped massage points from 2D into 3D are not always correct. The robot requires additional verification that physical contact has been made with the foot. Hence the status of each button is frequently analyzed.

The hardware and software architecture are needed, in order to fulfill the requirements. In the next section, the methodology used in this thesis, is presented.

3.4 Methodology

In this thesis, a vision-based foot massage robot has to be developed. Alternative, *prototyping* is used as the research methodology. It is divided into the following four parts:

1. **Literature Review and Research:** This includes the investigation of state-of-the-art foot massage techniques, that can be performed with the presented robot. In addition, various methods of recognizing the foot, audio commands and controlling the robot are being investigated
2. **Sketching and Designing:** This thesis has two parts to build and develop: the software modules and hardware components. Thus, it is necessary to sketch and design the prototype.
3. **Prototype Development:** The construction of the hardware prototype includes a footrest, a platform and a new end-effector. The software framework includes controlling the robot, recognizing feet and performing a foot massage.
4. **Evaluation:** To evaluate the performance of the prototype, a qualitative evaluation should be conducted. For this purpose, a user study should be developed in which test users sit in front of the prototype for about twenty minutes and receive a foot massage. Some questions have to be answered before and after the massage, and then the results are analyzed and evaluated

In this chapter, the requirements, the architecture of the prototype and the methodology are presented. The implementation of this project was based on these requirements and processes, as can be seen in the next chapter.

CHAPTER 4

Implementation

4.1 Overview of the Chapter

This chapter presents the implementation of the vision-based foot massage robot used in the user study, which is discussed in Chapter 5. In this thesis, a robot capable of performing a state-of-the-art foot massage will be designed. In order to achieve this goal, several steps are required. First, Section 4.2 presents the construction of the hardware setup. This includes the footrest, the platform, the modification of the end-effector and some features of the robot. After that, an explanation of image preprocessing, where the images are undistorted, is given in Section 4.3. One of the main parts of this thesis is the recognition and detection of feet. Therefore, a Mask R-CNN model was trained and used, which is shown in Section 4.4. Then, the distance estimation function for converting 3D objects from world space to pixel space and vice versa is described in Section 4.5. Performing a foot massage with a robot involves moving the end-effector to target points on the sole of a foot. An explanation of the generation of these massage points is presented in Section 4.6. Another main component of this work is the calculation and simulation of the inverse kinematics concerning the robot, which is shown in Section 4.7. An additional computer, in this case an Arduino, is required to control the robot. The communication between the computer and the Arduino is presented in Section 4.8. Furthermore, an explanation of controlling the servomotors of the robot is shown in the same section. At the end of this chapter, Section 4.9 describes the recognition of audio commands, where a CNN model is trained and used for this purpose.

4.2 Hardware Setup

4.2.1 Construction: Footrest and Platform

The hardware setup includes the construction of the footrest, the main platform, the camera platforms and the modification of the end-effector. In this subsection, the overall setup is explained first. Figure 4.1 shows the assembled prototype.

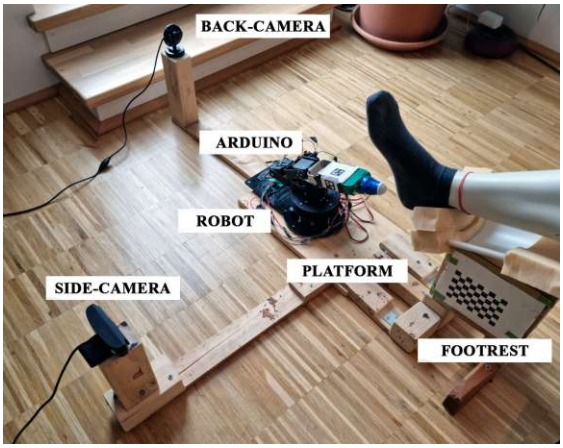

Figure 4.1: Hardware Setup.

The *back camera* and *side camera* have their own platform component, which is connected to the main platform via a plug-in system. This makes it possible to detach the camera platforms and get the setup to be transportable. One reason for the construction of the platforms is to get the cameras into the same spot every time. At the beginning of the massage, the cameras detect the foot and the robot. As can be seen in Figure 4.1, the end-effector could cover the sole of the foot if the back camera is in too low of a position. Further, the distance estimation function requires the detection of the aruco marker and the checkered board, which is explained in Section 4.5. Another component of the hardware setup is the main wooden platform, into which each wooden part can be plugged in. It is thus the center of the hardware setup. Also, the robot is attached to the platform, otherwise it would push itself away from the foot during the massage and the pressure could not be increased. In addition, the Arduino is also fixated on it. This computational unit is located on the left side behind the robot for safety reasons, as the robot requires additional cables connected to the top of the Arduino motor controller. If it were mounted directly behind the robot, a link could hit a plastic cover of a cable and break the connection as the base of the robot rotates.

Figure 4.2 shows the footrest component. A checkered board is attached to the same side as the side camera for the distance estimation function. To improve the stability of the footrest during the massage, additional wooden beams are mounted on the left and right sides. There are also three wooden beams at the front to prevent the footrest from twisting. The middle one is used to hold the platform and the footrest together to

Figure 4.2: Picture of the footrest with a mannequin foot.

prevent these components from being pushed away by the robot during the massage. At the top, a part of a pipe with foam glued onto it, is screwed together with the footrest. The shape of the pipe ensures that the foot stays in position and the foam changes the height of the foot, so that the sole is at an accessible angle for the massage.

4.2.2 Foot Massage Robot

A foot massage with a robotic arm is a challenging task. For this reason, the end-effector should be modified so that it does not injure the foot with its shape or surface. In this work, the claw of the robot has been changed as shown in Figure 4.3.

Figure 4.3: End-effector prototype.

The main idea for the new end-effector was to simulate a thumb. Therefore, a kickerball could be used, which is encapsulated in a 3D-printed case. In addition, only the case should be fixated on the arm, so that the ball can rotate freely in its case, when the robot massages the foot. To minimize the amount of work and the complexity of the design, a deodorant roller is used. The modification of the deo-roller is shown in Figure 4.3, which requires the following three steps:

1. A hole is drilled behind the ball, into which a button is inserted.
2. A small button is glued directly behind the ball. This is required to detect the physical contact between the foot and the robot.
3. Four additional buttons are attached to the aluminium link. Additionally, the head of the end-effector is glued only to the top of these buttons.

It is possible that the robot slips off the foot while applying pressure to it. Thus, it may happen that the button is not pressed and no physical contact is detected. To prevent this, the four additional buttons, which are bigger than the button behind the ball and mounted on the aluminium link, are hit more easily. No difference is made between all five buttons for detecting physical contact. They are connected to the Arduino.

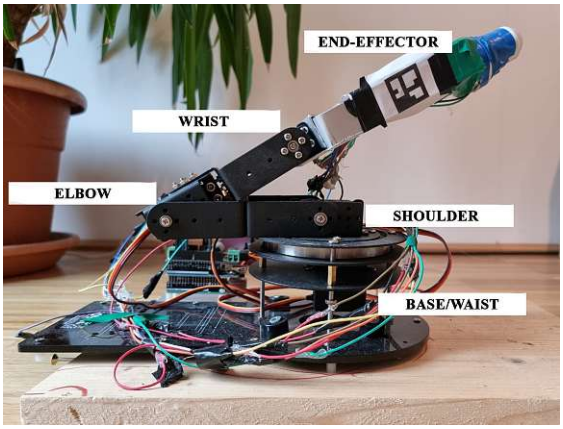

Figure 4.4: Robotic prototype arm.

In Figure 4.4 the complete prototype with additional markings for each joint is shown. The correlation between a real and a robotic arm is described in Section 2.4. The presented prototype is reduced to 4-DOF, to minimize the complexity of movement. Hence, a newly built aluminium link replaced the joint of the wrist.

4.3 Image Preprocessing

Before the images are used to detect the foot, robot, etc. they have to be preprocessed. This is useful to have more accurate distance estimation and recognition results. In order to achieve this goal, an undistortion method is used, which is explained in Section 2.2. First, the intrinsic parameters are required. To calculate them, the side and back cameras are configured before the massage begins. Each camera needs about twenty different images with a checkered board in it, taken by the camera itself. Then, for each image, the corners of the checkered board have to be detected. When the corners are found, the 3D points for the world space are simplified such as $(0,0,0)$,$(1,0,0)$ ... $(6,5,0)$ and

saved as object points. After that, all corners of a predefined number of rectangles on the checkered board are located in the image space. These are *2*D points. If all corners are found and each checkered board image has gone through these steps, the camera can be calibrated. For this purpose, an external library *OpenCV* [1] is used. As a result, the *3*x*3* camera intrinsic matrix and the distortion coefficients p_1, p_2, k_1, k_2, k_3 are calculated.

To undistort the image, the camera matrix, distortion coefficients and image size are used to compute an optimal camera matrix and a RoI (Region of Interest). Then, the optimal camera matrix and distortion coefficients are used for remapping and interpolation, which are presented in Section 2.2. A warped image is received as a result. In the final step, the warped image has to be cropped with the RoI box. Finally, the image is preprocessed and can be used for recognition tasks and distance estimation functions.

4.4 Mask Region-Based Convolutional Neural Network - Foot and Robot Recognition

For detecting the foot and the robot a mask region-based convolutional neural network (Mask R-CNN) is used, which is explained in Section 2.1.8. In addition, some changes have been made to the architecture. The input of a Mask R-CNN, such as an image, is forwarded to a convolutional network, also known as backbone structure. Section 2.1.8 presents a Mask R-CNN model with a ResNet backbone. In this thesis, a Mask R-CNN with a ResNet50 and an FPN (shown in Section 2.1.5) backbone is used. Figure 4.5 shows the pipeline of the Mask R-CNN, without resizing the image.

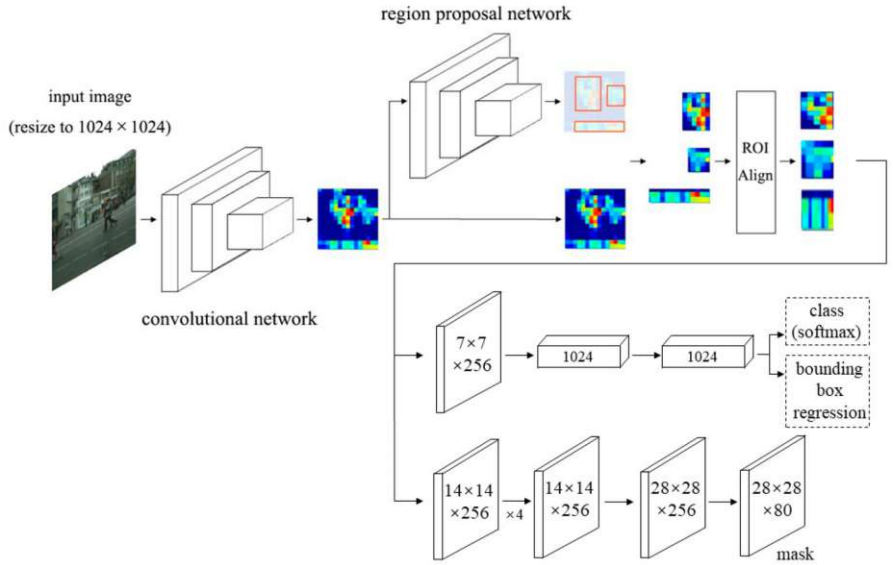

Figure 4.5: Pipeline of a Mask R-CNN without resizing the image [ZCLM20].

[1] https://docs.opencv.org/ (visited on 06.12.2022)

Specifically, the bottom-up pathway is the ResNet (convolutional network). Conversely, the top-down pathway generates feature maps that are a combination of lower and higher-level features to improve the accuracy of feature extraction. After that, all the feature maps from the top-down pathway are processed by an RPN (explained in Section 2.1.6). RPN is used to obtain candidate bounding boxes and can be divided into two paths. The first path obtains positive and negative classifications by classifying anchors. This classification is done by a softmax function [ZCLM20]. The parallel second path is to receive an accurate proposal for calculating the offset of a bounding box for each anchor. The last layer excludes all proposals that are too small. Moreover, RPN takes only the optimal scale from the FPN (P_2, P_3, P_4, P_5) [ZCLM20] to extract the RoI and passes it to the RoI-Align layer, which is explained in detail in Section 2.1.8. The RoI-Align layer extracts feature vectors from a feature map (P_2, P_3, P_4, P_5). These features are based on the RoI output of the RPN. In Section 2.1.8 a Mask R-CNN is presented with a simple ResNet rather than an FPN backbone. Therefore, the network *heads* differ from each other. The fifth stage of ResNet is computationally intensive. With an FPN backbone, this stage is already included. This is an advantage because fewer filters are required for the head and it is more efficient [HGDG17]. At the end, a mask, a class and the offset of the bounding boxes are generated.

To train such a Mask R-CNN, a dataset is required. This dataset contains images of three object categories consisting of more than *2400* labeled instances in *1440* images. Further, these images have different resolutions and both objects (robot and foot) are not always in the same one. Some were taken with different cameras as well. The dataset was split in the ratio *80*:*20*. *80*% are used for training and the other *20*% were taken for testing.

There are several ways to train such a model. In this thesis, a pre-trained model with a ResNet50 and FPN backbone was used and finetuned. For this purpose, the *head* of the Mask R-CNN model was replaced with a new one to recognize the foot and the robot. This approach was chosen due to the small dataset. Moreover, a detailed explanation of the implementation from PyTorch [2] was used. The results of the evaluation of the Mask R-CNN model are presented in Section 5.4.1.

4.5 Distance Estimation

4.5.1 Side Camera Distance Estimation

To perform a foot massage, several steps must be considered. First, the hardware setup has to be built. Second, the images from the camera are preprocessed to recognize the foot and the robot. And at third, the distance estimation function for the side and back cameras is performed and some other steps, that will be presented later on. This section is only about the distance estimation function.

[2] `https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html` (visited on 06.12.2022)

In the hardware setup of this thesis, the cameras are arranged orthogonally to each other. Each base has a certain height, so that the cameras have full sight of the foot and prevent the robot from covering parts of the foot. This is crucial in order to generate massage points, which is explained in Section 4.6. The robot has to move to these points to massage the foot. For this purpose, the distance estimation function estimates pixels to centimeters. This means that a 2D pixel is mapped to a certain centimeter value, which is then used as distance. This is important in order to map all 3D points of the real world to the simulation, which is presented in Section 4.7.

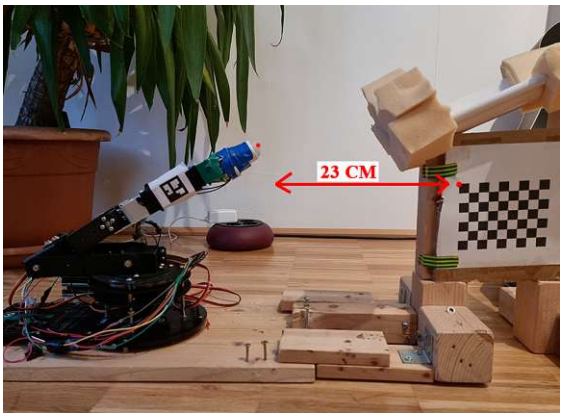

Figure 4.6: Fixed distance between the checkered board and the robot for the side camera.

For the distance estimation function, the real world distance between two known points is required. Therefore, the rightmost point of the end-effector of the robot and the upper left corner of the checkered board are used. With OpenCV it is possible to receive the corners of the checkered board. To find the rightmost point of the robot, the output mask of the presented Mask R-CNN model is taken. Then the real world centimeter distance (in this case *23* cm) is divided by the vertical distance (x-axis) in pixels between these points. Figure 4.6 shows the known points and the real world distance.

4.5.2 Back Camera Distance Estimation

Since the robot will not only move up, down, forward and backward, but also sidewards, the back camera is required. Due to this, a distance estimation function is required for the back camera. This function works similarly to the one of the side camera.

Figure 4.7 shows a visual representation of the known distance and the object point. To map a pixel to centimeters, a fiducial marker is required on the back of the robot. Moreover, an aruco marker is used as a fiducial marker. First, the aruco marker has to be detected and then the lower right corner is used. Then the topmost pixel of the robot mask from the Mask R-CNN model is taken as the second known point. For this reason, the default position of the robot is required. Hence, the real world distance (in this case *6.6 cm*) is divided by the horizontal distance (y-axis) in pixels between these two known points.

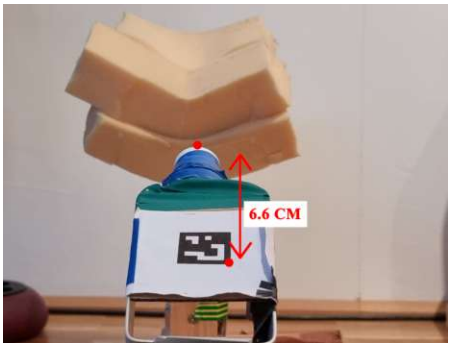

Figure 4.7: Fixed distance between the fiducial marker and the robot for the back camera.

4.6 Massage Point Generation

In the previous sections, the hardware setup, image preprocessing, foot recognition and distance estimation were presented. To perform a foot massage, the robot requires targets to move to. The generation of foot massage points is explained in this chapter. The basic idea for generating these points is to save a mask template of a sole and then mark positions on it. These marked points are saved and when a foot massage starts, the template sole with the marked points is loaded, rotated and warped to the actual mask of the sole. Figure 4.8 shows the template mask on the left with the predefined foot massage points (blue) and the actual foot mask with rotated and warped massage points. On the right, the actual foot mask with the modified massage points on the real-time frame of the back camera is shown.

Figure 4.8: Massage points generation: (left) template mask and actual foot mask with foot massage points; (right) actual foot mask on the real-time frame with the mapped foot massage points.

The process of generating these massage points can be essentially divided into two parts, namely the manual generation of massage points on a predefined sole mask and the loading of massage points onto the actual sole mask. Consequently several steps have to be considered:

1. **Generate template mask:** This step was performed once to generate a template mask. This was done by placing a mannequin foot on the footrest, then using Mask R-CNN to detect the mask of the sole and save it to the local drive in the project. A external library *Numpy* [3] was used for this step.

2. **Load the template:** To create new massage points, the sole of the foot has to be loaded into the program first. The loading process is also performed with *Numpy*. After that, it is shown in an extra window.

3. **Mark points:** When the extra window has loaded, it is possible to generate new massage points. For this purpose it is necessary to mark the points within the foot template. For each new point, all new massage points are saved locally in a file.

4. **Load points:** All previous steps are used to generate new massage points. The normal massage program includes the steps starting with this one. Therefore, the predefined massage points with the template mask are required to be loaded first. Further, the top and bottom points of the foot mask are saved and loaded as well. In Figure 4.8 on the left, the blue points are the loaded massage points as well as the top and bottom point of the foot mask.

5. **Translate template:** After detection and recognition of the actual foot mask with the Mask R-CNN, the top and bottom points of the new mask are calculated. Then the mask of the template is translated to the actual foot mask so that both bottom points are in the same position. The translation matrix $\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix}$ is used for this process.

6. **Rotate template:** In this step, the template mask is rotated. For this purpose, the arc cosine of the dot product of two unit vectors (bottom point - top point template foot/bottom point - top point actual foot) is calculated.

7. **Translate/Rotate massage points:** After the foot mask has been moved and rotated, the massage points are also translated with the same difference t_x and t_y. The rotation angle is the same as in the previous step and the points are rotated around the bottom point of the current foot mask.

8. **Warp massage points:** At this point in the process, all massage points are translated and rotated in relation to the actual foot mask. Since each foot has a different shape and size, the points have to be warped. First, the massage points

[3] `https://numpy.org` (visited on 13.12.2022)

are shifted in the y-direction with the difference of both foot sizes as a percentage. Then, the points are shifted with the percentage difference of the two foot sizes in the x-direction. The size of the foot in the x-direction is the point furthest to the left and furthest to the right of the foot mask.

9. **(Optional) mirror points:** All the steps above require the sole of a right foot. If the left foot is to be massaged, then all the massage points and also the top and bottom point have to be mirrored. To do this, the points are shifted by the difference between the width of the image and the actual x-position of a point. This step is done before translating the template in step *5*.

10. **Map to real-time frame:** The last step of generating the massage points is to map the points to the real-time frame. It is possible that the foot rotates during the massage. It is assumed that the bottom point of the foot mask never changes and therefore the massage points are rotated only to the top point of the newly detected foot mask. In Figure 4.8 on the right, the real-time frame with the mapped massage points on it, is shown on the right. Additionally, the translations and rotations are calculated with OpenCV [4].

4.7 Inverse Kinematics for Robot Arms

4.7.1 2D Simulation

One of the main parts of this thesis is to move the robot to the generated massage points. For this purpose, inverse kinematics is used within a 2D and a 3D simulation. In Figure 4.9, the 2D simulation of the robot is shown.

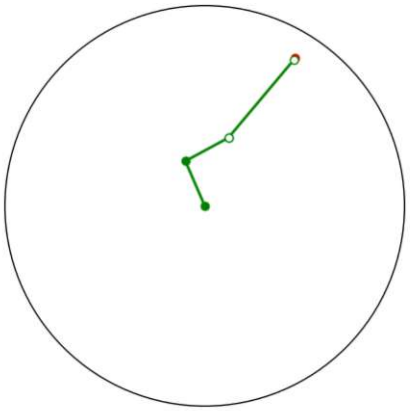

Figure 4.9: 2D Simulation of inverse kinematics.

The 2D simulation is the same view as the side camera. The base rotator is ignored because the robot only moves up, down, forward and backward in this simulation. The

[4] `https://opencv.org` (visited on 13.12.2022)

links have the same length as the robot in the real world. The movement of the robot in the simulation requires a target. Section 4.6 introduces the generation of massage points for the sole of the foot. These points are mapped for the back camera view. Since the 2D simulation has the same view as the side camera, the massage points must be mapped to that view. For this purpose, the top and bottom point of the foot mask of both frames (side and back camera) are required. The y-position of the massage point is mapped from the back view mask to the side view mask. After that, the leftmost point of the side view mask in the y-position (from the massage point) is taken as the x-position. The massage point is now mapped from the back view to the side view. Moreover, it is now located in a 3D position because the x- and y-positions are taken from the side view and the depth or z-position is the x-position of the back view.

Since the position of the massage point is calculated in pixels, the distance estimation function (explained in Section 4.5) is required to obtain the position and distance between the end-effector of the robot arm and the massage points in centimeters. The position of the end-effector is known by the detection of the Mask R-CNN. After that, it is possible to display the massage point in the 2D simulation.

To move the end-effector to the target point (massage point), the distance between those two is first calculated with numpy.linalg.norm (using Frobenius norm). This distance is always updated so that the position of the end-effector approaches the target position. Then the Jacobian matrix is calculated. Therefore, a new matrix (*3*x*3* - *3* for x,y position and *3* for the joints) is created, which is filled with zeroes. The position of the end-effector is subtracted from the position of the current joint (beginning with the base). After that, the cross product is calculated with the z-unit vector (*0,0,1*) and entered into the first column of the Jacobian matrix. Then the pseudo-inverse is calculated from the resulting Jacobian matrix. A dot-product of the x- and y-difference between the target and the end-effector is calculated afterwards. The results are angles, which are added to the existing angles of the joints. Further, the new angles are checked for violating policies.

If an angle violates the policies, it means that a servomotor cannot rotate to that angle due to physical limitations. If everything is fine, the positions are updated. For this purpose, the rotation matrix for the z-axis is used [Wei03]. The new distance between the end-effector and the target is calculated, and if it is below a certain threshold, the inverse kinematics calculation for the 2D simulation is finished. A detailed description is presented in Section 2.4.

4.7.2 3D Simulation

In the previous subsection, the 2D simulation of the robot was presented. In this subsection, the 3D simulation is shown. It is necessary because the base of the robot can be rotated to the left or right side, so that points near the edges of the sole can be reached. Figure 4.10 shows the 3D simulation of the robot.

For simplicity, the 3D simulation is only required for the rotation of the base of the robot. For this purpose, the robot in the start position and also the massage point is mapped in

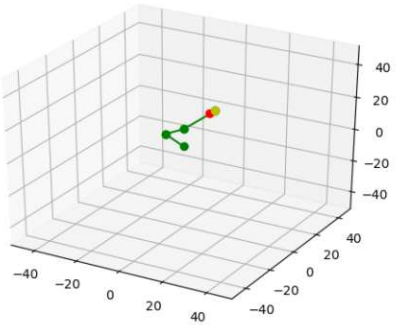

Figure 4.10: 3D Simulation of inverse kinematics.

the 3D simulation. The basic idea for calculating the inverse kinematics in 3D for the robot used in this thesis, is to rotate the massage point in front of the end-effector. The rotation angle is calculated using the unit vector of the target and the end-effector. Then the massage point is rotated around the base, located at the position $(0,0)$. This rotation angle is used to rotate the base servomotor. From there, the target point with its new position can be mapped into the 2D simulation presented in the previous subsection.

An advantage of being able to see a visual representation of the joints and servomotors in the simulation as well as real life, is to compare them both at the same time. It is possible to check whether a servomotor is rotating correctly or not. In the next chapter the communication between the robot controller and the PC as well as how to move the robotic arm in the real world, are presented.

4.8 Robot Communication & Movement

4.8.1 Communication PC

For this thesis, two devices are required: a PC and an Arduino. The PC receives the frames from the cameras, detects the foot and robot, estimates the distance and calculates the inverse kinematics. The Arduino controls the servo motors from the robot, so that a foot massage is possible. Both devices are connected with a serial cable. Further, an additional component that is attached to the top of the Arduino, is required to supply the servo motors with power. For communication on the PC side, an additional library *serial* [5] is used for opening a serial port. All the messages have an *UTF-8* format. In addition, a start- and an end-character are defined, which marks the beginning and the ending of a message. This is required to prevent the system from missing important information, for example an angle for the servomotor or the physical contact message.

[5] `https://pythonhosted.org/pyserial/` (visited on 16.12.2022)

4.8.2 Robot Movement via Arduino

The robot moves through the rotations of the servo motors. To control them, an additional library *Adafruit PWMServoDriver* [6] is used. Further, the Arduino receives rotation angles for every motor from the PC. These angles have to be converted into pulses, because the angles cannot be used for rotating the servos. Therefore, the minimum and maximum pulse values are required for each motor. This has to be taken into account because no joint of the robot can rotate *360* degrees due to physical limitations. All the servos are configured and tested, so that the minimum and maximum pulse values are known. With this information it is possible to map an angle of the simulation to a pulse.

Before a massage starts, the robot moves to its starting position with predefined angles. The movement of the robot is regulated with a speed value, that is added to the current pulse until the newly mapped pulse value is reached. In addition, the motors are controlled from the bottom to the top. This means that the order is base, shoulder, elbow and wrist. Each joint rotates alternately to the new angle, so that simultaneous movements are performed since path planning is not implemented. Also, when all motors have reached the new pulses, the Arduino sends a message if a button has been pressed and physical contact has occurred. It is often the case that no contact was made, then a new massage point is created, which refers to a forward movement of the robot, and the new angles are sent to the Arduino.

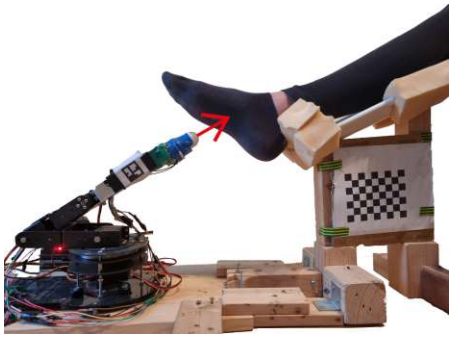

Figure 4.11: Massage Technique: Frictioning.

When a button is pressed, the robot starts to perform *frictioning*. For this reason, the difference between the last two angles is added again to the new angle, so that the robot moves forward again and applies additional pressure. After two seconds, the robot moves back to its original position before frictioning. This massage technique is shown in Figure 4.11. In addition, more pressure can be applied via an audio command, which is explained in Section 4.9.

Another massage technique, which is optional, is *stroking*. It is shown in Figure 4.12. This movement is performed by detecting the position of the robot and moving it to the

[6] `https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library` (visited on 16.12.2022)

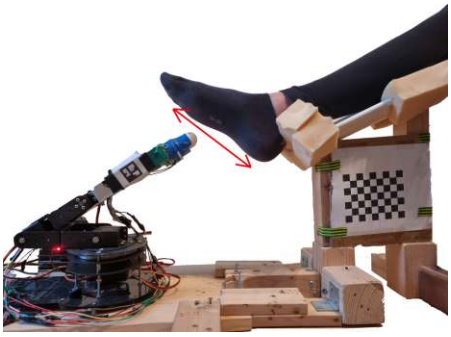

Figure 4.12: Massage Technique: Stroking.

opposite top or bottom point and then to the center of the foot so that the end-effector strokes along the foot sole. Due to the lack of path planning and the poor performance of the servo motors, the robot may get stuck in one place or miss contact with the foot during the stroking massage. The speed of the movement can also be controlled with audio recognition, which is explained in the next section.

4.9 Audio Recognition

When the user is sitting in front of the robot, it is difficult to start, stop or modify the movement of the robot during the massage by pressing a button on the keyboard. Hence, another task of this thesis is to train and implement a model that recognizes specified audio commands. This model is a CNN with an *M5*-architecture, that uses four convolutional layers and a fully connected one. Each convolutional layer is followed by a batch normalization layer, which reduces vanishing gradients, and a max pooling layer to limit computational costs. The CNN architecture is shown in Figure 4.13.

Figure 4.13: Architecture of the CNN model for the audio recognition.

This model architecture is set up after Dai et al. [DDQ+17]. Also, the first layer has a kernel size of *80*. Therefore, audio samples at *8*kHz have a receptive field of *10*ms, which is similar to a receptive field from speech processing which uses about *20*ms. If samples of *4*kHz are used, the receptive field is about *20*ms. As an optimizer *Adam*, which is also known as *Adaptive with Momentum*, is used with a learning rate of *0.001*. The optimizer computes adaptive learning rates for each parameter.

This CNN model predicts labels for raw waveforms. The set from Warden [War18] is used as training data. Additional samples, recorded with a microphone from non-native speakers, are added. Three CNNs were trained to reduce the number of *False Positive*. The commands are explained in detail in Section 5.2. The first CNN only recognizes the start command *GO* and some other words, which are ignored. The other

two CNNs recognize commands to modify the massage. These two networks are always used to process the same audio sample during the massage to reduce the false predictions. Therefore, both predictions have to be the same output. The dataset was split in a ratio of *80:20* and *85:15* for training and test set. The second model has an accuracy of *95%* and the third one an accuracy of *94%*. In addition, the first model has an accuracy of *96%*. For training *PyTorch*[7] was used.

In order to filter the audio samples that do not contain commands, and to reduce the amount of data that is further processed with the audio CNN model, a peak detection has been implemented. For this purpose, only audio samples above an average threshold, which is calculated before the massage start, are processed. To not block any process, threading is used. Also, small audio samples are stitched together, so that no command is overheard. When a peak above the threshold is detected, *0.2* seconds before the peak are added to the sample. For more appropriate audio samples, a microphone is used during the massage, to record the speech commands.

[7] `https://pytorch.org/` (visited on 19.12.2022)

CHAPTER 5

Evaluation

In this chapter, the steps for evaluating the prototype are presented. The first two sections cover the technical setup of the foot, robot and audio recognition. The third one deals with the case study of a defined test group. Finally, the results of the evaluation from the technical part and the case study are presented and discussed.

5.1 Foot & Robot Recognition

One of the main goals was to recognize and detect the mask of each individual foot. Therefore, a pre-trained model on Common Objects in Common (COCO) train2017 with ResNet50 was fine-tuned [LMB+14]. To evaluate the performance of the mask region-based convolutional neural network (Mask R-CNN), which is used to detect the foot and robot, COCO Evaluator was used [HGDG17]. Furthermore, all results of each epoch were documented during the training. The dataset contains images of three object categories consisting of more than *2400* labeled instances in *1440* images. The categories are divided into *foot*, *robot* and *background*.

5.2 Audio Commands

Another main goal was the interaction with the robot. This was realized through audio commands and three convolutional neural networks (CNN), representing a specific M5 architecture [DDQ+17]. Three networks were trained to reduce the number of *False Positive* commands. The first only recognized the command *GO*, which was used to start the massage. Furthermore, to reduce the *False Positive* some other words like *bed*, *cat* etc. were in the dataset. The other two were trained with about *20000* audio samples [War18]. Both of them had to recognize the same word when reaching a certain recognition-threshold, to process the command. The following commands were recognized:

- *backward* - if frictioning is performed, the robot will press less
- *forward* - if frictioning is performed the robot will press more
- *one* - slowest speed
- *two* - middle speed
- *three* - fastest speed
- *stop* - the robot stops and moves back to its default position

5.3 Case Study

The case study was designed to evaluate the practicality of the system for a simple foot massage. The goal is to find out if the system is able to reduce the stress level of a person, as well as to improve their well-being and relax them. Further, if the robot can perform a state-of-the-art foot massage and if it is possible to adjust the massage in a convenient way based on the audio feedback from the user.

5.3.1 Test Group

For the evaluation of the case study, thirteen participants were included. To cover a wide range of possible experiences, no restrictions to a specific gender nor age requirement were set. However, the test users had to speak in English, with the reason being that the audio model only recognized commands in English. In order to obtain more significant results, it was also important that the feet of the participants were as diverse as possible.

5.3.2 Setup

For the setup, the participants had to sit on a chair in front of the robot for about twenty minutes. To obtain more significant results, both feet were tested and each foot was massaged for ten minutes. All of the participants had to answer a few questions before and after the massage test, to compare their previous comfort and the stress level to their current one. Further, each person was given a headset with a microphone to interact with the robot. In addition, the participants were informed about and introduced to the audio commands. Afterwards, it was explained how the robot moves and what massage techniques were going to be performed. In the first half, the system only performed frictioning, and in the second half stroking was added [WK] . To add stroking, a variable in the program was manually changed and the system was restarted. In order to process the frames as fast as possible, an NVIDIA graphics card and CUDA [1] were used.

[1] `https://developer.nvidia.com/cuda-downloads` (visited on 17.10.2022)

5.3.3 Feedback

As mentioned in Section 5.3.2, each participant had to answer a few questions. A goal of the feedback was to collect data about their experience with foot massages, details about their foot shape and their relaxation level. These questions were part qualitative and part quantitative. Another goal of the case study was to evaluate the performance of the robot. The questionnaire included *20* questions with the following answer types:

- *Scale X-Y*: One value on a scale between X and Y can be selected.
- *Single Choice*: Only one option can be selected. An example is yes/no.
- *Free Text*: Answer in sentences.

Questions in Table 5.1 were asked before the test started. After the robot finished massaging both feet, the participants provided feedback using the following questions in Table 5.2.

Age	Scale 1 - 99
Gender	Female / Male
Body height	Scale 140 - 200 Centimeter
Weight	Scale 40 - 100 Kilogram
Shoe size	Scale 30 - 50 EU
Foot Shape	Normal/ Flat/ Hollow
Have you been to a professional foot massage before?	Yes/No
How long did the professional foot massage last approximately?	Scale 1 - 59 Minutes
What is your current state of relaxation before the massage?	Scale 1 - 10 1 = Stressed 5 = Normal 10 = Relaxed

Table 5.1: body information and asked questions before the massage.

How well did the robot respond to your commands?	Scale 1 - 10 1 = Not at all 10 = Very good
Would you have preferred more voice commands? - if yes, which commands?	Yes/No Free Text
How does your RIGHT foot feel after the massage?	Scale 1-10 1 = Worse 5 = Same 10 = Better
How does your LEFT foot feel after the massage?	Scale 1-10 1 = Worse 5 = Same 10 = Better
Were there moments where the robot applied too much pressure or caused pain?	Yes/No
Compared to a professional foot massage, is there a big difference - if yes, which one? (only asked if the participant had been to a professional foot massage)	Yes/No Free Text
How pleasant was the massage?	Scale 1-10 1 = Not pleasant 10 = Very pleasant
What is your current state of relaxation?	Scale 1-10 1 = Stressed 5 = Normal 10 = Relaxed
Was frictioning and stroking better than only frictioning - if yes why?	Yes/No Free Text
Would you change the movement pattern of the robot - if yes, what would you change?	Yes/No Free Text
Do you have any suggestions for improvement?	Free Text

Table 5.2: Feedback after the massage.

5.4 Results

In this section, the evaluation of the convolutional neural network is presented and the results of the case study and its evaluation are shown. It is important to mention that only thirteen participants were included in the case study. The following presented results are not statistically representative and the conclusion should be seen as hypothesis.

5.4.1 Mask Region-Based Convolutional Neural Network

As mentioned in the previous section 5.1, the COCO evaluator was used to evaluate the fine-tuned Mask R-CNN model. The model was trained for *60* epochs with *1140* images for training and *300* images for testing and evaluation. In Figure 5.1 the end results are presented. They are measured with two Intersection over Union (IoU) metrics which are explained in Section 2.1.9. The mean average precision for the bounding boxes is *0.85*. Moreover, with the IoU metric mask segmentation, the model reaches a value of *0.827*.

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.849
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.994
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.959
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.808
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.855
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.845
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.882
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.886
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.886
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.821
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.875
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.883
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.824
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.995
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.930
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.772
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.815
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.839
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.860
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.863
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.863
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.865
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.872
```

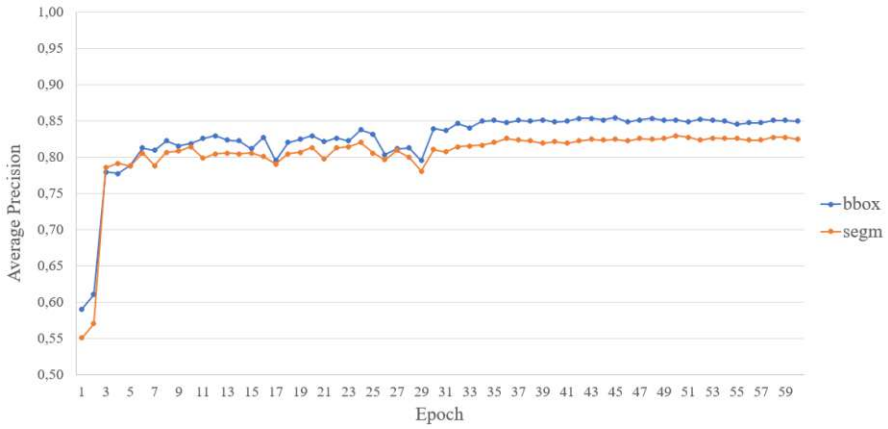

Figure 5.1: Results of the M-RCNN model after *60* epochs.

The average precision could be increased if two Mask R-CNNs were trained. The first model would only recognize the robot and the second one only the foot. Another problem could be that the dataset contained images of the robot without the head modification. Therefore the label was different and due to that it could be possible that the model was

overfitted. In addition, the shape of the feet were different in some images as well, due to the robot covering some parts of it. Those pictures were important for recognizing the foot because during the massage the sole is never fully visible for the back view camera. An example of the foot and robot detection is shown in Figure 5.2.

Figure 5.2: The ground truth is shown on the left side and the prediction on the right side.

5.4.2 Test Group

The test group consisted of thirteen people with different foot shapes. As shown in Table 5.3 the gender was fairly even distributed. The shoe size ranged from *37* to *44* EU, with all females having a shoe size between *36* and *40* EU and males being between *40* and *44* EU. Half of the participants had the foot shape *normal* and the other half had *hollow* or *flat*.

ID	Age	Gender	Body Size	Weight	Foot Shape	Shoe Size
N01	25	female	171	51	normal	39
N02	32	female	171	65	normal	37
N03	26	male	178	89	normal	43
H04	51	female	168	48	hollow	38
F05	27	male	172	77	flat	43
N06	60	male	180	74	normal	44
H07	77	male	178	71	hollow	43
N08	77	female	165	62	normal	38
H09	57	female	165	60	hollow	40
H10	59	male	183	75	hollow	42
N11	26	female	172	46	normal	38
H12	25	female	167	60	hollow	39
N13	29	male	182	85	normal	44

Table 5.3: Information about the participants.

5.4.3 Massage & Relaxation Effect

The main goals of the case study were to evaluate if a robot can perform a state-of-the-art foot massage. Further, if it is possible to adjust the massage in a convenient way based on the audio feedback from the user. Finally, if the robot can reduce the stress level and improve the well-being of the user within a twenty minute foot massage. The case study was designed to answer these questions. First of all, to answer the question if a robot can perform a state-of-the art foot massage, it is important as to how many participants have been to a professional foot massage before.

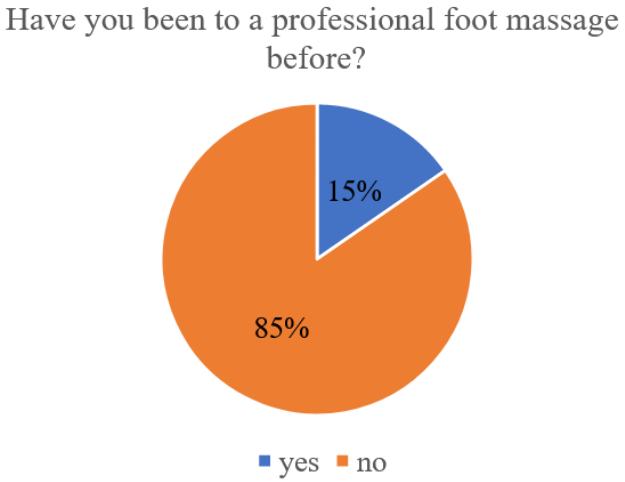

Figure 5.3: Overview of the participant's foot massage experience.

In Figure 5.3 it is shown that only *15%* of the participants had experience with a professional foot massage. The massage lasted about twenty minutes for both feet. The other *85%* have never been to a professional foot masseur, but all indicated that they have received a massage from their partner or a friend. After the massage, those *15%* said that there was a significant difference to a professional foot massage. The main difference was that the head of the robot is a ball, and not a robotic hand that can grasp and knead a foot. Since the robot has no fingers, it can be concluded that the robot cannot perform a fully human-like massage for a foot because the instep was neglected. Nevertheless, it is important to point out, that a state-of-the-art massage can be performed only on a sole as well. Therefore, each participant was asked how the performance of frictioning and stroking was. All thirteen users mentioned that the performance of the frictioning massage was good. Furthermore, all participants felt that the option with the stroking movement was better than without, because it felt more like a normal foot massage, even if it did not always work as well.

Figure 5.4 shows that every user had a better feeling on both feet afterwards. Nevertheless, the robot had a better performance on right foot. The reason could be attributed to the footrest, because it was turned a little to the right, so that the distance between

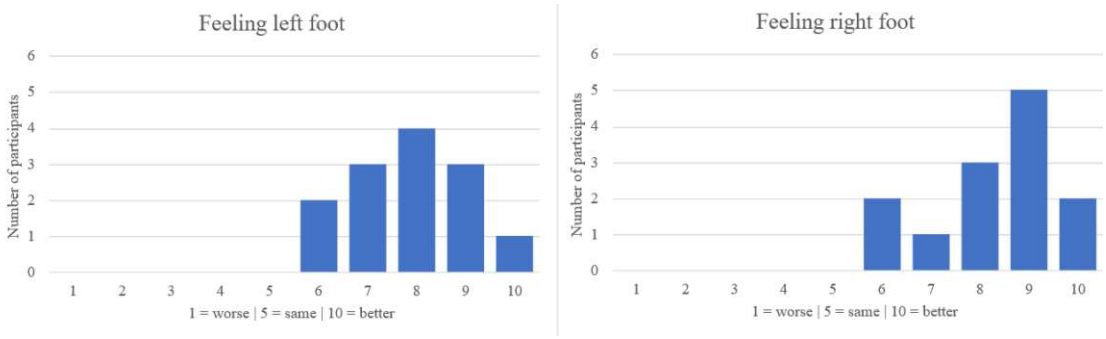

Figure 5.4: Survey results for left and right foot feeling.

the longitudinal arch of the foot and the head of the robot is minimized. In case of a left foot, the rotation was a disadvantage because the head of the robot was further away. Moreover, the side view camera was mounted on the right. The head of the robot was always visible if a right foot was massaged, however when massaging a left foot, it partially covered the robot on camera as soon as it tried to touch the sole. If the head is not detected and localized, the function for the correction of its position fails.

Another question was if it is possible to control the robot via voice commands to adjust the massage in a convenient way. Unfortunately, some participants did not change much in the options. The start command *GO* and the stop command *STOP* were used by every participant. In addition, the robot did not always respond well, because sometimes the command was predicted incorrectly or it was not recognized. The reason could be that the background noise was often very loud. Further, the transformed audio sample that should be analyzed, occasionally did not contain the whole word of the spoken command, due to the audio samples being merged, when a peak was louder than the defined threshold. To point this out the second model has an accuracy of *95*% and the third model *94*%. The dataset was split into a training and a test set with a ratio of *80*:*20* for the second model and for the third model with a ratio of *85*:*15*.

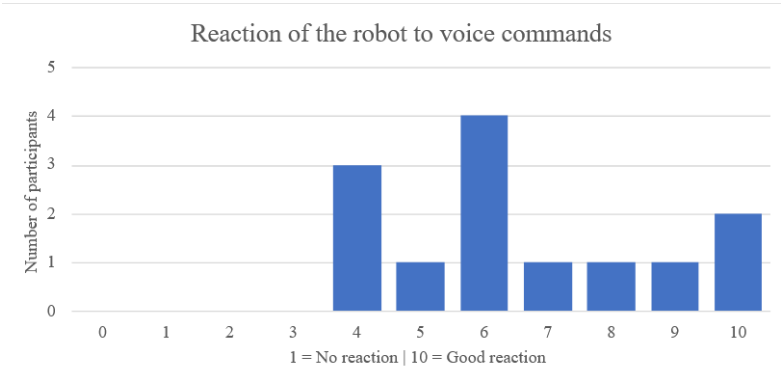

Figure 5.5: Survey results for the robot reaction to the voice commands.

The participants were asked how well the robot responded to their voice commands. In Figure 5.5 the results from the reaction of the robot to the given voice commands are shown. More than *76%* answered that the robot reacted well to their commands and they felt changes within the massage. The other *23%* said that the robot did not respond well to their commands. The participants were also asked if they would prefer more voice commands. All of them were satisfied with the existing audio options. Since most participants did not use the audio commands often, except the commands *STOP* and *GO*, it possibly implies that the basic settings were satisfying enough. In addition, none of the users would change the movement of the robot or the order and position of the massage points. Further, it seems to be the case that none of the participants felt pain, considering they had the possibility to lower the pressure of the massage and did not do so. Since the participants changed the options only a few times during the massage, it can hardly be said if it is possible to adjust the massage in a convenient way.

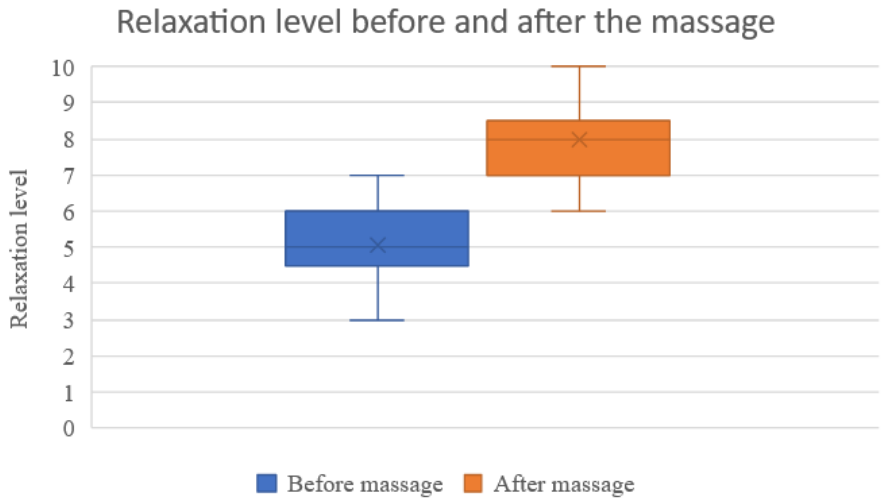

Figure 5.6: Survey results before and after the massage.

The last main goal was to find out if a foot massage by a robot can reduce the stress level and improve the well-being of a person. To answer this question, all of the participants were asked before and after the massage what their current state of relaxation was. In Figure 5.6 the survey results are shown. The scale ranged from *1* to *10*, whereby *1* represents *stressed*, *5* stands for *normal* and *10* is for *relaxed*. Before the massage the average relaxation level was *5.07* and the deviation *0.977*. Nearly every person was in a normal or good mood. Only a few user were below *5*. After the massage, the average relaxation level increased from *5* to *8* with a deviation of *1.109*. This means that every person was in a better mood as before. An important observation is that every participant, who had a relaxation level below five, had a significantly higher level than before the massage. If a person was already in a relaxed mood, the massage increased the well-being by only one or two points. Moreover, *24%* of the participants had extra feedback to improve the massage. Additionally, all of them said that a different structure

of the ball would be nice, such as a rougher surface or a covering in massage oil. To conclude this evaluation part, it can be said that the robot is able to reduce the stress level and to improve the well-being of a participant.

5.5 Discussion

In the previous sections the results of the evaluation were presented. In Section 5.1 and 5.2 the technical main goals were explained. Further, Section 5.3 presented the case study and the physical goals of this thesis. The main focus of the case study referred to the questions if a robot can perform a state-of-the-art foot massage, if it is possible to adjust the massage in a convenient way based on the audio feedback from the user and if it is possible to reduce the stress level and to improve the well-being of the user within a twenty minute foot massage from a robot. In the following, the results of the evaluation are discussed.

The implementation and the training of the mask region-based neural network were discussed in the previous chapter. It is important to recall that the dataset contained images from the robot without the modified head. Further, the robot occluded parts of the foot in some pictures, in order to recognize the sole even if it is not fully visible. As shown in Figure 5.1 the achieved bounding box accuracy was at *85*% and mask segmentation was at *82*%. It is worth pointing out that the model recognized the foot it was presented, even if the user wore socks with different colours and patterns. Furthermore, if the robot occluded parts of the sole, the detection still worked for the visible area. This was an important task, because when a participant rotated the foot during the massage, the massage points had to be updated in such way that the robot would not miss the sole. Sometimes the robot missed it, implying that this could be an indication that there was a problem with the recognition.

If the socks and pants of the participant were the same colour, the sole and the legs could not always be distinguished. Therefore, the massage points were rotated and translated in a wrong way. Further, the robot was not always recognized well, because the model could be overfitted with images of the robot without its modified head. Another problem could be a situation, where the prototype is in a room with bad lighting conditions, due to the fact that the cameras used are not the most expensive ones. All in all it can be concluded that this model is robust enough.

As mentioned in the previous section, only few participants had been to a professional foot massage, but all thirteen had at least a regular massage. To point this out, each person knew the feeling of being massaged. Since the head of the robot was a ball and not a robotic hand, it was not possible to perform kneading. Therefore, a full foot massage is not possible, despite that, Figure 5.4 has shown that the feet of each person felt better. To answer the question if a robot is capable of performing a state-of-the-art foot massage, each participant was asked about the performance of frictioning and stroking. These techniques were limited to the sole of the foot. Everyone said that frictioning was good and the pressure was not too much. To perform a more professional and varied massage,

stroking was manually added at the second half. All of the participants mentioned that the combination of both techniques felt better than with frictioning only. However, the users indicated that the stroking movement did not always work well, because the exact shape of the sole is not taken into account while performing this technique. To optimize the stroking movement, a convex surface detection could be implemented. In summary to answer the first question, the robot is capable of performing a state-of-the-art foot massage.

It was mentioned in the previous section that the participants did not use the voice command function often. Since the default options of the massage were pleasant, none of them had a reason to change something via the audio commands. Nevertheless, it was found that the variety of commands was enough to adjust the massage points. Another observation was that some voice commands were not always recognized correctly. Therefore, the accuracy in a real world scenario is lower than 95% and 94%. A reason could be the difference between the quality from the voice commands and the samples of the training set. In the previous sections it was already mentioned, that the audio models are trained with audio samples, that are a raw waveform. This could be another reason for the audio recognition problem. A solution could be to train the models with audio samples, where mel-frequency cepstral coefficients are extracted. As mentioned in the previous section, it is difficult to say whether it is possible to adjust the massage in a convenient way, with the reason being that the participants did not interact with the robot often.

To answer the last question whether it is possible to reduce the stress level and improve the well-being of a participant, each user reported to be more relaxed afterwards. Even those, who were already in a pleasant mood, saw an improvement of their well-being. In conclusion, it is possible to reduce the stress level and improve the well-being of a participant, after a foot massage by a robot.

CHAPTER 6

Summary

6.1 Conclusion

In this work, a hardware and software prototype was developed. Modifying an existing robotic arm, building a footrest and connecting all components to the platform are parts of the process in creating the hardware prototype. The software prototype includes foot and robot recognition, distance estimation, audio recognition, massage point generation and inverse kinematics calculation within a simulation. In particular, all this steps were necessary to investigate whether it is possible that a robotic arm can perform a state-of-the-art foot massage.

To achieve this goal, the hardware prototype was developed first. For this purpose, the robotic arm was modified so that it has *4*-DOF and an end-effector consisting of a deodorant roller (with built-in buttons to verify physical contact) instead of a claw to ensure a comfortable contact with the sole of a foot. Building the footrest and connecting each component to a platform was the next step. After that, the software prototype was developed. The first part was the recognition of the foot and the robot. Accordingly, many images with different feet were collected and labeled. Since a robot was modified for this thesis, only images of that robot were taken. Then, a Mask R-CNN was trained and implemented into the program. A distance estimation function for the side and back camera was the next step. Further, a mask of a sole template was saved and massage points were marked on it. The mask and points are always transformed to the mask of the foot to be massaged. To move the robot, the massage points are mapped into a *2*D and *3*D simulation. Then the inverse kinematics are calculated and simulated. Finally, the angles of individual joints are sent to the Arduino, which controls the robot, and the foot massage begins. The Arduino also checks for physical contact and performs some additional movements for a state-of-the-art foot massage. To manipulate the movements of the robot during the massage, audio recognition was implemented. Due to

this implementation, it was possible to start and stop the robot and change the pressure or speed of the massage via voice commands.

To answer the research questions of this thesis presented in Chapter 1, a case study was designed. For this reason, thirteen people were tested for approximately twenty minutes. Before and after the massage, the participants had to fill out a questionnaire about their experience. Later, these questions were analyzed and evaluated. The result was that every person had a better feeling and was more relaxed after the foot massage. In addition, the combination of stroking and frictioning felt better than a massage that only involved frictioning, although stroking did not always work well.

Another observation was that the CNN used for audio recognition had some issues in predicting audio commands from raw wave samples. The reason could be the difference between the quality from the voice commands and the samples of the training set. The participants did not interact with the robot often, as the default configurations were comfortable enough. In conclusion, it is possible to decrease stress and perform a state-of-the-art foot massage with a robot.

6.2 Future Work

Since this thesis consists of a hardware and software part, some limitations have been faced. The provided robot had a claw as end-effector, therefore modifications were made so that the end-effector is a deodorant roller with a comfortable surface for a massage. Further, the robot misses an additional joint before the end-effector, keeping it from reaching the outer edges of a sole during a massage and from performing more complex movements. Because of this, and due to the limitation of time for this thesis, the robot was kept simple. Moreover, the processing, detection and recognition tasks are very expensive to compute. A newer GPU that supports CUDA is required to process these tasks as quickly as possible to move the robot smoothly. For this reason, and in fact that a Raspberry has no GPU which supports CUDA, the prototype was not as easy to transport. The robot can massage the left and right foot. Because the side camera is on the left, the longitudinal vault of the left foot is not visible. As a result, the massages on the left are less accurate.

In a future work, the robot could be extended with an additional joint and the inverse kinematics could be modified so that the movement can be more complex. An advantage would be that the robot would not slip off the sole when the pressure is increased and when the end-effector is rotated in the correction direction. In addition, with refined distance estimation and a convex surface detection, the stroking massage technique would be more accurate and comfortable. Another improvement can be made in the audio command recognition. For this purpose, a model with mel-frequency cepstral coefficients feature could be trained. Further, the audio recognition models could be extended to recognize more commands, so that more modifications to the movements of the massage are possible. An example would be the command *left*, limiting the movement of the

robot to the left side of the sole. To conclude, all these possible implementations and improvements could lead to a more pleasant massage by the robotic arm.

List of Figures

List of Tables

Bibliography

[AAE10] Ahmed Z Alassar, Iyad M Abuhadrous, and Hatem A Elaydi. Modeling and control of 5 dof robot arm using supervisory control. In *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, volume 3, pages 351–355. IEEE, 2010.

[AMAZ17] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[Ben12] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[CHM20] Keene Chin, Tess Hellebrekers, and Carmel Majidi. Machine learning for soft robotic sensing and control. *Advanced Intelligent Systems*, 2(6):1900171, 2020.

[CHP21] Konstantinos Chatzilygeroudis, Ioannis Hatzilygeroudis, and Isidoros Perikos. Machine learning basics. In *Intelligent Computing for Interactive System Design: Statistics, Digital Signal Processing, and Machine Learning in Practice*, pages 143–193. 2021.

[Cor07] Peter I Corke. A simple and systematic approach to assigning denavit–hartenberg parameters. *IEEE transactions on robotics*, 23(3):590–594, 2007.

[Cos21] Luciano da F Costa. Further generalizations of the jaccard index. *arXiv preprint arXiv:2110.09619*, 2021.

[CV14] Visesh Chari and Ashok Veeraraghavan. *Lens Distortion, Radial Distortion*, pages 443–445. Springer US, Boston, MA, 2014.

[Cyr75] James Cyriax. Textbook of orthopaedic medicine. In *Textbook of orthopaedic medicine*, pages xii–756. 1975.

[DDQ+17] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. Very deep convolutional neural networks for raw waveforms. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 421–425, 2017.

[DG01] Lijun Ding and Ardeshir Goshtasby. On the canny edge detector. *Pattern recognition*, 34(3):721–725, 2001.

[DPWJ22] Yanmei Dong, Mingtao Pei, Yuwei Wu, and Yunde Jia. Stitching images from a conventional camera and a fisheye camera based on nonrigid warping. *Multimedia Tools and Applications*, 81(13):18417–18435, 2022.

[DQZ18] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841, 2018.

[DSSF+17] Ivan Nunes Da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, and Silas Franco dos Reis Alves. Artificial neural networks. *Cham: Springer International Publishing*, 39, 2017.

[ENM15] Issam El Naqa and Martin J. Murphy. *What Is Machine Learning?*, pages 3–11. 2015.

[Fia05] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005.

[Fia09] Mark Fiala. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern analysis and machine intelligence*, 32(7):1317–1324, 2009.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[GJMSMCMJ14] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[GS93] François Guély and Patrick Siarry. Gradient descent method for optimizing various fuzzy rule bases. In *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*, pages 1241–1246. IEEE, 1993.

[GWB17] Mehdi Gheisari, Guojun Wang, and Md Zakirul Alam Bhuiyan. A survey on deep learning in big data. In *2017 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC)*, volume 2, pages 173–180, 2017.

[HDO+98] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[HDY+12] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. 29(6):82–97, 2012.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[HJ99] Cox C. Hayes J. Immediate effects of a five-minute foot massage on patients in critical care. *Intensive and Critical Care Nursing*, 15(2):77–82, 1999.

[HS97] Janne Heikkila and Olli Silvén. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1106–1112. IEEE, 1997.

[HS15] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360, 2015.

[Hui18a] Jonathan Hui. Image segmentation with Mask R-CNN . `https://jonathan-hui.medium.com/image-segmentation-with-mask-r-cnn-ebe6d793272`, 2018. [Online; accessed 09-November-2022].

[Hui18b] Jonathan Hui. Understanding feature pyramid networks for object detection (fpn). `https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c`, 2018. [Online; accessed 07-November-2022].

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE*

conference on computer vision and pattern recognition, pages 770–778, 2016.

[iC] COCO Common Objects in Context. Detection evaluation. `https://cocodataset.org/#detection-eval`. [Online; accessed 05-November-2022].

[Jap06] Nathalie Japkowicz. Why question machine learning evaluation methods. In *AAAI workshop on evaluation methods for machine learning*, pages 6–11, 2006.

[JMM96] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[Joa83] Gloria Joachim. How to give a great foot massage. *Geriatric nursing (New York, NY)*, 4(1):28–29, 1983.

[KB99] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94. IEEE, 1999.

[KB06] Serdar Kucuk and Zafer Bingul. *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK, 2006.

[KBK11] Andrej Krenker, Janez Bešter, and Andrej Kos. Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pages 1–18, 2011.

[Kee03] Lynn Keegan. Therapies to reduce stress and anxiety. *Critical Care Nursing Clinics*, 15(3):321–327, 2003.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[KYW18] Ho Chuen Kam, Ying Kin Yu, and Kin Hong Wong. An improvement on aruco marker for pose tracking using kalman filter. In *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 65–69. IEEE, 2018.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[LES20] Igor Lebedev, Aleksei Erashov, and Aleksandra Shabanova. Accurate autonomous uav landing using vision-based detection of aruco-marker. In *International Conference on Interactive Collaborative Robotics*, pages 179–188. Springer, 2020.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755, 2014.

[LPW+17] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.

[LRM20] Valliappa Lakshmanan, Sara Robinson, and Michael Munn. *Machine learning design patterns*. O'Reilly Media, 2020.

[Mat] Mathworks. Image undistortion. `https://de.mathworks.com/help/visionhdl/ug/image-undistort.html`. [Online; accessed 05-November-2022].

[MD18] MATLAB and Simulink MathWorks Deutschland. What is camera calibration. `https://de.mathworks.com/help/vision/ug/camera-calibration.html`, 2018. [Online; accessed 11-November-2022].

[Mit97] Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[ML18] Dechrit Maneetham and Sivhour Leng. Pc - based 5dof industrial robotic arm with object color sorting by image processing. 2018.

[MM04] Michael Meredith and Steve Maddock. Real-time inverse kinematics: The return of the jacobian. 2004.

[Qin20] Tao Qin. Machine learning basics. In *Dual Learning*, pages 11–23. Springer, 2020.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[RNZ18] Muhammad Imran Razzak, Saeeda Naz, and Ahmad Zaib. Deep learning for medical image processing: Overview, challenges and the future. *Classification in BioApps*, pages 323–350, 2018.

[RTG+19] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SMB10] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[Soo08] Tan Jin Soon. Qr code. *synthesis journal*, 2008:59–78, 2008.

[SS18] Pramila P. Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–6, 2018.

[SSA17] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[SSVO09] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Kinematics*. Springer, 2009.

[Sta22] Amazon Staff. 10 years of Amazon robotics: how robots help sort packages, move product, and improve safety . `https://www.aboutamazon.com/news/operations/10-years-of-amazon-robotics-how-robots-help-sort-packages-move-product-and-improve-safety`, 2022. [Online; accessed 20-October-2022].

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[UVDSGS13] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[VJMJ21] Prerna Varma, Moira Junge, Hailey Meaklim, and Melinda L. Jackson. Younger people are more vulnerable to stress, anxiety and depression during covid-19 pandemic: A global cross-sectional survey. *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, 109:110236, 2021.

[VZ99] Andrew Vickers and Catherine Zollman. Massage therapies. *BMJ*, 319(7219):1254–1257, 1999.

[War18] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[Wei03] Eric W Weisstein. Rotation matrix. *https://mathworld. wolfram. com/*, 2003.

[WK] Hsiao-Lan Wang and Juanita F. Keck. Foot and hand massage as an intervention for postoperative pain.

[YNDT18] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[You89] Matt Young. The pinhole camera: Imaging without lenses or mirrors. *The Physics Teacher*, 27(9):648–655, 1989.

[YS20] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[ZCLM20] Yiqing Zhang, Jun Chu, Lu Leng, and Jun Miao. Mask-refined r-cnn: a network for refining object details in instance segmentation. *Sensors*, 20(4):1010, 2020.

[Zha10] Yagang Zhang. *Machine Learning*. BoD–Books on Demand, 2010.

[ZYM18] Yufeng Zheng, Clifford Yang, and Alex Merkulov. Breast cancer screening using convolutional neural network and follow-up digital mammography. In *Computational Imaging III*, volume 10669, page 1066905. SPIE, 2018.

[ZZJ19] Haotian Zhang, Lin Zhang, and Yuan Jiang. Overfitting and underfitting analysis for deep learning based end-to-end communication systems. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2019.