Robotic Grasping of Deformable Food Objects

A Human-Inspired Reinforcement Learning Approach

Jelle Luijkx

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Robotic Grasping of Deformable Food Objects

A Human-Inspired Reinforcement Learning Approach

Master of Science Thesis

For the degree of Master of Science in Systems and Control & Mechanical Engineering at Delft University of Technology

Jelle Luijkx

October 19, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Summary

During the process of meat, fruit and vegetable processing and packaging, many stages involve humans handling the food objects. For a number of reasons, it is desirable to look for robotic solutions to either assist the human or in some cases even take over the manipulation tasks. First of all, some of these tasks can be tedious and physically demanding for the workers. Next to reducing the workload, there is another benefit that comes with robotic manipulation of food objects. The risk of contamination is inherent in human handling of food objects and a robotic system can almost completely eliminate this risk. Furthermore, the economic perspective cannot be ignored when looking at robotic solutions for harvesting and processing food objects. Automation of manipulation tasks might also result in lower production costs, higher productivity and lower costs of the food products.

Grasping forms the basis of many manipulation tasks and it is therefore an essential skill for robotic manipulation of deformable food objects. Solving the grasping problem will be an important step towards robotic harvesting, packaging and other tasks involving food objects. Robotic grasping of deformable food objects is a challenging task due to the discontinuous nature of the task, the large variety in deformation properties and the large number of contact points. Inspired by the way humans effortlessly grasp objects that they have never manipulated before, we propose to solve the task by using a combination of tactile and visual sensory information. This study will combine visual and tactile information in our grasp approach as follows. Initially, the desired grasp force is determined based on visual features. The desired grasp force is adapted while holding the object, based on tactile information. It is a reinforcement learning approach, which allows the robot to grasp food objects without the need for explicit programming or complex modelling. The aim of this study is to develop and evaluate a novel approach for grasping deformable food objects with a human-inspired reinforcement learning approach using visual features and pressure information.

The goal of the grasp approach is to grasp deformable food objects with minimal force in order to prevent damage to the object, since food objects are often fragile and have to be in perfect shape to be saleable. The hardware that is involved in this approach consists of an Red Green Blue - Depth (RGB-D) camera and a robotic hand. Furthermore, the grasp approach involves three modules, i.e. the vision, control and learning module. The purpose of the vision module is to detect objects and obtain visual features of the object. For object detection and feature extraction, the depth information of the RGB-D camera is used. The goal of the control module is to track the desired grasp forces that are provided by the learning module and to provide tactile information to the learning module. The learning module is

responsible for setting the desired grasp force and does this with a reinforcement learning approach.

A novel algorithm is proposed for learning to grasp with minimal force: Learning of Initial Force and Tuning (LIFT). The learning approach consists of two parts, i.e. an outer loop that is responsible for choosing the initial desired grasp force based on the visual features and an inner loop that is responsible for adapting the desired grasp force based on the tactile information in combination with the visual features. The reason for this two-step approach, is that in this way a force range can be chosen based on the visual features and that the force can be adapted based on the tactile information, e.g. to increase the force when the object starts slipping.

The grasp approach is evaluated in two ways. The first evaluation is performed in simulation, while the second evaluation is performed on a real-world robotic setup. The benefit of the evaluation in simulation is that system interactions are not expensive, since in simulation the setup cannot be damaged. Also, in simulation the experiments can be fully automated. From the experiments can be concluded that the approach results in successful grasping in both simulation and on the real-world setup. The method results in decent grasp behaviour in tens or hundreds of learning iterations, depending on the size of the state-action space. The success rate improves over the number of episodes in simulation and on the real-world setup. A success rate of 96.3 % is obtained in simulation. On the real-world setup, a success rate of 96.7 % is obtained over the last ten episodes. The results from the experiments indicate that the approach is successful in grasping with minimal force, since the success rate improved over the learning episodes in the experiments, whereas the grasp force did not increase considerably.

Contents

Acknowledgements

Over the last year I have been working on my graduation project. This was a memorable year, not only due to the fact that it is my last year as a student, but also because it was the first year that I wanted to go to the university more often than I was allowed. Despite the fact that the university closed for some time during this year, I was able to continue to work on my thesis. This was possible thanks to my supervisors Robert Babuska and Padmaja Kulkarni. They allowed me to work from home with hardware from the lab. I am grateful for their flexible and trustful attitude, which enabled me to keep on working. I would also like to thank them for their valuable feedback on progress reports and drafts. Next to my supervisors, I would like to thank Taeke de Haan with whom I drank many cups of coffee and had discussions on our graduation projects.

Delft, University of Technology — Jelle Luijkx
October 19, 2020

“ Failures, repeated failures, are finger posts on the road to achievement. One fails forward toward success.”

— *C. S. Lewis*

Chapter 1

Introduction

1-1 Motivation and Problem Description

During the process of meat, fruit and vegetable processing and packaging, many stages involve humans handling the food objects. For a number of reasons, it is desirable to look for robotic solutions to either assist the human or in some cases even take over the manipulation tasks. First of all, some of these tasks can be tedious and physically demanding for the workers. Next to reducing the workload, there is another benefit that comes with robotic manipulation of food objects. The risk of contamination is inherent in human handling of food objects and a robotic system can almost completely eliminate this risk. Furthermore, the economic perspective cannot be ignored when looking at robotic solutions for harvesting and processing food objects. Automation of manipulation tasks might also result in lower production costs, higher productivity and lower costs of the food products. On the other hand, there are also possible downsides to robotising manipulation tasks. Replacing humans for robotic systems might result in workers losing their jobs or changing the task of the worker such that it will become an uninspiring surveillance task. Another possible undesirable side effect is that it could lead to unfair competition for smaller farmers who are unable to make the large investments that are needed for installing robotic equipment as opposed to large scale farmers who do have the resources to automate their processes. All in all, the consequences of robotising food manipulation tasks should also be studied from perspectives other than the technical one, but we believe that studying the possibilities for robotic manipulation in the agricultural industry is certainly worth the effort.

Recently, the motivation for rethinking our current agricultural processes became even clearer. The COVID-19 crisis has revealed vulnerabilities in our current way of securing and processing foods. Due to travelling restrictions that were introduced to manage the virus outbreak, seasonal labourers were unable to reach the farms where they were needed for harvesting crops (Alderman et al., 2020). Also, meat processing plants turned out to be *"corona hotspots"* (Dickerson and Jordan, 2020), due to the large number of labourers working closely to each other at the plants, the housing conditions of the workers and means of transportation from their homes to the processing plants. This is not only a major health risk for the labourers,

but it also allows the virus to spread further with all the consequences that will entail. This means that the current food production chains can be imperilled in the case of a virus outbreak and even play a significant role in the spreading of the virus. Robotic manipulation of food objects could help to prevent these problems in the case of a new wave of the COVID-19 virus or a similar virus outbreak, since it might reduce the need for seasonal workers and lower the number of workers that are needed at food processing sites.

So far, terms like *robotic manipulation of food objects* are used, but this is a very broad notion. The author does not have the ambition to solve all manipulation tasks that are required to get fruits, vegetables and meat from the farm to the store. Instead, this work focuses on force controlled grasping of deformable food objects using tactile feedback. Grasping forms the basis of many manipulation tasks and it is therefore an essential skill for robotic manipulation of deformable food objects. Solving the grasping problem will be an important step towards robotic harvesting, packaging and other tasks involving food objects. There are two distinct types of constraining grasps studied in the literature, i.e. force closure and form closure grasps. A force closure grasp is defined as: "*a grasp on an object is a force closure grasp if and only if we can exert, through the set of contacts, arbitrary force and moment on this object.*" (Nguyen, 1988) On the other hand, "*a grasp is defined as form closed if and only if it is force closed with frictionless contacts.*" (Bicchi and Kumar, 2000) From these definitions becomes clear that form closure is a stronger condition than force closure. In our grasp approach, the robot aims to grasp objects through force closure. This does not mean that we believe that force closure grasping is a better strategy to grasp deformable food objects than form closure grasping. Many situations can be imagined in which form closure would be preferred over force closure. For example, when picking hanging fruits, a form closure grasp would make more sense than force closing the object, since in such case the object can be enveloped by the end-effector. On the other hand, force closing the object is a superior strategy when grasping food objects from a surface, for example when picking objects from a conveyor belt. For this task, it will be hard to get the tips of the end-effector underneath the object, which is required to obtain a form closure grasp. Therefore both forms of grasping will be useful in the production chain of agricultural goods, but in this work the problem of force closure grasping of deformable food objects is studied.

Robotic grasping of deformable food objects is a challenging task due to the discontinuous nature of the task, the large variety in deformation properties and the large number of contact points. Inspired by the way humans effortlessly grasp objects that they have never manipulated before, we propose to solve the task by using a combination of tactile and visual sensory information. Humans profoundly rely on their tactile senses when grasping objects. This becomes clear when the tactile senses are impaired and manipulation tasks such as grasping become difficult (Johansson and Flanagan, 2009). Nonetheless, the tactile modality is not the only form of sensory information that is used by humans when grasping objects. Humans also make predictions of mechanical events based on visual cues (Land and Furneaux, 1997; Flanagan and Johansson, 2003) and auditory cues (Furukawa et al., 2000; Nelken et al., 2005). The human brain integrates the input from multiple modalities to improve the accuracy of the predictions. Vision and touch provide complementary information about the problem. Whereas visual input can be used to acquire information about the object's geometry, configuration and possible damages; touch can be exploited to learn surface properties, slipping effects and deformations. Next to the integration of these sensory input signals, the dexterity of human grasping can also be attributed to the learning capacities of the human brain. The

efficiency of human grasping can be attributed to accurate predictions of the required grasp force that are based on experience and consequentially learned size-weight relations of objects (Johansson and Flanagan, 2009).

This study will combine visual and tactile information in our grasp approach as follows. Initially, the desired grasp force is determined based on visual features. The desired grasp force is adapted while holding the object, based on tactile information. It is a reinforcement learning approach, which allows the robot to grasp food objects without the need for explicit programming or complex modelling. Furthermore, it allows the system to improve itself and to grasp novel objects without further tuning. While a classical approach does not learn from its mistakes and possibly makes the same error again and again, the method we propose is able to learn from failures and regrasp objects differently after a failed grasp. The aim of this study is to develop and evaluate a novel approach for grasping deformable food objects with a human-inspired reinforcement learning approach using visual features and pressure information.

1-2 Related Work

Due to the hardness of the problem, relatively little research has been conducted on 3D deformable object grasping (Sanchez et al., 2018; Nadon et al., 2018). Currently, some approaches rely on deformation models of the objects, which require a 3D Finite Element Method (FEM) model of the object and measurements of the physical parameters (Lin et al., 2015; Zaidi et al., 2017; Jia et al., 2014). Before grasping, simulations are run to predict grasp forces. Simulating the FEM models is computationally expensive and therefore time-consuming. What makes these approaches even more impractical, is that every object that is to be grasped should be 3D scanned in advance and that physical parameters such as the Young's modulus and Poisson's ratio must be known or have to be measured. Another type of solution that is proposed in the literature, is the caging approach based on geometry models of the object (Kim et al., 2019). Although such a method provides good stability guarantees of the grasp, it is only suitable for simple geometries and requires that the geometry is known on beforehand. Model-free approaches have also been proposed in the literature (Delgado et al., 2017a,b), but these approaches involve heuristically defined control parameters. This makes them unreliable when grasping novel objects.

The idea of combining visual and tactile input in deformable object manipulation is not new (Cretu et al., 2011). Petit et al. (2017) propose a FEM based force estimation technique that consists of two steps. Firstly, a force sensor is used to estimate the Poisson's ratio and Young's modulus. Secondly, 3D forces are estimated based on deformations that are captured with an Red Green Blue - Depth (RGB-D) camera. The limitation of this approach is that it cannot be employed real-time, since the FEM model is computationally demanding. Furthermore, it is not time efficient since it consists of two steps and therefore cannot be used to grasp objects immediately. Arriola-Rios and Wyatt (2017) propose a generative approach to learn the properties of a mass-spring model. Their method includes a force predictor and shape predictor that estimate forces and deformations. Also, it classifies the material of the object that is manipulated and predicts the shape. However, such an approach is not suitable for a grasping problem, because of occlusion of the camera system during grasping which makes it impossible to accurately track deformations. Gil et al. (2016) developed a method

for grasping planar deformable objects with a multi-fingered robot hand. They propose a hybrid approach with a force and vision-based switching controller. This method has several limitations. First, the method is developed for planar objects only and cannot be used for solid objects. Second, it is unclear from the evaluation that is presented how successful the method is in grasping novel objects and how much time it takes to grasp an object using this method. Caccamo et al. (2016) introduce an active perception method for estimating heterogeneous deformation properties of objects. Through the use of position-based dynamics simulations, local deformation parameters are estimated and a global deformability map is obtained by employing Gaussian process regression. However, this approach cannot be used for grasping directly, since the deformations of the surface are obtained by an RGB-D camera system. This is impossible during grasping, because of occlusion of the camera system. Cretu et al. (2011) present a method for extracting and predicting shape deformations. By combining RGB-D video input and force sensor measurements, a deformation model is learned by employing neural networks that can be used in a model-based control strategy. A downside of this approach is that it requires that a deformation model is learned for every object that is handled. In the case of food processing, this is not very efficient since here it is desirable that the system learns a more generalised skill of grasping deformable food objects.

In the field of deep (reinforcement) learning, multimodal representation learning is a research topic that currently attracts a lot of attention. Calandra et al. (2017, 2018) showed that the incorporation of touch sensory input improved the performance of their grasping system significantly as compared to a system that only makes use of visual and proprioceptive sensory information. Lee et al. (2019) propose a deep reinforcement learning method for contact-rich tasks. The contact-rich task on which the method is evaluated is a peg insertion task. Their control scheme includes a convolutional neural network structure that learns a multimodal representation of tactile, proprioception and vision data through self-supervision. This multimodal representation is used to train an impedance controller by employing reinforcement learning. These works show interesting results by combining different kinds of input signals. However, these self-supervised learning approaches require many system interactions in order to learn a successful policy, which is undesirable because of time considerations and the risk of damaging the robotic system.

Research in the field of neuroscience by Johansson and Flanagan (2009) lead to the conclusion that pick-and-place tasks consist of seven stages: the reaching, loading, lifting, holding, replacing, unloading and releasing phase. This work inspired Romano et al. (2011) to introduce a human-based grasping controller for robotic grippers. The structure of the controller contains the same phases as found by Johansson and Flanagan (2009). Next to the structure of the controller, the input signals that are used in this approach are also based on human grasping. That is to say, the pressure sensors of the robotic gripper are employed in such a way that they imitate the Slow-Adapting type I (SA-I), Slow-Adapting type II (SA-II), and Fast-Adapting Type I (FA-I) tactile afferents which play an important role in human grasping. A limitation of this approach is the large number of control parameters that need to be tuned. Also, this method is not able to learn from failed grasps. Other limitations of this approach are the omission of visual information and the inability to find minimal grasping force in order to prevent damaging the object.

Another work that is closely related to our study, is the research performed by Misimi et al. (2018). They employ learning from demonstration for grasping food objects. A state representation is used, that among other things, consists of features that are extracted from

RGB-D camera images. This feature vector consists of the configuration and some geometric characteristics. The action space is formed by the configuration, preshape and finger figure of the end-effector and significant pressure threshold values for each finger. For evaluation of the method, the system is provided with 525 demonstrations through teleoperation. This resulted in a 75 % average success rate of the grasping system. Limitations of this approach are the need for a large number of demonstrations and the imperfect performance of the resulting grasping system. Moreover, it is likely that it is possible to achieve better performance while learning from fewer demonstrations, since the procedure involves learning trivial features. For example, the orientation angle of the object is extracted from the RGB-D image and the orientation of the end-effector could be derived from this directly. However, this relationship is learned in the approach by Misimi et al. (2018) which leads to failed grasps and an unnecessarily high number of required demonstrations.

1-3 Contributions

This study will contribute to the aforementioned related work as follows:

- The human-inspired controller from Romano et al. (2011) will be extended with visual sensory information. In this way, the grasp controller can make predictions on the required grasp force. Not only will this result in a closer resemblance to human grasping, but this will also result in more accurate predictions of the required grasp force. In the original controller that was proposed by Romano et al. (2011), the initial grasp force is set to a proportion of the force that is measured at contact. While this contact force will be similar for objects with similar material properties (e.g. same type of object), the required grasp force also heavily depends on the weight of an object. With visual sensory information, the size of an object can be measured which is in many cases correlated with the weight of an object. In this way, the robot is able to make more accurate predictions of the required grasp force, since it will not apply the same force for small and large objects with similar material properties.

- The human-inspired controller from Romano et al. (2011) will also be extended with learning capacities based on reinforcement learning principles. In this way, the system is able to apply grasp forces that are high enough to lift the object, but not too high in order to prevent damaging the (food) objects. Here the rewards the robot obtains depend on the grasp success, the desired force and the value of a signal that indicates slipping. In this way the robot tries to maximise the grasp success, while minimising the applied force and slipping. Also, the algorithm that is used for learning is a novel hierarchical version of Q-learning with incorporation of prior knowledge on the grasping problem in order to speed up convergence.

- With respect to the work by Misimi et al. (2018), which is the closest related work in the literature on grasping food objects with a learning approach, this work provides a novel approach that reduces the number of required learning interactions by reducing the learning problem to only learning the required grasp force. Namely, the path planning, the object configuration and finger shape do not have to be learned, but can be inferred directly from the RGB-D images for many pick-and-place problems.

For example, if one knows the orientation and location of a courgette, cucumber or eggplant, the center of mass can be approximated by calculating the centroid of the depth images and the finger shape could be a simple pinch grasp at the center of mass along the orientation of the object. Furthermore, the path planning problem can be solved using motion planning algorithms such as variants on Rapidly-exploring Random Trees (RRT) (Kuffner and LaValle, 2000; LaValle and Kuffner, 2001) or Probabilistic Roadmap Planners (PRM) (Kavraki et al., 1996; Bohlin and Kavraki, 2000). In this way, the required learning interactions can be reduced to the order of tens of trials instead of hundreds.

Chapter 2

Overview of the Grasp Approach

The goal of the grasp approach is to grasp deformable food objects with minimal force in order to prevent damage to the object, since food objects are often fragile and have to be in perfect shape to be saleable. The approach can be used for tasks that involve grasping deformable food objects, such as in sorting, packaging, placing, inspection and in the future maybe even in harvesting. In this work only the grasp procedure is studied and not the motion planning problems of moving the end-effector to the object location and moving the object to its goal location. However, we do refer to existing methods that can solve these motion planning problems in the case of static obstacles.

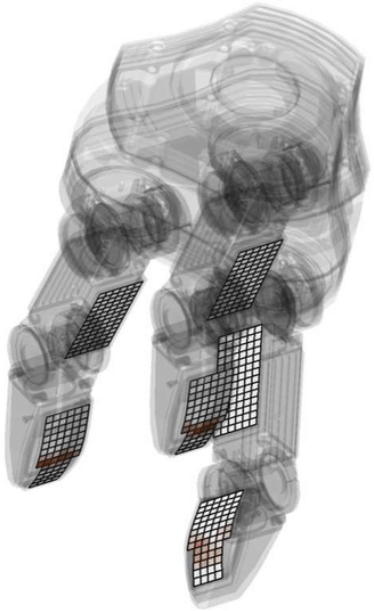
The hardware that is involved in this approach consists of an Red Green Blue - Depth (RGB-D) camera and a robotic hand. A brief introduction to the hardware is given in Section 2-1. Furthermore, the grasp approach involves three modules, i.e. the vision, control and learning module. The main principles of these modules are discussed in Sections 2-2, 2-3 and 2-4. They are discussed in depth in Chapters 3, 4 and 5, respectively.

2-1 Hardware

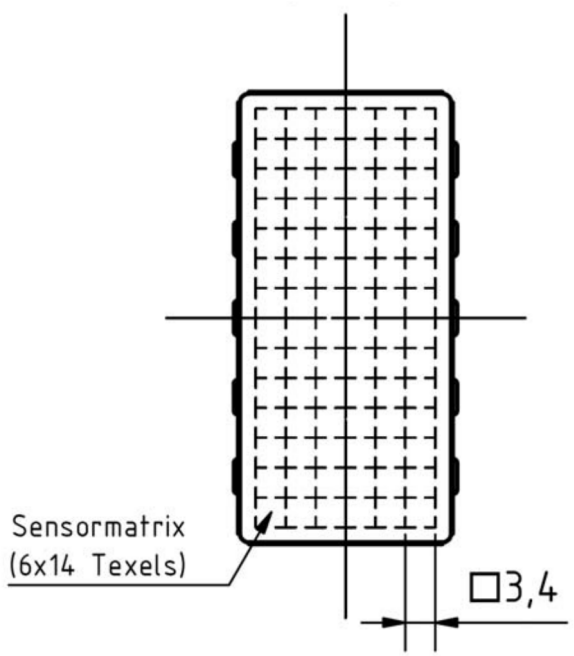
The hardware that is required for our grasp approach consists of an RGB-D camera and a robotic end-effector that is equipped with pressure sensors. The RGB-D camera is used for providing the vision module with depth information in the form of depth images. The RGB-D camera that is used in this work is the Asus Xtion Pro LIVE camera.

For our grasp approach, it is required that the end-effector is equipped with pressure sensors, because the approach involves force control. The end-effector that is used by us, is the 7 Degrees Of Freedom (DOF) Schunk Dexterous Hand (SDH). Since objects are grasped using pinch grasps in our approach, only two of the three fingers of the SDH are used for grasping, as shown in Figure 2-2. Therefore, the approach can also be used for grasping with parallel grippers that are equipped with pressure sensors. The pressure information that is provided by the SDH is The SDH has two tactile pads on each finger, which leads to a total of six pads. Figure 2-1a shows where the tactile pads are located on the hand. Each pad consists of

a matrix of texels, as shown in Figure 2-1b. At every texel location the pressure is measured. In our approach, the robotic hand sends the pressure information together with the joint states to the control module with a frequency of 30 Hz. The joint states consist of the joint positions and velocities.



(a) Image from Krug et al. (2016) showing the tactile pads on the SDH robotic hand.



(b) Image from the product sheet of the DSA 9205 tactile sensors (Weiss Robotics, 2008).

Figure 2-1: The SDH robotic hand is equipped with six DSA 9205 tactile sensors, i.e. two on each finger. These sensors consist of 6×14 texels for the pads on the proximal phalanges and 6×13 on the distal phalanges. The output of the sensor is a matrix containing a voltage value for each texel. A linear relation is used to convert these voltages to pressure values.

2-2 Vision Module

The purpose of the vision module is to detect objects and obtain visual features of the object. Object detection allows the robot to obtain the location and orientation of the object in order to move the end-effector towards the object. The visual features consist of length and height measures of the object and are obtained from depth information that is provided by the RGB-D camera. The depth information consists of a reference depth image that is made before the object is present and a depth image that includes the object. The reference and object depth image are made once for each object. The reference image is used to ease object detection. The depth images are processed using standard image processing techniques with operations such as threshold, dilation and erosion. This makes the vision module easy to implement, robust and omits the need for a database to train on. The vision pipeline as described in this work is able to detect, but not recognise objects. Therefore, the grasp approach is suited for settings in which a single type of object is to be grasped.

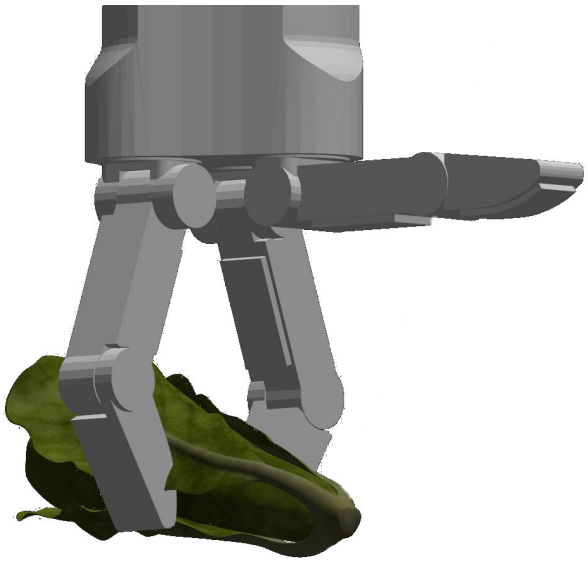


Figure 2-2: A pinch grasp with the SDH in simulation.

2-3 Control Module

The purpose of the control module is to track the desired grasp forces that are provided by the learning module and to provide tactile information to the learning module. The desired grasp force is defined as the desired mean value of the two finger tip forces. These finger tip forces are obtained by summing up the forces at every texel location. The forces at the texel locations are obtained by multiplying the pressure value at each texel location with the texel area. The design of the control module is based on the controller that was proposed by Romano et al. (2011), which in turn was based on research in the field of neurology by Johansson and Flanagan (2009). From the work by Romano et al. (2011) two concepts are used for our grasp control module, i.e. the action phases in grasping and the imitated tactile afferent signals.

The action-phases in the control module divide grasping into three phases, where in each phase a different type of control is used. The first phase is the closing phase. During this phase the fingers are closed by sending desired joint velocities to the robotic hand. The lifting and holding phase comes after the closing phase and starts when all fingers that are used for grasping are in contact with the object. During this phase the desired grasp force that is obtained from the learning module is tracked by sending desired joint velocities and accelerations to the robotic hand. The last phase of the grasp is the opening phase. This phase starts when the control module receives the command for opening. During this phase the fingers are opened by sending desired joint positions to the hand. The desired joint states are tracked by a low-level Proportional Integral Derivative (PID) controller in the robotic hand.

The second concept that is adopted from Romano et al. (2011), is the usage of the imitated tactile afferents that are proposed in that work. Tactile afferents are neurons that innervate the human skin. They provide sensory information on mechanical events in the human skin and are therefore responsible for the sense of touch in humans. Romano et al. (2011) proposed

processing procedures for pressure sensor signals in order to imitate the characteristics of the human tactile afferents. In the control module of our grasp approach, the pressure information that is obtained from the robotic hand is processed to imitate the tactile afferents and sent to the learning module. We use the notion of tactile information to refer to these imitated tactile afferents in our approach. The tactile information is sent to the learning module with the same frequency at which the robotic hand sends pressure information and joint states to the control module.

2-4 Learning Module

The learning module is responsible for setting the desired grasp force and does this with a reinforcement learning approach. The goal is to apply minimal grasp forces in order to prevent damage to the object, but at the same time the grasp force should be high enough to lift the object and prevent slipping. The learning module receives visual features of the object once per grasp attempt from the vision module. These visual features are received before the closing phase. Based on these visual features, the learning module chooses the initial desired grasp force. During the lifting and holding phase, the learning module receives tactile information from the control module. Every time the learning module receives the tactile information, it updates the desired grasp force, so in our implementation the desired grasp force is updated with a frequency of 30 Hz. However, during the lifting and holding phase the desired grasp force is chosen from a small set of forces, i.e. a set that consists of the initial desired grasp force and a small number of discrete forces that are higher than the initial force. Therefore, the initial desired grasp force also determines the grasp force range that is used during the lifting and holding phase.

The learning procedure consists of two parts: choosing the initial desired grasp force based on the visual features and adapting the desired grasp force during the lifting and holding phase based on the visual features and the tactile information. In this way, actions are initially chosen from a larger action space, while having a smaller state space, i.e. the visual features only. During the lifting and holding phase, the action space is smaller, while the state space is larger, since here the state consists of the visual features and the tactile information. The rationale is that this approach will converge faster as compared to a one-step approach, thanks to the initial reduction of the state space and the reduction of the action space during the lifting and holding phase. The rate of convergence is crucial, since learning interactions are expensive on real-world robotic equipment.

2-5 Combining the Modules

The modules modules are interconnected as shown in Figure 2-3. This figure shows the inputs and outputs of each module and the hardware components. In this figure, the dashed arrows are used for information that is received every once every grasp attempt. The solid arrows are used for information that is sent with a frequency of 30 Hz in our implementation.

In practice the grasp approach can be employed as follows. First, a reference depth image is made without the object that is to be grasped. Next, the object arrives – e.g. via a conveyor belt system – and a new depth image is made. A robotic manipulator moves the robot hand

towards the object location that was extracted from the depth image. The motion planning problem can be solved using MoveIt (Chitta et al., 2012), e.g. with the Open Motion Planning Library (OMPL) (Sucan et al., 2012). The finger shape is fixed, since for many objects a simple pinch grasp will suffice, but for more complex object geometries the finger shape can also be derived using software like GraspIt! (Miller and Allen, 2004). When the robot hand is put in place, the fingers will close and the initial desired grasp force is determined. As soon as the fingers are in contact with the object, the desired grasp force is adjusted based on the tactile information. During the lifting and holding phase, the manipulator moves the hand towards the goal location and once the goal location is reached, the hand will open to release the object. The pressure sensors can detect whether the grasp has failed. Also, the depth information can be used to check whether the object is close enough to the goal configuration. In case of a failed grasp, the robot will try again and the procedure is repeated.

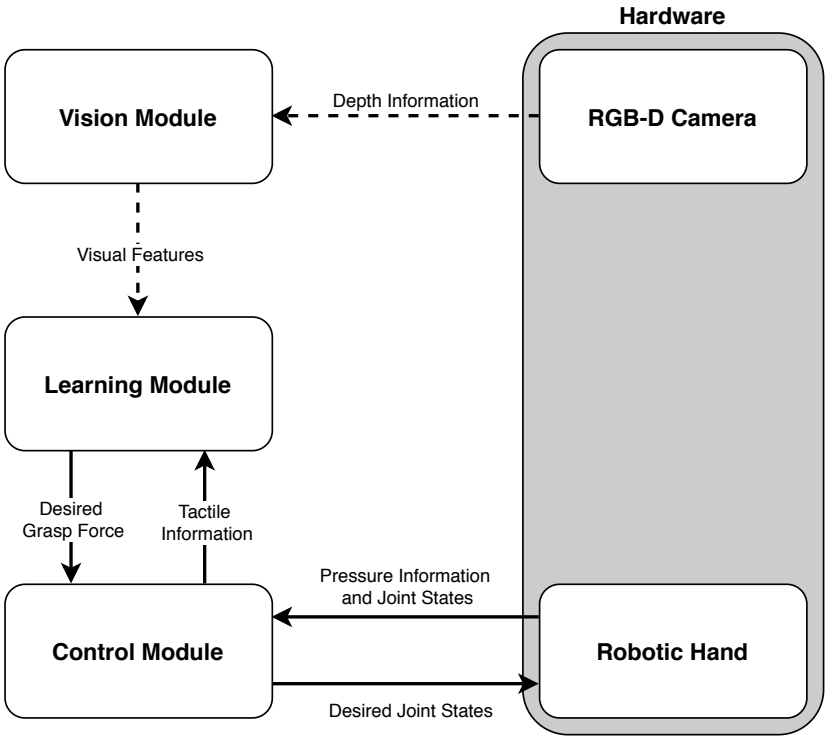


Figure 2-3: A flow diagram showing the information that each module sends and receives from the other modules and hardware components. The dashed arrows indicate information that is sent once every grasp attempt, while the solid arrows are used for information that is sent with a frequency of 30 Hz in our implementation.

Chapter 3

Vision Module

This chapter describes the way visual information is obtained and processed in the robotic grasp system. The visual information is obtained using a Red Green Blue - Depth (RGB-D) camera. The images shown in this chapter are obtained using an Asus Xtion Pro LIVE camera. This camera – which is shown in Figure 3-1 – is equipped with two audio sensors, an RGB camera, and an infrared sensor. Thanks to the infrared sensor the camera is capable of obtaining depth information.

Figure 3-1: Asus Xtion RGB-D camera (Asus, nd). Depth images obtained by this camera are used to localise the object and extract features for predicting the required load force. The depth images have a resolution of 640x480 pixels. The distance of use as specified by the manufacturer is between 0.8 m and 3.5 m.

The goal of the vision module is to detect the food object and extract visual features that are used for choosing the desired grasp force. For object detection and feature extraction, the depth information of the RGB-D camera is used. In this way the performance of the vision module does not depend on lighting conditions. A graphical summary of the vision pipeline is shown in Figure 3-2. All image processing operations are performed using the OpenCV image processing library (Bradski, 2000).

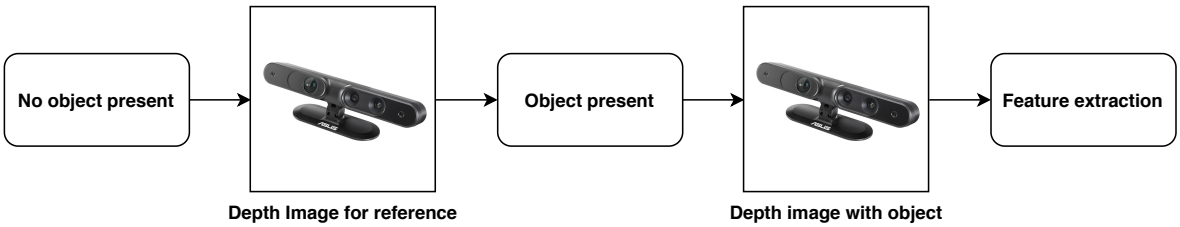


Figure 3-2: Flow diagram of the vision pipeline.

3-1 Preprocessing of the Depth Image

The first step of the vision pipeline consists of preprocessing the depth images. The process that an image undergoes before detection is visualised in Figure 3-3. This figure shows a typical example of what the steps look like before detecting the food object, which in this case is a courgette. Following the same procedure as in Misimi et al. (2018), a reference depth image (Figure 3-3a) is made before the object that is to be grasped is present. When the command is sent to pick up the object, a new depth image is made, which now also includes the object (Figure 3-3c). The next step is subtracting the depth image from the reference image (Figure 3-3d). This results in relatively high depth values at the pixel locations of the courgette. Due to the noise in the depth information and slight movements, there are still many non-zero pixels in the image that do not correspond to the object. Therefore, a threshold is applied to the image (Figure 3-3e). The threshold operation is the following:

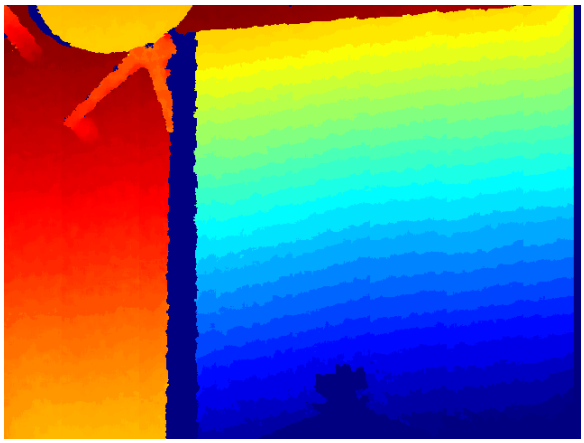
$$\text{dst}(x,y) = \begin{cases} \text{src}(x,y) & \text{if } \text{src}(x,y) > \text{ thresh} \\ 0 & \text{otherwise} \end{cases} \tag{3-1}$$

where $\text{src}(x,y)$ is the depth value of the input image at pixel (x,y), thresh is the threshold value that is used and $\text{dst}(x,y)$ is the depth value of the output image at pixel (x,y). The value for thresh that is used in Figure 3-3e is 15, which corresponds to 1.5 cm. After thresholding the image, there might still be some noise present in the image, as visible in Figure 3-3e. To get rid of the last noisy pixels, an opening operation is applied to the image. Opening is an operation that consists of an erosion step, followed by a dilation step. The kernel that is used for this operation is a 2×2 matrix consisting of ones and the number of iterations is five. This results in the image that is shown in Figure 3-3f.

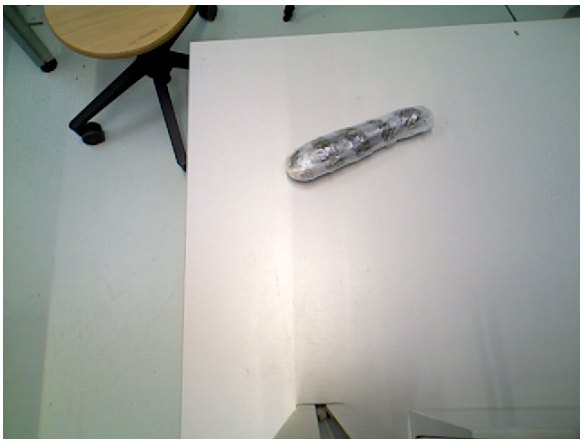
3-2 Object Position and Orientation Detection

The next step is to localise objects in the preprocessed depth image. This is done by first converting the image to a binary image and then searching for contours in the image. Since there still might be noise present in the image, contours that have an area under 500 or over 10^5 pixels are ignored. The other contours are treated as the contours of actual objects. Now that the contours of the objects are detected, the centre of the object is obtained by calculating the empirical mean as follows:

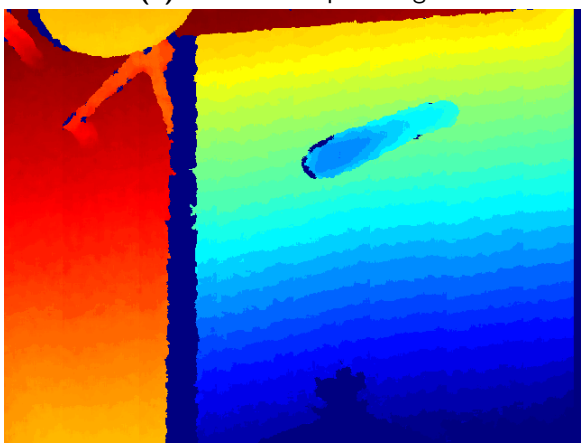
$$\mathbf{u}[\mathbf{j}] = \frac{1}{n} \sum_{i=1}^{n} \mathbf{X}[\mathbf{i},\mathbf{j}] \tag{3-2}$$



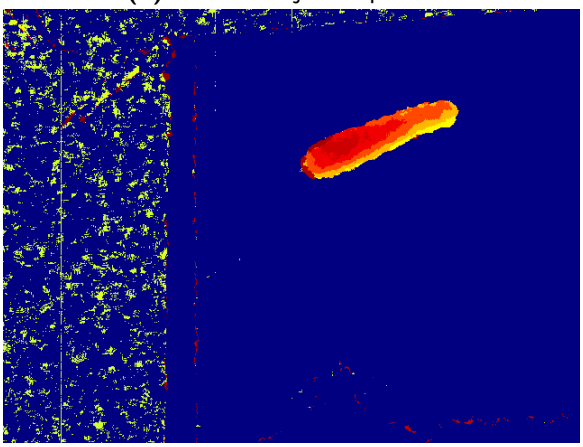
(a) Reference depth image.



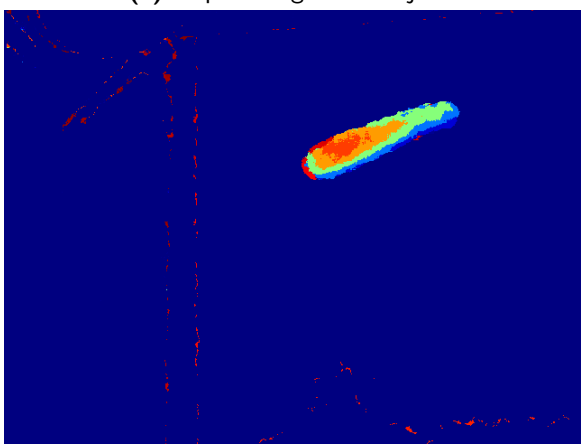
(b) Now an object is present.



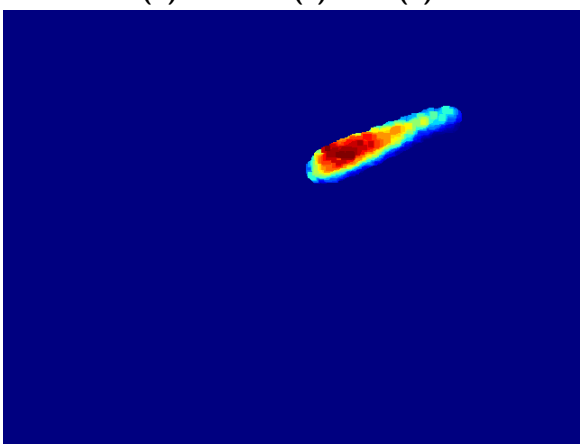
(c) Depth image with object.



(d) Subtract **(c)** from **(a)**.



(e) Threshold the image.



(f) Perform opening on **(e)**

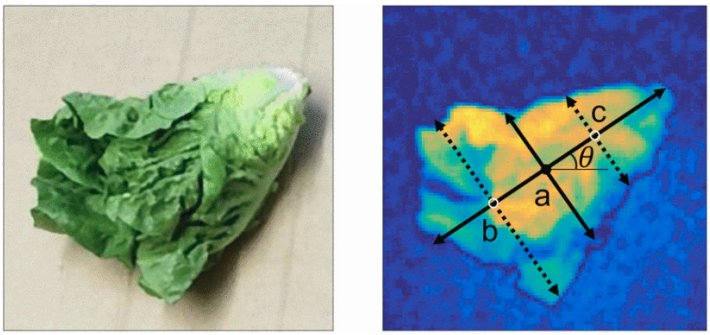
Figure 3-3: Depth images showing the steps that are involved in the image processing pipeline. It should be noted that the colours in the depth images are the result of applying a colour map after equalising the image. This is done to improve the contrast of the images. As a result, the same colour corresponds to different depth values in the different images.

with $\mathbf{X}$ a $2 \times n$ matrix containing the x and y locations of the pixels inside the contour, n the number of pixels inside the contour and $j = 1, 2$ representing the x and y dimensions of the image, respectively.

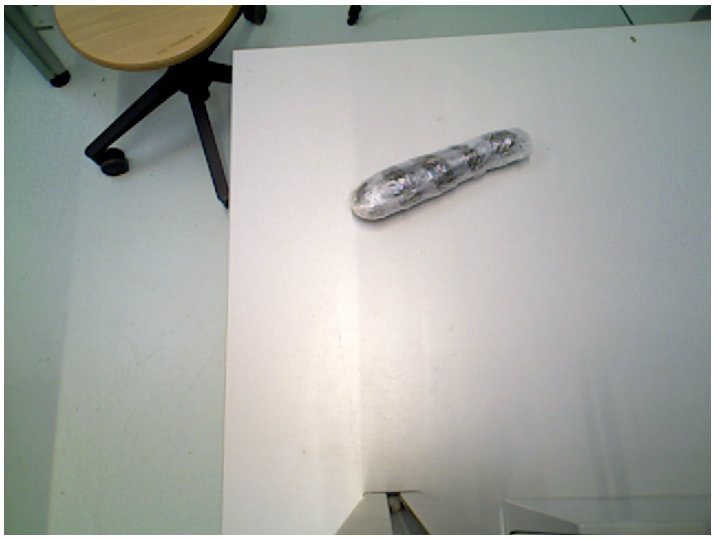
With the contour and centre of the object known, what rests is to find the orientation of the object. A frequently used strategy for detecting object orientation, is to perform Principle Component Analysis (PCA) (Wold et al., 1987; Yi and Marshall, 2000) and use the resulting eigenvectors as indicators for the orientation of the object. This works well for objects that have a shape with dimensions that are large in a certain direction, while small in another. The orientation of a long and thin object is easily detected using this technique. However, for objects that do not adhere to this property, it is not a suitable solution. For some sorts of leaf lettuces for example, this is not the case, so for such objects PCA is not a suitable strategy. In the rest of this work, the object orientation is obtained using PCA, since it is outside the scope of this work to study complex computer vision procedures.

3-3 Feature Extraction

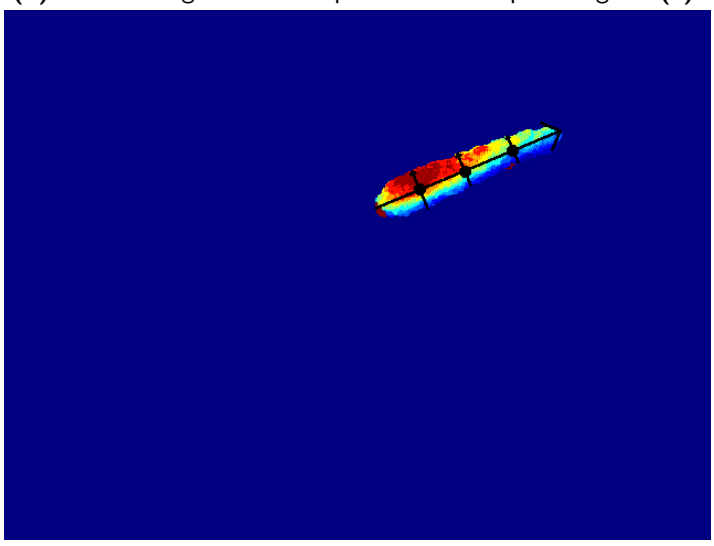
After detecting the object and obtaining its configuration, we can extract the features from the image that will be used for predicting the required load force. Following the work by Misimi et al. (2018), we extract the same features. Figure 3-4a shows a visual representation of these features. The features that are used by Misimi et al. (2018) are: the lettuce length l_a together with the width and height of the lettuce at point a (the centre), b (halfway the centre and the top) and c (halfway the centre and the bottom). The widths and heights are referred to as w_i and h_i, where w_i is the width and h_i the height of the lettuce at point $i \in \{a, b, c\}$. These features are obtained as follows. The orientation angle of the object θ as defined in Figure 3-4a was obtained as described in Section 3-2. The contour of the lettuce and centre point a are also known from the procedure in Section 3-2. First, the locations of point b and c have to be calculated. This done by drawing a line through the centre of the image with an angle of $\theta + \pi$ and calculating the points where this line intersects the contour. The two intersection points that are the furthest away from each other give the top and bottom point of the object (x_{p1}, y_{p1}) and (x_{p2}, y_{p2}), respectively. Now, point b is obtained as follows: $x_b = \text{round}((x_{p1} + x_a)/2)$, $y_b = \text{round}((y_{p1} + y_a)/2)$, where x_i and y_i are the coordinates of point i. In a similar fashion the pixel location of point c can be obtained. The widths and the length of the object can be calculated using the intersection points on the contour. However, these 2D pixel locations in the depth image have to be transformed to 3D coordinates in the camera frame, in order to make these features independent of the distance between the camera and the object. This transformation from 2D to 3D coordinates is obtained using the camera intrinsics. After this transformation, the values for the widths and length can be easily computed. For calculating h_a, h_b, h_c, the image of Figure 3-3d is used. This image is the difference between the depth image with the object and the reference image. Since this image can be quite noisy, h_a, h_b, h_c are not obtained directly from the depth image, but defined as the median of a 5×5 window at that location.



(a) Image from Misimi et al. (2018) showing the extracted features.



(b) Colour image that corresponds to the depth image in **(c)**.



(c) Depth image with arrows indicating the extracted features.

Figure 3-4: Image **(a)** comes from the work by Misimi et al. (2018) and shows a visual representation of the extracted features. Image **(b)** shows a colour image made by the RGB-D camera and image **(d)** shows the extracted features as described in Section 3-3.

3-4 Summary and Concluding Remarks

The vision module obtains depth images from the RGB-D camera and extracts from these images the object location, orientation and visual features. The vision module obtains two depth images per object, i.e. one reference image without the object and a depth image that includes the object. A number of preprocessing steps are performed to discriminate the object from the background in the image. The object orientation is obtained with PCA. This method is only valid for objects that have an oblong shape, e.g. cucumbers, egg plants, courgettes and leek. The visual features that are obtained are the ones from Misimi et al. (2018), which consist of length, width and height measures at three points along the principle axis of the object. The visual features are used by the learning module for choosing the desired grasp force. The object location and orientation can be used for determining the grasp location.

A limitation of the vision module is the fact that the module cannot recognise objects, but is only capable of detecting them. This means that the vision module cannot distinguish a cucumber from a courgette. Therefore, the visual features only contain useful information when a single type of food object is used. Another limitation of the vision module is that the orientation detection procedure, which is based on PCA, is only valid for long and thin shaped objects. Therefore, we propose to extend the vision module with an object recognition procedure and a more advanced procedure for detecting the orientation of the object. The You Only Look Once (YOLO) algorithm is a candidate for the object recognition part (Redmon et al., 2016; Simony et al., 2018; Tian et al., 2019). A procedure based on frustum pointnets by Qi et al. (2018) is a possible candidate for improving the object (orientation) detection from RGB-D data.

Chapter 4

Control Module

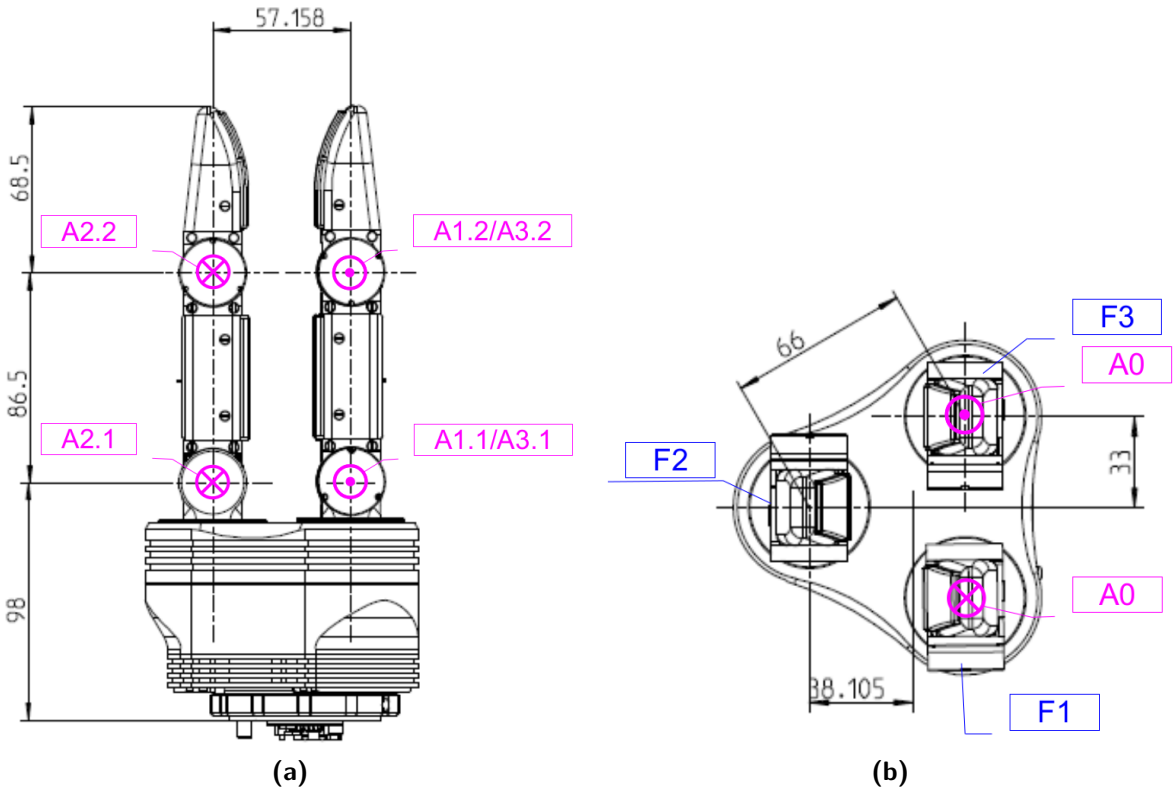
While grasping is a highly complex task – e.g. due to the discontinuous nature of the task, the large variety of possible surface properties and contact points that are changing during grasping – humans are able to grasp seemingly effortless. Humans are able to make accurate predictions of mechanical events while grasping based on visual and tactile sensory information (Johansson and Flanagan, 2009). Therefore, the mechanisms behind human grasping are used as inspiration for the novel grasp approach that is introduced in this work. In Appendix A, a very brief introduction to concepts from human grasping is given, since these are incorporated in the control module. Furthermore, the grasp controller in this work is based on the one that was introduced by Romano et al. (2011). Therefore, a brief description of the controller from that work is also given in Appendix A, in order to clarify the choices behind the grasp controller in this work.

4-1 Robotic Hand

The control module commands a robotic hand with pressure sensors in the finger tips, which in our case is the Schunk Dexterous Hand (SDH). The SDH is a three-fingered hand, but since we are studying pinch grasps, only two fingers are used. The SDH is a 7 Degrees Of Freedom (DOF) robot hand. The kinematics of this hand are shown in Figure 4-1. Finger i is referred to as Fi, with $i \in \{1, 2, 3\}$. Since we are dealing with pinch grasps, F3 is referred to as the left finger and F1 is referred to as the right finger. Furthermore, rotational axis j at finger i is referred to as A$j.i$, with $j \in \{1, 2\}$. An exception is axis A0, since this axis belongs to a coupled rotation of fingers 1 and 3. The joint positions of the robot are referred to as:

$$\boldsymbol{\theta}(t) = \begin{bmatrix} \theta_0(t) & \theta_1(t) & \theta_2(t) & \theta_3(t) & \theta_4(t) & \theta_5(t) & \theta_6(t) \end{bmatrix}^T \tag{4-1}$$

where $\theta_0(t)$ is the angle around axis A0, $\theta_1(t)$ is the angle around axis A2.1, $\theta_2(t)$ is the angle around axis A2.2, $\theta_3(t)$ is the angle around axis A.1.1, θ_4 is the angle around axis A.1.2, θ_5 is the angle around axis A.3.1 and θ_6 is the angle around axis A.3.2. All angles are in radians



(a) **(b)**

Figure 4-1: Kinematics of the SDH as published in the product documentation (SCHUNK Robotics, 2008). Figure **(a)** shows a side view of the hand, while figure **(b)** shows a top view. The notation in this figures is defined as follows. Finger i is referred to as Fi (in blue), with $i \in \{1, 2, 3\}$. Furthermore, rotational axis j at finger i is referred to as A$j.i$ (in pink), with $j \in \{1, 2\}$. An exception is axis A0, since this axis belongs to a coupled rotation of fingers 1 and 3. Finally, if the positive direction of the axis is counter-clockwise, this is indicated with the $\odot$ sign. If the positive direction is clockwise, this is indicated with the $\otimes$ sign. The dimensions are shown in millimetres.

and the configuration in Figure 4-1 is defined as the zero configuration. Furthermore, the joint velocities and accelerations are referred to as $\dot{\boldsymbol{\theta}}(t)$ and $\ddot{\boldsymbol{\theta}}(t)$, respectively.

On each finger of the SDH two tactile sensor pads are located, i.e. one on the distal finger part and one on the proximal finger part. The pressure sensors on the proximal phalanges consist of 6×14 texels (discrete sensor cells), while the sensor pads on the distal phalanges consist of 6×13 texels. The working principle of these resistive tactile sensors is as follows. At every texel, the conductivity of the rubber material is measured. If the material is compressed, the electrical resistance of the material decreases. In this way, the voltage at each texel is related to the pressure at that location (Weiß and Worn, 2005; Schneider et al., 2009).

The robot hand is commanded using Robot Operating System (ROS) with open source packages (Fraunhofer IPA, nd) in combination with packages written by my supervisor ir. P. Kulkarni and packages written by the author. In our implementation, the robot hand publishes its joint positions and velocities 30 times per second, as well as the pressure values at each texel.

The SDH contains a low level Proportional Integral Derivative (PID) controller, that is designed by the manufacturer. This PID controller is designed to track desired joint velocities and accelerations, but can also be used for position control.

4-2 Imitated Tactile Afferents

As mentioned before, the grasp controller we propose is based on the one from Romano et al. (2011). From this work, the imitated Slow-Adapting type I (SA-I) and Fast-Adapting Type I (FA-I) afferent signals are used. The imitated FA-I signal is however not directly used by the control module, but is part of the tactile information that is sent to the learning module. The imitated SA-I signal is the mean force that is measured at the finger tips. So, for the left and right finger tip, the finger tip force is:

$$F_{gl}(t) = \sum_{i=1}^{6} \sum_{j=1}^{13} f_{l(i,j)}(t) \tag{4-2}$$

$$F_{gr}(t) = \sum_{i=1}^{6} \sum_{j=1}^{13} f_{r(i,j)}(t) \tag{4-3}$$

where $f_{l(i,j)}$ is the force that is measured at texel location (i, j) on the left finger tip, while $f_{r(i,j)}$ is the force that is measured at texel location (i, j) on the right finger tip. The grasp force (the imitated SA-I signal) is defined as the mean of these two finger tips forces:

$$F_g(t) = \frac{1}{2} \left(F_{gl}(t) + F_{gr}(t) \right) \tag{4-4}$$

The grasp force $F_g(t)$ is the force that is controlled during the lifting and holding phase.

The imitated FA-I signal, serves to indicate force disturbances such as slipping and is defined for the left and right finger tip as:

$$\widetilde{F}_{gl}(z) = \sum_{i=1}^{6} \sum_{j=1}^{13} H_F(z) f_{l(i,j)}(z) \tag{4-5}$$

$$\widetilde{F}_{gr}(z) = \sum_{i=1}^{6} \sum_{j=1}^{13} H_F(z) f_{r(i,j)}(z) \tag{4-6}$$

where $H_F(z)$ is a discrete-time first order Butterworth high-pass filter with a 5 Hz cut-off frequency. The cut-off frequency is set to 5 Hz by Romano et al. (2011), since the response frequency of the FA-I afferent is 5-50 Hz in humans. The number of dimensions of the state vector in the learning module should be as low as possible, we reduce the signals from Equations 4-5 and 4-6 to a single signal:

$$\widetilde{F}_g(z) = \max(\text{abs}(\widetilde{F}_{gl}(z), \widetilde{F}_{gr}(z))) \tag{4-7}$$

The reason for taking the maximum of the absolute values is that this signal is used by Romano et al. (2011) to detect slipping. In that work, the robot assumed that the object was slipping if the value for $\max(\text{abs}(\widetilde{F}_{gl}(z), \widetilde{F}_{gr}(z)))$ was higher than a certain threshold. Together, the values for $F_g(t)$ and $\widetilde{F}_g(z)$ form the tactile information that is sent to the learning module.

4-3 Action-Phases

The action-phases of our grasp controller are based on the ones from Romano et al. (2011). In that work, there are five action-phases: the closing, loading, lifting and holding, replacing, unloading and opening phase. These action-phases are described in Appendix A. Unlike the controller from Romano et al. (2011), the control scheme in our work consists of three action-phases: reaching, lifting and holding and opening, since this simplifies the control structure and in this way only the most critical action-phases are present in the structure. These phases are the most critical, since they correspond to transitions from velocity to force control and from force control to position control. A flow diagram of the action-phases of the grasp controller is shown in Figure 4-2.

Before the reaching phase, the hand is put in its pre-grasp posture, in order to create space between the fingers. This pre-grasp posture is defined as $\boldsymbol{\theta}(0) = \begin{bmatrix} \frac{\pi}{2} & -\frac{\pi}{2} & 0 & -\frac{\pi}{4} & \frac{\pi}{6} & -\frac{\pi}{4} & \frac{\pi}{6} \end{bmatrix}^T$ rad. When a command is sent to the robot to start the grasp procedure – e.g. when the end-effector is put in place for grasping an object – the reaching phase commences. During this phase, velocity control is used to move the left and right finger towards each other. This is done by sending the following desired joint velocities to the low level controller of the robotic hand:

$$\dot{\boldsymbol{\theta}}_{\text{des}}(t) = \begin{bmatrix} 0 & 0 & 0 & \dot{\theta}_{g,\text{des}}(t) & 0 & \dot{\theta}_{g,\text{des}}(t) & 0 \end{bmatrix}^T \tag{4-8}$$

where $\dot{\theta}_{g,\text{des}}$ is the desired grasp velocity, which is set to VCLOSE during the reaching phase, where VCLOSE is a control parameter. The reaching phase ends when both the left and the

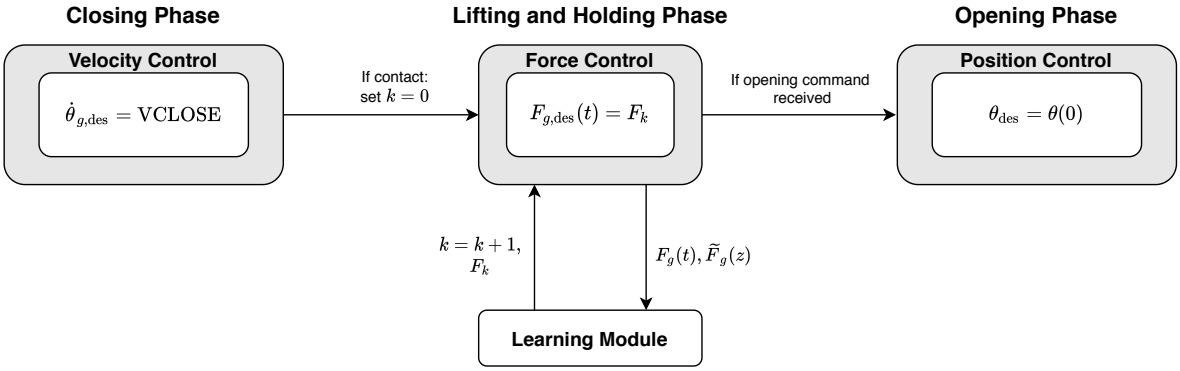


Figure 4-2: Flow diagram of the action-phases in the grasp controller that is proposed in this work. Here $\dot{\theta}_{g,des}$ is the desired grasp velocity, $F_{g,\text{des}}(t)$ is the desired grasp force at time t, F_k is the desired force that is obtained from the learning module at time step k, $\boldsymbol{\theta}_{\text{des}}$ is a vector describing the desired joint positions and $\boldsymbol{\theta}(0)$ is the pre-grasp joint pose. The signals $F_g(t)$ and $\widetilde{F}_g(z)$ form the tactile information that is sent to the learning module, where $F_g(t)$ is the grasp force and $\widetilde{F}_g(t)$ is the force disturbance. Finally, VCLOSE is a control parameter that defines the closing velocity of the hand during the reaching phase.

right finger tip are in contact with the object. A tactile sensor pad is considered to be in contact if one of the texel voltages is over 10 mV. Once in contact, the reaching phase ends and the lifting and holding phase starts.

During the lifting and holding phase, force control is employed to control the grasp force. However, force control is achieved by lower level velocity and acceleration control, since the software of the hand does not support torque control. When the lifting and holding force starts, the time instance k is set to zero. With the same frequency at which the robot hand publishes joint states and pressure values (30 Hz), the grasp controller receives a desired grasp force F_k from the learning module and every time this happens, k is incremented with one. The desired grasp force is set to the desired force from the learning module: $F_{g,\text{des}}(t) = F_k$ How this desired force is determined, is described in Chapter 5. Tracking of the grasp force is realised as follows. At every time instance k, the difference between the grasp force and the desired grasp force is calculated:

$$\Delta F(t) = F_{g,\text{des}}(t) - F_g(t). \tag{4-9}$$

The desired grasp force is tracked by setting the desired grasp joint velocity and acceleration proportional to the grasp force error:

$$\dot{\theta}_{g,des} = K_v \Delta F(t) \tag{4-10}$$

$$\ddot{\theta}_{g,des} = K_a \Delta F(t) \tag{4-11}$$

with

$$K_v = \begin{cases} \text{KVCLOSE} & \text{if} \quad \Delta F(t) \geq 0 \\ \text{KVOPEN} & \text{if} \quad \Delta F(t) < 0 \end{cases} \tag{4-12}$$

and

$$K_a = \begin{cases} \text{KACLOSE} & \text{if} \quad \Delta F(t) \geq 0 \\ \text{KAOPEN} & \text{if} \quad \Delta F(t) < 0 \end{cases} \tag{4-13}$$

where KVOPEN, KVCLOSE, KAOPEN and KACLOSE are control parameters. This control law can be seen as a form of PI control in the case of an object with non-negligible stiffness. There are different gains specified for closing and opening actions, since contact can be lost quickly during opening. Therefore the closing gains KVCLOSE and KACLOSE are advised to be set higher than KVOPEN and KAOPEN. Furthermore, the finger tips are used to control the force, so the desired joint velocities and joint accelerations that are sent to the lower level PID controller are:

$$\dot{\boldsymbol{\theta}}_{\text{des}} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dot{\theta}_{g,\text{des}} & 0 & \dot{\theta}_{g,\text{des}} \end{bmatrix}^T \tag{4-14}$$

$$\ddot{\boldsymbol{\theta}}_{\text{des}} = \begin{bmatrix} 0 & 0 & 0 & 0 & \ddot{\theta}_{g,\text{des}} & 0 & \ddot{\theta}_{g,\text{des}} \end{bmatrix}^T \tag{4-15}$$

The lifting and holding phase ends when the end-effector is at the object goal position or when the grasp has failed.

The last action-phase in the grasp controller is the opening phase. During this phase the hand is switched to position control to bring the hand back to the pre-grasp pose $\boldsymbol{\theta}(0)$:

$$\boldsymbol{\theta}_{\text{des}} = \boldsymbol{\theta}(0) \tag{4-16}$$

Position control is realised by the low level PID controller.

4-4 Summary and Concluding Remarks

The control module has two main functions. First of all, it is responsible for commanding the robotic hand. Secondly, it processes the pressure information from the robotic hand in order to provide lower dimensional tactile information to the learning module. Tracking of the desired grasp force is realised by sending desired joint states to the SDH, which is the robotic hand that is used in this work. The low level PID controller of the hand converts these desired joint states into joint torques. The control module has three action-phases: the closing, the lifting/holding and the opening phase. Velocity control, force control and position control are employed during these phases, respectively. During the lifting and holding phase, a desired grasp force that is obtained from the learning module is tracked. This is done by sending desired joint velocities and accelerations to the robotic hand. The control module is also responsible for processing the pressure information. Since the pressure information that is obtained from the robotic hand is high-dimensional, dimensionality reduction is performed. This is done by using imitated tactile afferents as proposed by Romano et al. (2011).

A limitation of the control module is that force control is realised through velocity and acceleration control of the robotic hand. We propose to use a form of direct torque control for future versions of the control module. This will simplify the control structure and might improve tracking, because it allows for higher sampling frequencies. Direct torque control is not employed in this work, since it required writing extra ROS packages for commanding the robotic hand, which was not possible due to limited time resources. Another limitation of the control module is that the control gains have to be tuned heuristically. We propose to study the possibility of using adaptive control gains for a future version of the control module. In this way, the control module will be able to accommodate for varying damping and stiffness properties of objects.

Chapter 5

Learning Module

The goal of the learning module is to learn from visual features and tactile information to grasp objects with minimal force, while maximising the success rate and minimising the force disturbance. The visual features are obtained with an RGB-D camera as explained in Chapter 3. Applying the right grasp force is critical, since too low forces will result in a failed grasp, while too high forces will damage the food objects. We have chosen for a reinforcement learning approach, because we are dealing with food objects that have a large biological variety and are therefore hard to model. Another benefit of a reinforcement learning approach is that the robot is able to grasp differently after a failed grasp and will therefore not make the same mistakes over and over again.

The learning approach consists of two parts, i.e. an outer loop that is responsible for choosing the initial desired grasp force based on the visual features and an inner loop that is responsible for adapting the desired grasp force based on the tactile information at that time step in combination with the visual features. Whereas the outer loop takes one action every episode (grasp attempt), the inner loop takes an action every time step, which means in our case 30 times per second. The reason for this two-step approach, is that in this way a force range can be chosen based on the visual features and that the force can be adapted based on the tactile information, e.g. to increase the force when the object starts slipping. The idea is that in this way the method converges faster as compared to the case where the force is chosen from the full force range based on the tactile information and the visual features. Also, in this way the desired grasp force will not change too much during the grasp. This is desirable, since a strongly varying desired grasp force can lead to instability and slipping.

5-1 Initial Desired Grasp Force

The first step in the learning algorithm is to choose an initial force based on the visual features that are obtained from the vision module. The algorithm we propose is a modified version of the Q-learning algorithm (Watkins and Dayan, 1992).

The procedure for choosing the initial desired grasp force can be described as follows. The agent observes object features $\boldsymbol{s}_{0,i} \in \mathbb{N}_0^{n_f}$ every episode i, where n_f is the number of features that is used. So, we have:

$$\boldsymbol{s}_{0,i} = \Big[x_{1,i}, x_{2,i}, ..., x_{n_f,i}\Big]^T \tag{5-1}$$

where $x_{1,i}, x_{2,i}, ..., x_{n_f,i}$ are discrete visual feature values that are observed at episode i. These discrete feature values are obtained through discretisation of the original features from the vision module. The features that are used, are length measures of the object, as described in Chapter 3. Based on these features, the agent takes in episode i action $a_{0,i} \in \mathcal{A}_0$. Here we have:

$$\mathcal{A}_0 = \{0, 1, ..., a_\mathrm{n} - 1\} \tag{5-2}$$

with a_n the number of actions. Each action corresponds to a grasp force value that is obtained by discretisation of the force range in a_n values. However, this action is not only the initial force that is tracked, but also defines the range of actions that can be taken for adaptation of the desired grasp force. More specifically, for adaptation of the desired grasp force, the agent can take at time step k action $a_k \in \mathcal{A}_0$, where we have:

$$a_{0,i} \leq a_k \leq (a_{0,i} + n_\mathrm{adapt}) \tag{5-3}$$

where n_adapt is the number of extra actions that can be taken for adaptation of the desired grasp force.

When action $a_{0,i}$ is taken, the grasp force adaptation starts. Grasp force adaptation continues until the episode has ended, which can either have failed or succeeded. When the grasp has ended, the agent observes $r_{0,i}$ and updates the state-action value function $Q_0(\boldsymbol{s}_0, a_0)$. The reward function R_0 is defined as follows:

$$R_0(a_{0,i}, \text{failure}) = \begin{cases} -10 & \text{if failure} \\ 1 - 0.5\frac{a_{0,i}}{a_\mathrm{n} - 1} & \text{otherwise} \end{cases} \tag{5-4}$$

In this way, rewards are positive in case of a successful grasp and negative in case of a failure. Also, the reward is higher when $a_{0,i}$ is lower. Lower values for $a_{0,i}$ correspond to lower reference grasp force values. The idea behind this reward function, is that in this way the agent will try to maximise the number of successful grasps, while at the same time tries to minimise the force that is applied.

Choosing the initial desired grasp force can be seen as a multi-armed bandit problem. Prior knowledge on this problem is incorporated in the learning algorithm, by making the following assumptions: *"if the grasp is successful after applying a certain force for certain feature values, then this grasp would also have succeeded if a higher force was applied or if any of the feature values would have been lower"* and *"if the grasp fails after applying a certain force for certain feature values, then this grasp would also have failed if a lower force was applied or if any of the feature values would have been higher"*. The idea behind these assumptions is that a higher force is required when the grasp fails and that if the object is larger – the object features consist of length measures in our case – a larger force is required than for smaller objects. Please do keep in mind that the grasp approach is suitable for a single type of food object,

because otherwise the assumptions do not make sense, because the density can vary a lot for different types of objects. These assumptions are incorporated in the algorithm as follows. Instead of updating only the state-action values that belong to $\boldsymbol{s}_{0,i}$ and $a_{0,i}$, all state-action values of which the grasp success can be inferred based on the assumptions are updated. This leads to the update rule that is shown in Algorithm 1.

Algorithm 1 Q_0 Update rule based on assumptions

Require: Number of actions a_n, number of features n_f, learning rate α_0, reward function $R_0(a_0, \text{failure})$, state-action value function $Q_0(\boldsymbol{s}_0, a_0)$, episode i, initial action $a_{0,i}$, feature vector $\boldsymbol{s}_{0,i} = \begin{bmatrix} x_{1,i}, x_{2,i}, ..., x_{n_f,i} \end{bmatrix}^T$

if Grasp has failed **then**
  for all $x_1 \geq x_{1,i},\ x_2 \geq x_{2,i},\ ...,\ x_{n_f} \geq x_{n_f,i}$ **do**
    $\boldsymbol{s}_0 \leftarrow \begin{bmatrix} x_1, x_2, ..., x_{n_f} \end{bmatrix}^T$
    for $a_0 \leq a_{0,i}$ **do**
      $r_0 \leftarrow R_0(a_0, \text{True})$
      $Q_0\left(\boldsymbol{s}_0, a_0\right) \leftarrow Q_0\left(\boldsymbol{s}_0, a_0\right) + \alpha_{0,i}\Big(r_0 - Q_0\left(\boldsymbol{s}_0, a_0\right)\Big)$
else
  for all $x_1 \leq x_{1,i},\ x_2 \leq x_{2,i},\ ...,\ x_{n_f} \leq x_{n_f,i}$ **do**
    $\boldsymbol{s}_0 \leftarrow \begin{bmatrix} x_1, x_2, ..., x_{n_f} \end{bmatrix}^T$
    for all $a_0 \geq a_{0,i}$ **do**
      $r_0 \leftarrow R_0(a_0, \text{False})$
      $Q_0\left(\boldsymbol{s}_0, a_0\right) \leftarrow Q_0\left(\boldsymbol{s}_0, a_0\right) + \alpha_{0,i}\Big(r_0 - Q_0\left(\boldsymbol{s}_0, a_0\right)\Big)$

5-2 Adaptation of the Desired Grasp Force

The goal of the adaptation of the desired grasp force is to reduce slipping effects, while still minimising the grasp force and maximising the success. In order to realise this, Q(λ) is used (Sutton and Barto, 2018). This algorithm is chosen for the following reasons. First of all, we are dealing with a large state-space that consists of visual features and the tactile information. At the same time, it is desirable to learn a decent grasp behaviour from relatively few interactions, since the robotic equipment is expensive and interactions can possibly damage the system. Therefore, approaches that require many system interactions – such as deep reinforcement learning approaches – are not appropriate. To reduce the dimensionality of the problem, an approach is chosen with discrete state-action spaces. Furthermore, eligibility traces are chosen for this approach, since we are dealing with a delayed reward. Namely, if the grasp fails, there is a delay between the moment the object starts slipping and the moment the grasp is considered to have failed. The idea is that thanks to the eligibility traces, the agent will learn which states are "dangerous" – i.e. might result in a failed grasp – faster than without eligibility traces.

The dimensions of the state-space are different for grasp force adaptation as compared to choosing the initial desired grasp force, since here the state also consists of the tactile information. The tactile information consists of the imitated tactile afferents $F_g(t)$ and $\widetilde{F}_g(z)$ as

described in Chapter 4. Therefore, the state-action value function for adaptation $Q(\boldsymbol{s}_k, a_k)$ is higher dimensional than $Q_0(\boldsymbol{s}_0, a_0)$. For adaptation, $\boldsymbol{s}_0$ is extended with the values for the grasp force $f_k \in \mathbb{N}_0$ and the grasp force disturbance $\widetilde{f}_k \in \mathbb{N}_0$. The values for f_k and $\widetilde{f}_k$ are obtained after discretisation of $F_g(t)$ in f_n and $\widetilde{F}_g(z)$ in $\widetilde{f}_\mathrm{n}$ values, respectively. Therefore we have:

$$f_k \in \{0, 1, ..., f_\mathrm{n} - 1\} \tag{5-5}$$

$$\widetilde{f}_k \in \{0, 1, ..., \widetilde{f}_\mathrm{n} - 1\} \tag{5-6}$$

This leads to the following state at time step k:

$$\boldsymbol{s}_k = \begin{bmatrix} x_{1,i} & x_{2,i} & ... & x_{n_f,i} & f_k & \widetilde{f}_k \end{bmatrix}^T \tag{5-7}$$

The reward function for adaptation is designed to minimise both the applied force and slipping. Therefore the following reward function is used:

$$R(a_k, \widetilde{f}_k, \text{failure}) = \begin{cases} -1000 & \text{if failure} \\ 1 - 0.5(\frac{a_k}{a_\mathrm{n} - 1} + \frac{\widetilde{f}_k}{\widetilde{f}_\mathrm{n} - 1}) & \text{otherwise} \end{cases} \tag{5-8}$$

In this way, the agent is not only rewarded for low force values a_k, but also for low force disturbance values $\widetilde{f}_k$.

When combining the procedure for choosing the initial desired grasp force with the adaptation part, we get the algorithm that is shown in Algorithm 2. The algorithm is called Learning of Initial Force and Tuning (LIFT). A flow diagram of the procedure is shown in Figure 5-1.

5-3 Taking Actions

In our implementation, a simple ϵ-greedy approach is used for taking actions in both choosing the initial desired grasp force and for adaption, where the probability is ϵ that a greedy action is taken. In order to allow different values of ϵ for the initial desired grasp force and adaptation procedures, we have $\epsilon_{0,i}$ for the initial step and ϵ_i for adaptation. The ϵ-greedy approach for choosing the initial desired grasp force is modified slightly, to steer exploration. Namely, the agent takes the middle action if multiple actions have the same Q-value. Therefore, the agent takes for example the action a_0 halfway its force range the first episode. In order to reduce exploration and stimulate exploitation over time, a decay factor is used to diminish ϵ_i every episode i:

$$\epsilon_{i+1} = 1 - c_\epsilon(1 - \epsilon_i) \tag{5-9}$$

where c_ϵ is a factor between 0 and 1. A similar procedure is used to diminish $\epsilon_{0,i}$. Furthermore, the learning rates of the initial and adaptation procedures are also diminished every episode i using the following rule:

$$\alpha_{i+1} = c_\alpha \alpha_i \tag{5-10}$$

where c_α is a factor between 0 and 1.

Algorithm 2 LIFT

Require: Number of episodes n_{episodes}, number of extra actions for force adaptation n_{adapt}, learning rate α_i, discount factors γ, decay weight λ

$i \leftarrow 0$
Initialise $Q_0(\boldsymbol{s}_0, a_0)$
Initialise $Q(\boldsymbol{s}, a)$
while $i < n_{\text{episodes}}$ **do**
    $k \leftarrow 0$
    $E(\boldsymbol{s}, a) = 0$, for all s, a
    $\boldsymbol{s}_{0,i} \leftarrow$ Observe visual features
    Choose $a_{0,i}$ derived from $Q_0(\mathbf{s_0}, a_0)$
    while $\boldsymbol{s_k}$ is not terminal **do**
        Choose a_k derived from $Q(\mathbf{s}, a)$, where $a_{0,i} \leq a_k \leq a_{0,i} + n_{\text{adapt}}$
        Take action a_k, observe r_{k+1} and $\boldsymbol{s}_{k+1}$
        $a_{k+1} \leftarrow \arg\max_a Q(\boldsymbol{s}_{k+1}, a)$, where $a_{0,i} \leq a_{k+1} \leq a_{0,i} + n_{\text{adapt}}$
        $\delta \leftarrow r_{k+1} + \gamma Q(\boldsymbol{s}_{k+1}, a_{k+1}) - Q(\boldsymbol{s}_k, a_k)$
        $E(\boldsymbol{s}_k, a_k) \leftarrow E(\boldsymbol{s}_k, a_k) + 1$
        for all $\boldsymbol{s}$, a **do**
            $Q(\boldsymbol{s}, a) \leftarrow Q(\boldsymbol{s}, a) + \alpha_i \delta E(\boldsymbol{s}, a)$
            $E(\boldsymbol{s}, a) \leftarrow \gamma\lambda\ E(\boldsymbol{s}, a)$
        $k \leftarrow k + 1$
    Observe $r_{0,i}$ and update $Q_0(\boldsymbol{s}_0, a_0)$ as in Algorithm 1
    $i \leftarrow i + 1$

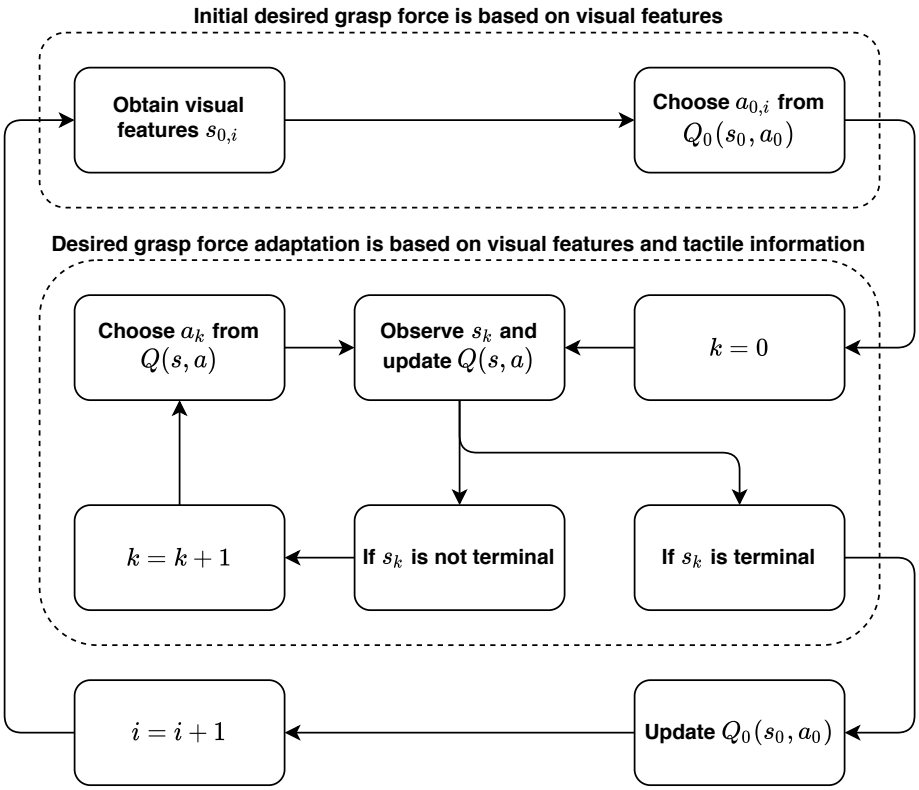


Figure 5-1: A graphical overview of the learning method. Two value functions are trained: $Q_0(\boldsymbol{s}_0, a_0)$ and $Q(\boldsymbol{s}, a)$. The difference between these two value functions is that $Q_0(\boldsymbol{s}_0, a_0)$ is trained to choose an initial action based on the visual features, while $Q(\boldsymbol{s}, a)$ is trained to adapt this desired grasp force based on the visual features and the tactile information.

5-4 Summary and Concluding Remarks

The learning module is responsible for setting the desired grasp force. The learning module has two types of sensory information at its disposal, i.e. visual features and tactile information. A novel algorithm is proposed for doing this: LIFT. This algorithm is a two-step approach that consists of an initial procedure and an adaptation procedure. The initial procedure is responsible for choosing the initial desired grasp force, which can be adapted during the lifting and holding phase by the grasp force adaptation procedure. During the adaptation procedure, the desired grasp force is chosen from a smaller action set as compared to the action set from the initial procedure. This smaller action set is based on the initial desired grasp force. The initial procedure is a modified version of the Q-learning algorithm, but with a different update rule. This modified update rule is based on assumptions on the grasp problem. Adaptation of the desired grasp force is done using the Q(λ) algorithm with a state representation that consists of the visual features and the tactile information.

For future work, we propose to study black-box optimization procedures for choosing the initial grasp force based on the visual features. During the initial step, choosing the optimal initial desired grasp force can be seen as a black-box optimization problem. Namely, the problem can be written as follows:

Let $\boldsymbol{s}_0 \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, the problem is to find $\max_{\boldsymbol{s}_0} f(\boldsymbol{s}_0)$, where we can only evaluate $f(\boldsymbol{s}_0)$ for any $\boldsymbol{s}_0 \in \mathbb{R}^n$, where f is a mapping from the visual features to rewards.

A class of algorithms that can be used to solve this problem is the class of stochastic search algorithms (Spall, 2005). A candidate algorithm is Covariance Matrix Adaptation (CMA) (Hansen, 2006), which is an evolution strategy that involves estimating local gradients.

Chapter 6

Simulation

There are various reasons for creating an environment with a simulated version of the robot. Firstly, in this way it is possible to test the approach without the physical version of the robot, which makes testing outside of the lab possible. Secondly, it is safer than testing directly on the real setup, because the code might contain bugs and the program could crash. Since we are dealing with expensive robotic equipment it is of great importance that the approach is safe to use and will not result in unexpected behaviour. Furthermore, the evaluation of the approach can involve many iterations of certain procedures, for example when testing a learning scheme. This results in wear and tear and it is therefore preferable to evaluate these procedures in simulation. Also, it is a fast way to check newly implemented parts, since it does not require to set up the robotic equipment in advance.

6-1 Gazebo

The simulation environment of the robot is created in Gazebo. Gazebo is an open-source multi-robot 3D simulator with Robot Operating System (ROS) integration and four different physics engines (Koenig and Howard, 2004). Alternative robotics simulators are Mujoco (Todorov et al., 2012) and V-REP (Rohmer et al., 2013). There are three reasons why we chose for Gazebo instead of one of its alternatives. First of all, the Gazebo environment has full ROS integration and therefore it allows us to use the same code for simulation as for the real world setup, which saves time and allows for bug checking of the code in simulation. Secondly, there was already open source code available for simulation of the Schunk Dexterous Hand (SDH) (Fraunhofer IPA, nd). Finally, the author has more experience with Gazebo than with the other simulators, which also contributes to the choice for Gazebo.

6-2 Environment

The simulation environment is designed such that it allows to test and evaluate the grasp approach. A minimal simulation environment was created with the SDH, a simulated Red

Green Blue - Depth (RGB-D) camera and an object that is to be grasped. The simulation environment does not include a robotic manipulator for performing pick-and-place tasks, since this would increase the complexity of the simulations and thereby also increase the computational complexity. The tactile sensors are simulated as described in Appendix B. The depth camera that is used in simulation is the Intel RealSense D435, since there are well-maintained ROS packages available for this camera in simulation (Intel, nd). The simulation environment is shown in Figure 6-1

In the simulation environment, a simulated lettuce is used as object for grasping. This object was created with a 3D mesh that was freely available (free3d, 2019). Since the mesh is quite complex in terms of geometry, the mesh is only used for visual appearance. For collisions, the lettuce is modelled as a simple cylinder. In this way, the computational burden of the simulation is reduced, while still having a realistic appearance that can be used to test the vision module of the grasp approach.

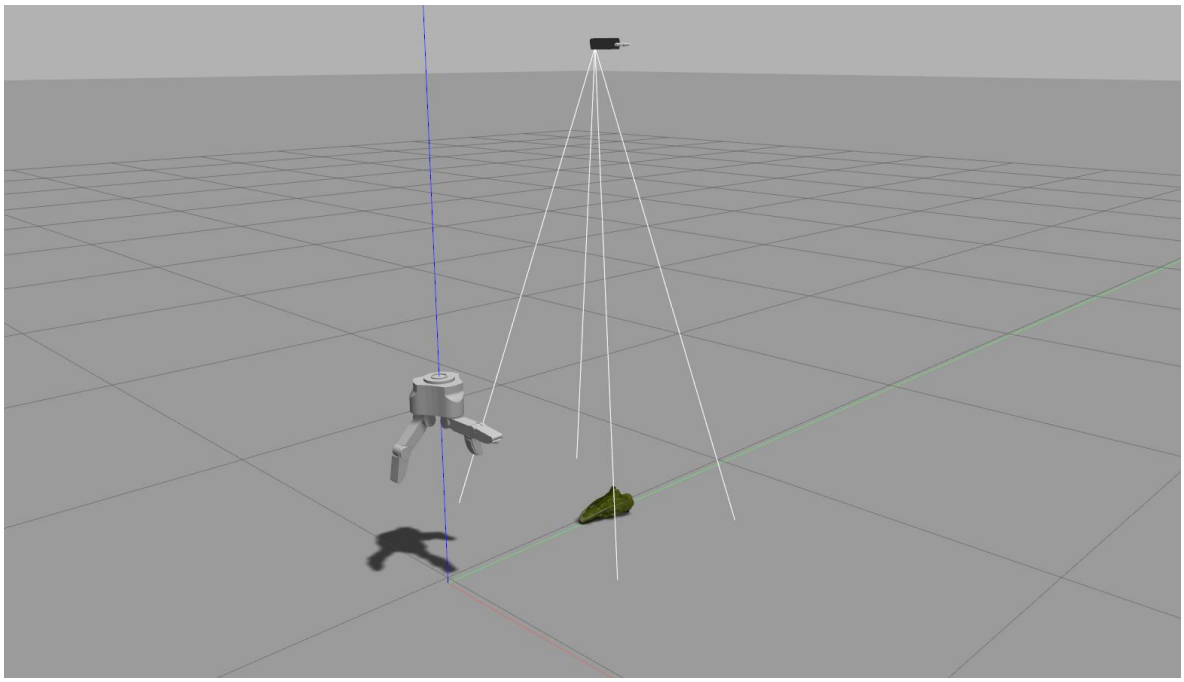
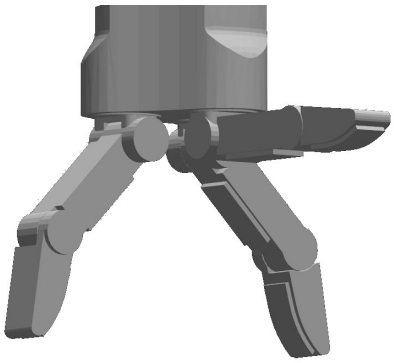
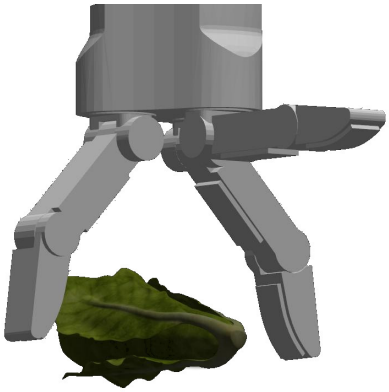


Figure 6-1: The Gazebo environment that is used for simulating grasping. The environment contains a simulated version of the SDH, an RGB-D camera and a lettuce model.

The grasp procedure goes as follows in the simulation environment. First a reference depth image is made with the RGB-D camera. Next, an object is placed in the field of view of the camera as shown in Figure 6-1. After extracting the features, gravity is turned off and the object is placed in the hand as shown in Figures 6-2a and 6-2b. This is when the closing phase of the grasp controller starts and the fingers start moving towards each other. When both fingers are in contact with the object as shown in Figure 6-2c, gravity is turned on again.



(a) Step 1: The hand is opened and gravity is turned off.



(b) Step 2: The object is placed between the finger and the hand closes.

(c) Step 3: When lifting and holding, gravity is turned on.

Figure 6-2: Images showing the steps that are involved in the Gazebo simulation.

Chapter 7

Experiment Design

The grasp approach is evaluated in two ways. The first evaluation is performed in simulation, while the second evaluation is performed on a real-world robotic setup. In this way, we can evaluate the performance of the approach for many learning episodes in the simulation environment, while also evaluating the approach with real objects.

7-1 Experiment Design in Simulation

Before evaluating the approach on the real-world setup, the approach is evaluated in simulation. The same experiment is performed ten times, in order to have a realistic image of the performance. These ten repetitions are referred to as roll-outs. Every roll-out consists of 500 episodes in the simulation environment. Each episode consists of a single grasp trial. Three sets of experiments are performed. Firstly, a set of experiments is performed using Learning of Initial Force and Tuning (LIFT) from Algorithm 2. Secondly, a set of experiments is performed in order to validate whether the two-step approach converges faster as compared to a single-step approach. This is done by doing the same experiments with $Q(\lambda)$, where the state consists of the visual features and the tactile information and the agent is allowed to take any action from the action space. Thirdly, to evaluate whether the assumptions improve the performance of the LIFT algorithm, we also perform a set of experiments with the LIFT algorithm, but instead of using the update rule from Algorithm 1, the standard Q-learning update rule is used.

During the experiments, the simulated lettuce as described in Chapter 6 is spawned into the environment very episode. However, to simulate biological variety, the lettuce is scaled every episode with a value that is drawn from a normal distribution with mean $\mu = 0.9$ and standard deviation $\sigma = 0.1$. All lengths of the lettuce are multiplied with this scale factor and the weight of the object is multiplied with this scale factor to the power of three. The grasps are considered to have failed if contact is lost with the object for longer than 0.33 seconds. On the other hand, a grasp is considered to be successful if the grasp did not fail after five seconds. Finally, the visual features that are used are l_a and w_a from Chapter 3. These are

the length and the width at the center of the object. These features are chosen, since they are expected to provide the most information on the geometry of the object. Also, taking all the features from Chapter 3 results in a larger state-space. Therefore, it is expected that using more features will only worsen the performance of the approach and increases the number of learning iterations that is required.

The control parameters that are used in the simulation experiments are shown in Table 7-1. As a short reminder, VCLOSE is the closing velocity of the hand in rad/s. This parameter is not very critical for the performance of the system. The controller gains KVOPEN and KVCLOSE are heuristically tuned. KVOPEN is set lower than KVCLOSE, since contact can be lost quickly when opening the hand. The simulated Schunk Dexterous Hand (SDH) does not support velocity-acceleration control, therefore velocity control is used instead. This is why there are no KAOPEN and KACLOSE gains specified for the simulated experiments.

Table 7-1: The grasp control parameters that are used during the experiments.

	VCLOSE	KAOPEN	KACLOSE	KVOPEN	KVCLOSE
Simulation	0.4	-	-	0.005	0.025
Real-World	0.4	0.3	0.65	0.01	0.025

The parameters for discretisation of the state-action space are shown in Table 7-2 and Table 7-3. Here can be seen that the number of possible state-action pairs is $l_{a,\mathrm{n}} \times w_{a,\mathrm{n}} \times f_{\mathrm{n}} \times \widetilde{f}_{\mathrm{n}} \times a_{\mathrm{n}} = 10 \times 10 \times 10 \times 10 \times 20 = 200,000$. Also, n_{adapt} is set to two, which means that the agent can take three distinct actions for adaptation (unless $a_{0,i} \geq a_{\mathrm{n}} - n_{\mathrm{adapt}}$).

Table 7-2: The parameters that are related to the discretisation of the action space and the visual features. Here the discretisation is performed between the minimum value ($_{\mathrm{min}}$) and the maximum value ($_{\mathrm{max}}$) in a number ($_{\mathrm{n}}$) of discrete values.

	n_{adapt}	a_{n}	a_{min}	a_{max}	$l_{a,\mathrm{min}}$	$l_{a,\mathrm{max}}$	$l_{a,\mathrm{n}}$	$w_{a,\mathrm{min}}$	$w_{a,\mathrm{max}}$	$w_{a,\mathrm{n}}$
Simulation	2	20	1.5	30.0	100.0	150.0	10	30.0	80.0	10
Real-World	1	10	1.5	15.0	210	235	3	35	45	3

Table 7-3: The parameters that are related to the discretisation of the tactile information. Here the discretisation is performed between the minimum value ($_{\mathrm{min}}$) and the maximum value ($_{\mathrm{max}}$) in a number ($_{\mathrm{n}}$) of discrete values.

	f_{min}	f_{max}	f_{n}	$\widetilde{f}_{\mathrm{min}}$	$\widetilde{f}_{\mathrm{max}}$	$\widetilde{f}_{\mathrm{n}}$
Simulation	0.0	30.0	10	0.0	30.0	10
Real-World	0.0	10.0	10	0.0	3.0	5

Finally, Table 7-4 shows the parameters that are related to the learning algorithm. Furthermore, the Q-functions $Q_0(\boldsymbol{s}_0, a_0)$ and $Q(\boldsymbol{s}, a)$ are initialised with ones. The simulation experiments are performed on a Toshiba Satellite P50-C-17F laptop that runs on Ubuntu 18.04. It has an Intel Core i7-6500U CPU, a GeForce GTX 950M GPU and 16 GiB memory.

Table 7-4: The learning parameters that are used for the experiments.

	α_0	α	c_α	ϵ_0	ϵ	c_ϵ	γ	λ	roll-outs	episodes
Simulation	0.4	0.1	0.99	0.9	0.9	0.99	0.99	0.5	10	300
Real-World	0.4	0.1	0.95	1.0	0.9	0.95	0.99	0.5	3	50

7-2 Experiment Design in Real-World

The setup for the real-world experiments is shown in Figure 7-1. Here can be seen that the setup consists of the SDH, the Asus Xtion Red Green Blue - Depth (RGB-D) camera and an aluminium frame. The procedure of the experiments is a bit different for the real-world setup, since here we cannot easily turn off gravity. Instead, the hand is opened and an object is placed on a box, such that if the hand closes, the object is pinch grasped close to its center of mass. When the lifting phase starts, the box is removed and due to gravity the object can slip out of the hand. The objects that are used for the real-world experiments are courgettes, as shown in Figure 7-2. Courgettes are used, since they have a principle length and can therefore be detected easily by the vision module. Furthermore, the sizes of courgettes vary enough to justify different grasp forces. The courgettes were bought at a Dutch supermarket. The courgettes were not chosen randomly, but are used because they have different length, width and mass properties, in order to have a more general evaluation of the grasp approach. The masses of the objects are from left to right in the images: 259 g, 318 g and 428 g. For every episode, a pseudorandom number is drawn for deciding which object is used that episode.

The grasp control parameters are shown in Table 7-1. Again, the controller is heuristically tuned and the closing gains are set higher than the opening gains, in order to prevent losing contact quickly.

On the real-world setup, learning episodes are more expensive than in simulation. Therefore, the experiments on the real-world setup consist of three roll-outs consisting of 50 episodes. Since the agent has fewer episodes to learn from, the state-action space is smaller for the experiments in the real-world. The parameters for discretisation are shown in Table 7-2 and Table 7-3. Here can be seen that there are $l_{a,\mathrm{n}} \times w_{a,\mathrm{n}} \times f_\mathrm{n} \times \widetilde{f}_\mathrm{n} \times a_\mathrm{n} = 3 \times 3 \times 10 \times 5 \times 10 = 4,500$ possible state-action pairs, which is a considerably lower number than the 200,000 from the simulation. Also, for the real-world experiments, the agent can take two distinct actions in the inner loop, instead of three. In this way the approach should converge faster, since the number of possible actions is lower.

The learning parameters are shown in Table 7-4. A fully greedy approach is chosen for the outer loop, since the agent has to learn from only 50 episodes, which does not allow too much exploration. Exploration is achieved through optimistically initial values (Sutton and Barto, 2018), i.e. $Q_0(\boldsymbol{s}_0, a_0) = 10$, for all $\boldsymbol{s}_0, a_0$. The Q-function of the inner loop $Q(\boldsymbol{s}, a)$ is initialised with ones.

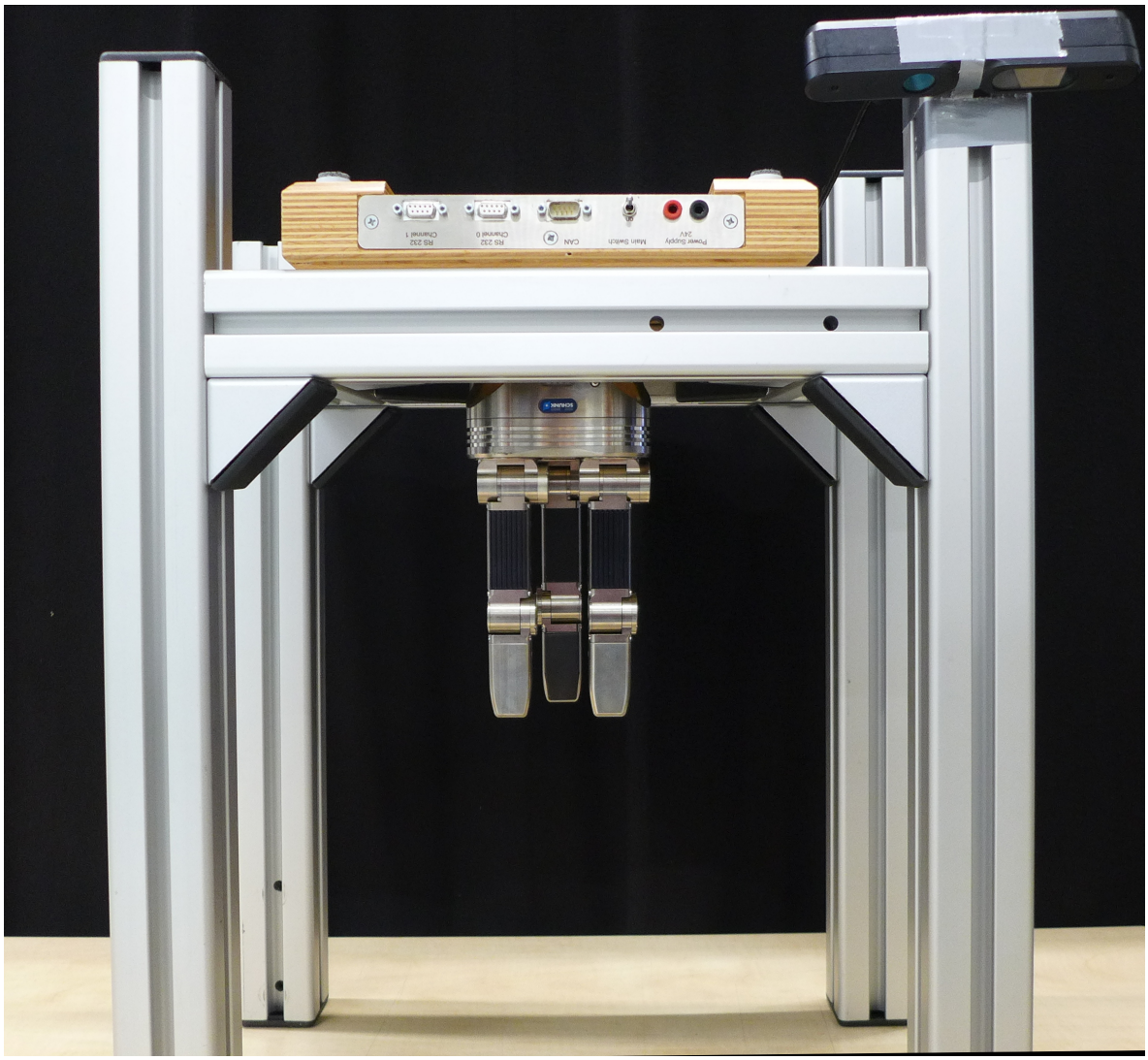


Figure 7-1: The experimental setup that is used to evaluate the grasp approach. It consists of the SDH and the Asus Xtion RGB-D camera. The robotic hand is placed on an aluminium frame. In this way, objects can be placed on a box underneath the hand and the box is removed when the lifting phase starts (when both fingers are in contact with the object).

(a) The courgettes that were used for the experiments.

(b) In order to prevent damaging the system, the courgettes were wrapped in plastic foil.

Figure 7-2: The objects that were used during the real-world experiments.

Chapter 8

Results

The results of the experiments are presented in a similar fashion for the simulated and real-world experiments. In order to provide a realistic impression of the performance of our grasp approach, all plots show the mean values and their standard deviations of the performed roll-outs. Furthermore, the results are presented in a way that allows to see if the robot improves over time. This is done by presenting the sum of the rewards for each learning episode. As a short reminder, the reward functions of the Learning of Initial Force and Tuning (LIFT) algorithm are shown below. The reward for the initial desired grasp force that is obtained by the agent at episode i is:

$$r_{0,i} = \begin{cases} -10 & \text{if failure} \\ 1 - 0.5\frac{a_{0,i}}{a_{\mathrm{n}} - 1} & \text{otherwise} \end{cases} \tag{8-1}$$

Furthermore, the reward that is obtained for adaptation of the desired grasp force at time step k is:

$$r_k = \begin{cases} -1000 & \text{if failure} \\ 1 - 0.5(\frac{a_k}{a_{\mathrm{n}} - 1} + \frac{\widetilde{f}_k}{\widetilde{f}_{\mathrm{n}} - 1}) & \text{otherwise} \end{cases} \tag{8-2}$$

We can split this reward into two parts, i.e. one that depends on a_k and one that depends on $\widetilde{f}_k$. In this way we can evaluate whether the agent is able to improve its performance in terms of minimising the grasp force and in terms of minimising the force disturbance, separately. These rewards are:

$$r_{a,k} = \begin{cases} 0 & \text{if failure} \\ 0.5(1.0 - \frac{a_k}{a_{\mathrm{n}} - 1}) & \text{otherwise} \end{cases} \tag{8-3}$$

and

$$r_{\widetilde{f},k} = \begin{cases} 0 & \text{if failure} \\ 0.5(1.0 - \frac{\widetilde{f}_k}{\widetilde{f}_{\mathrm{n}} - 1}) & \text{otherwise} \end{cases} \tag{8-4}$$

We define the return as the sum of the rewards and we can calculate the returns for $r_k, r_{a,k}$ and $r_{\widetilde{f},k}$:

$$J = \sum^{k} r_k \tag{8-5}$$

$$J_{\text{force}} = \sum^{k} r_{a,k} \tag{8-6}$$

$$J_{\text{disturbance}} = \sum^{k} r_{\widetilde{f},k} \tag{8-7}$$

These returns are used as performs metrics of the grasp approach.

8-1 Results from Experiments in Simulation

There are three sets of experiments performed in simulation, i.e. with LIFT, Q(λ) and LIFT without assumptions. The results from the experiments with Q(λ) are compared to those from LIFT, in order to evaluate whether the two-step approach leads to improved performance. The comparison with LIFT without assumptions is presented in order to evaluate the influence on the performance of the update rule from Algorithm 1 as compared to the standard Q-learning update rule.

8-1-1 Comparison between LIFT and Q(λ)

In Figures 8-1 and 8-2 the results from the experiments with LIFT and Q(λ) are shown. In Figure 8-1a, the rewards for the initial desired grasp force are shown. Here we can see that the mean reward improves over the episodes for LIFT, while this is not the case for Q(λ). The plot in Figure 8-1b shows similar results, where the return for the desired grasp force adaptation is improving for LIFT and not for Q(λ). If we look at the part of this return that depends on the desired grasp force, J_{force} in Figure 8-1c, we see that both algorithms perform equally well in terms of this metric. The mean for $J_{\text{disturbance}}$ is higher with LIFT over the entire course of episodes, as shown in Figure 8-1d. Therefore, it is not surprising that this is also the case for $J_{\text{force}} + J_{\text{disturbance}}$, as can be seen in Figure 8-2a. In Figure 8-2b the success rates of the two algorithms are plotted over the episodes. The success rate is calculated by taking the moving average over 30 episodes. This is why the plot in Figure 8-2b starts at episode 30. The success rate clearly improves over the episodes for LIFT, while this is not the case for Q(λ). The success rate of LIFT is $67.7 \pm 8.6\%$ over the first 30 consecutive episodes, while this is $43.3 \pm 8.6\%$ with Q(λ). The success rate of LIFT increases up to $96.3 \pm 2.3\%$ at episodes 240 up to 246. With Q(λ), a maximum success rate of $52.3 \pm 18.1\%$ is obtained after 265 episodes.

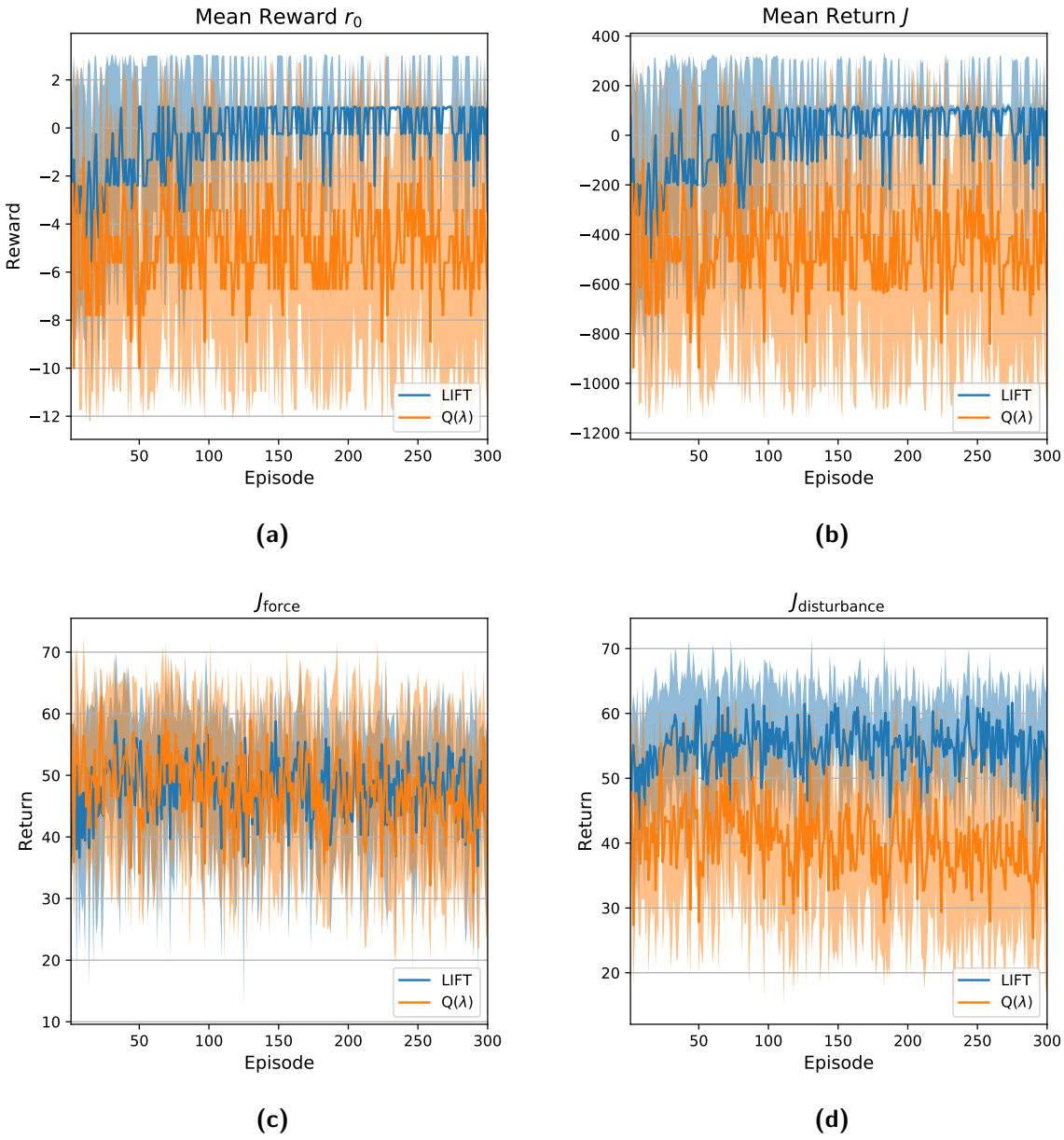
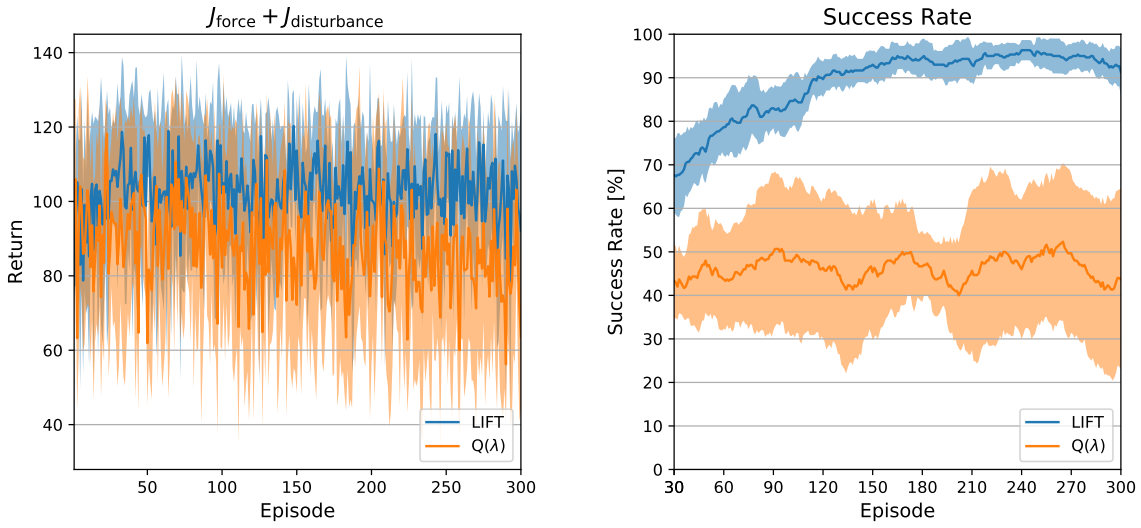


Figure 8-1: Plots showing the results of the experiments in simulation. Each plot shows the mean and standard deviation of the 10 roll-outs. Plot **(a)** shows the rewards for the initial desired grasp force. Plot **(b)** shows the return J for adaptation of the desired grasp force. Plot **(c)** shows the contribution of the return J that depends on a_k, while plot **(d)** shows the contribution of the return J that depends on $\widetilde{f}_k$.



(a) **(b)**

Figure 8-2: Plots showing the results of the experiments in simulation. Figure **(a)** shows the sum of the returns J_{force} and $J_{\text{disturbance}}$. Figure **(b)** shows the success rate per episode. This success rate is calculated as the moving average over 30 consecutive episodes.

8-1-2 Comparison between LIFT with and without Assumptions

The results of LIFT without assumptions are shown in Figures 8-3 and 8-4. In these plots, the results of LIFT are shown again, which makes comparison of the two easy. From these figures becomes clear that without the assumptions, the mean reward r_0 is lower for the first 50 episodes. This is also the case for J, the return for adaptation of the desired grasp force. Furthermore, we can see that without the assumptions, the returns J_{force} and $J_{\text{disturbance}}$ are lower. Finally, we can see that the assumptions also lead to better performance in terms of success rate. Especially during the first 150 episodes, the success rate with assumptions is significantly higher. During the last episodes, the two algorithms seem to come closer together in terms of success rate. Without assumptions, a success rate of $35.3 \pm 11.8\%$ is obtained after the first 30 episodes. A maximum success rate of $89.3 \pm 5.3\%$ is obtained without the assumptions after 273 episodes.

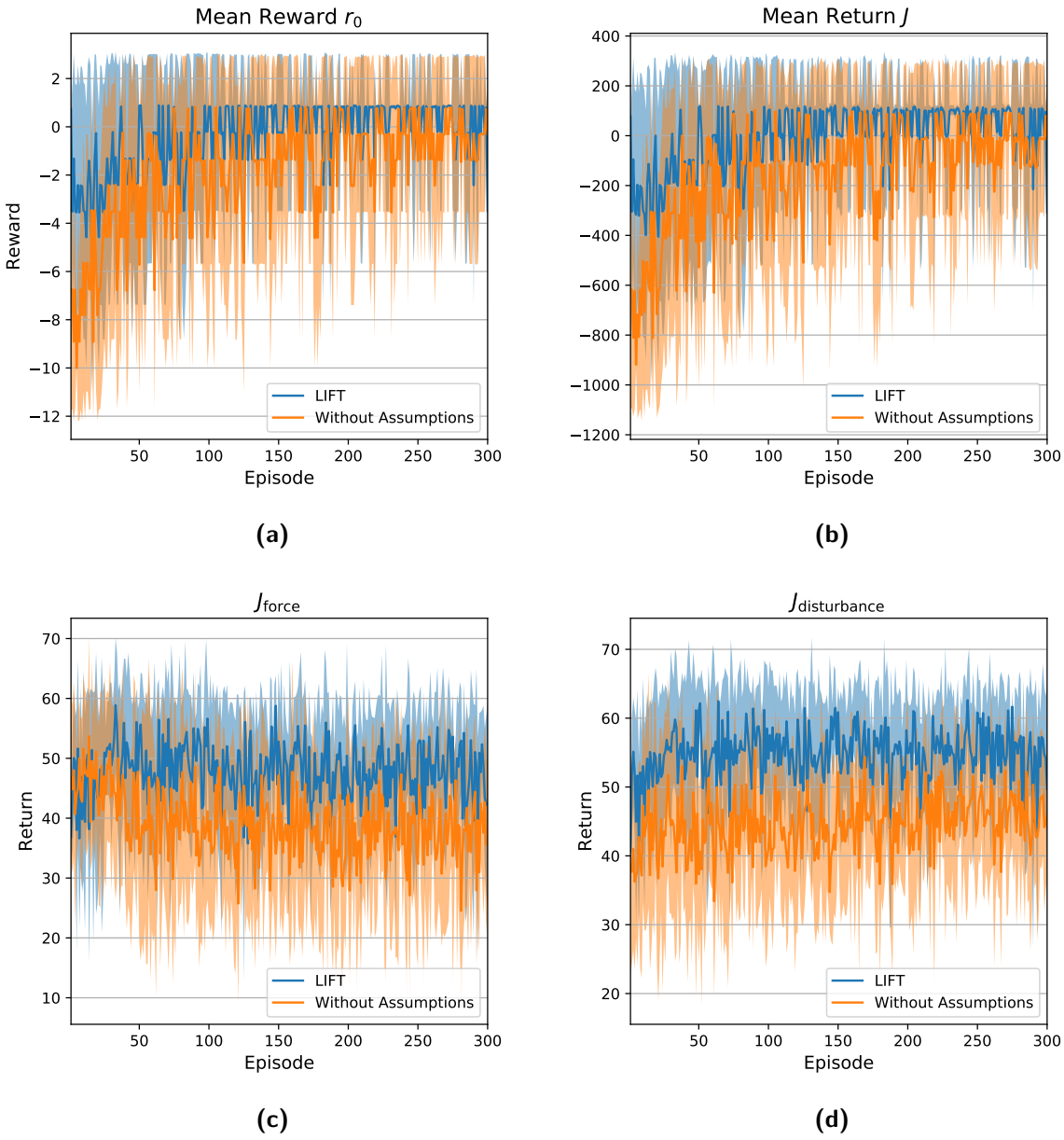
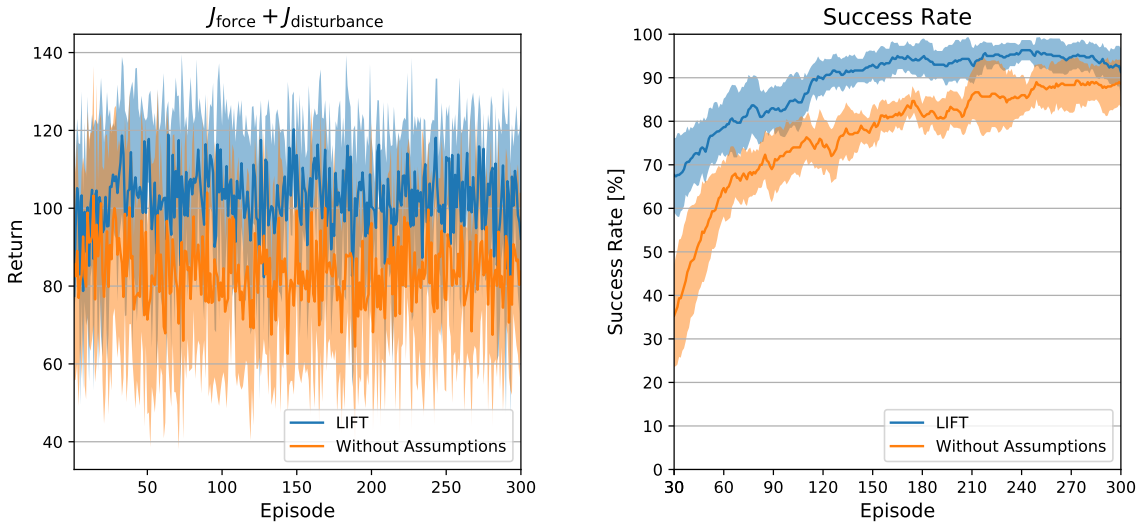


Figure 8-3: Plots showing the results of the experiments in simulation. Each plot shows the mean and standard deviation of the 10 roll-outs. Plot **(a)** shows the rewards for the initial desired grasp force. Plot **(b)** shows the return J for adaptation of the desired grasp force. Plot **(c)** shows the contribution of the return J that depends on a_k, while plot **(d)** shows the contribution of the return J that depends on $\widetilde{f}_k$.



(a) **(b)**

Figure 8-4: Plots showing the results of the experiments in simulation. Figure **(a)** shows the sum of the returns J_{force} and $J_{\text{disturbance}}$. Figure **(b)** shows the success rate per episode. This success rate is calculated as the moving average over 30 consecutive episodes.

8-2 Results from Real-World Experiments

For the real-world experiments, the same plots are made and shown in Figure 8-5 and Figure 8-6. The real-world experiments consisted of three roll-outs of 50 episodes. Again, the mean and standard deviation over the three roll-outs are shown in the plots. In Figure 8-5b the mean reward for the initial desired grasp force is plotted over the episodes. Here we can see that it seems to improve over the episodes. Again, the plot for J (Figure 8-5b) is very similar to the one for r_0. This was also the case in simulation. The mean return J_{force} is shown in Figure 8-5c. This plot shows capricious behaviour in the beginning and becomes steady after approximately half of the episodes. This is also the case for the mean return $J_{\text{disturbance}}$. Therefore, it is not a surprise that the sum of these parts shows the same behaviour, as can be seen in Figure 8-6a. The success rate is plotted in Figure 8-6b. Again, the success rate is calculated as the mean of a moving average over a number of episodes, in this case ten episodes. In this way, the success rate is calculated over thirty grasp trials. The plot shows that the success rate first decreases to its minimum in episodes 11 up to 13 ($76.7 \pm 9.4\%$). Calculating the success rate over the last 10 episodes results in a success rate of $96.7 \pm 4.7\%$ on the real-world setup.

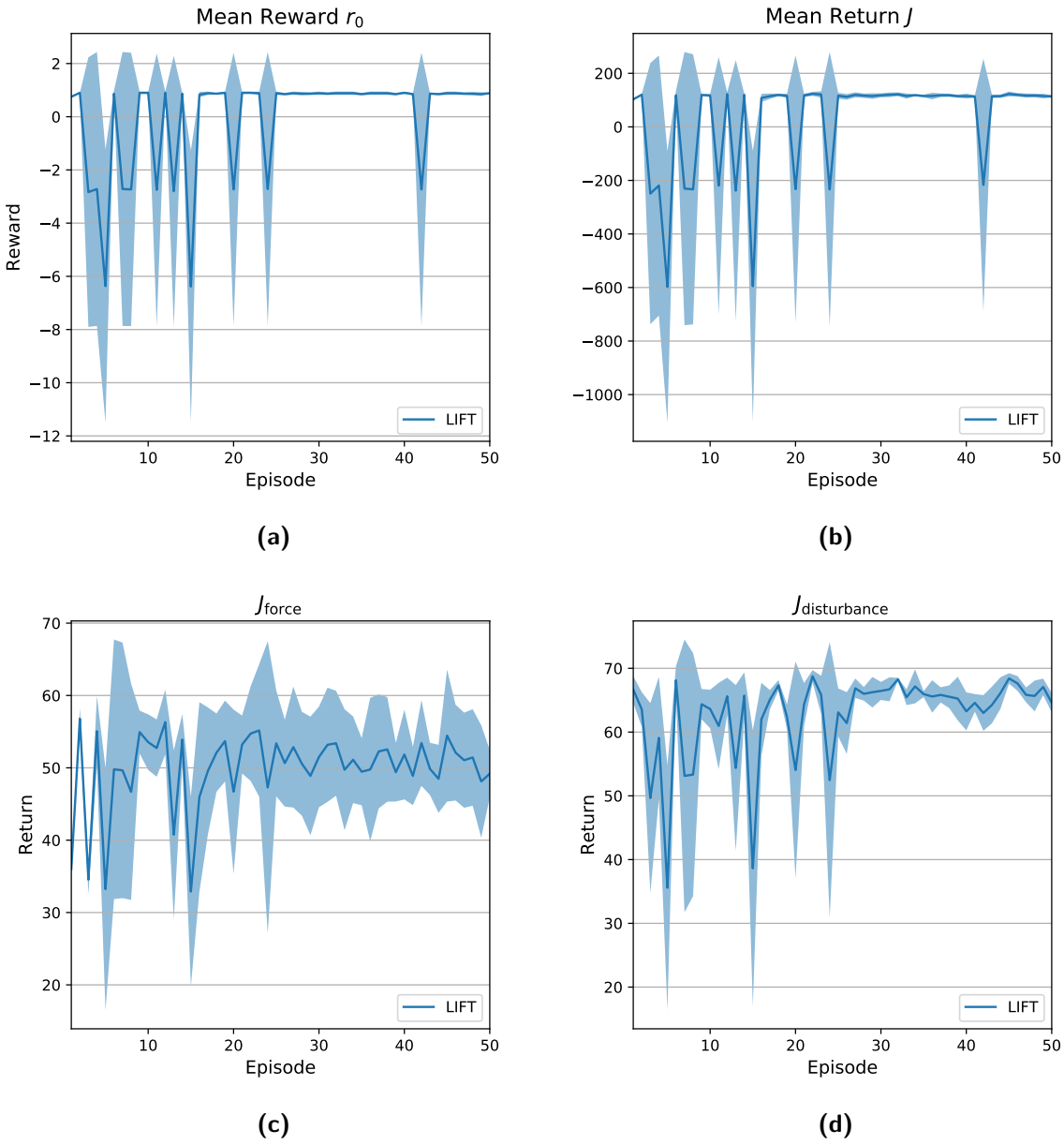
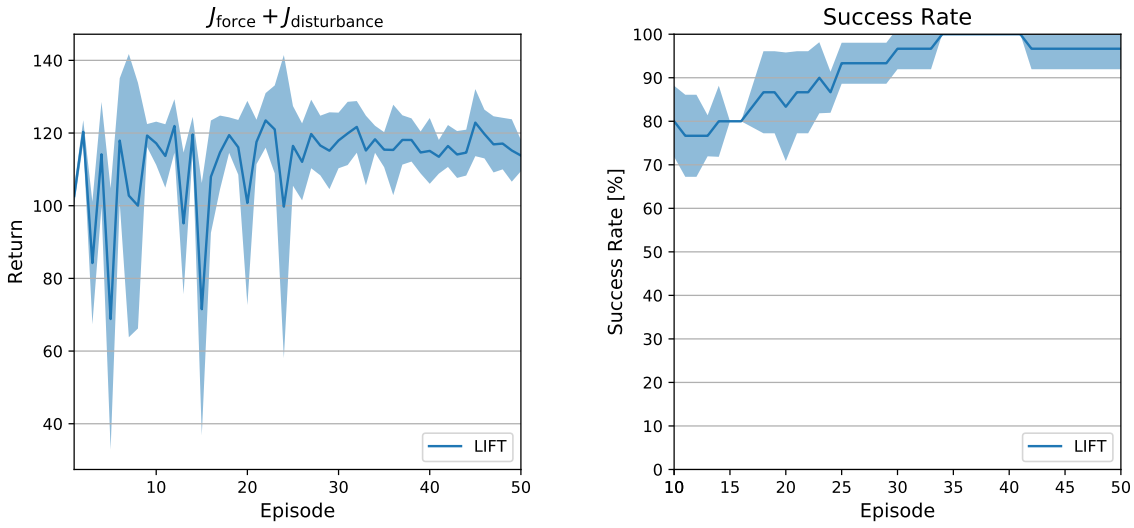


Figure 8-5: Plots showing the results of the experiments on the real-world setup. Each plot shows the mean and standard deviation of the 3 roll-outs. Plot **(a)** shows the rewards for the initial desired grasp force. Plot **(b)** shows the return J for adaptation of the desired grasp force. Plot **(c)** shows the contribution of the return J that depends on a_k, while plot **(d)** shows the contribution of the return J that depends on $\widetilde{f}_k$.



(a) **(b)**

Figure 8-6: Plots showing the results of the experiments on the real-world setup. Figure **(a)** shows the sum of the returns J_{force} and $J_{\text{disturbance}}$. Figure **(b)** shows the success rate per episode. This success rate is calculated as the moving average over 10 consecutive episodes.

Furthermore – in contrast to the simulation – the vision module does not have perfect depth information to its disposal during the real-world experiments. Therefore, the mean and standard deviation of the feature values that were obtained by the vision module are shown in Table 8-1, next to their true values. In this table we can see that all values obtained by the vision module are lower than their true values. The mean and standard deviation of the discretised versions of these features are also shown, since these discrete values were used for learning. In Table 8-2, for each possible feature vector the percentages of the times that the object was described by that discrete feature vector are shown. Here we can see that the feature vector $\boldsymbol{s}_0 = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$ was $\begin{bmatrix} 0 & 0 \end{bmatrix}$ for courgette 1 in all cases. However, for the other two objects, the feature vector that was obtained from the vision module was not always the same. This shows that the vision module was not able to describe the same object with the same feature set every time. Also, the same feature vector is sometimes obtained from different objects.

Table 8-1: Characteristics and features of the objects that were used during the experiments. The true feature values are shown ("Ground truth") as well as the mean and standard deviation of the values obtained by the camera system during the experiments ("Obtained by vision module"). The mean and standard deviation of the discretised feature values are shown ("Discrete features"). Here x_1 is obtained after discretising l_a and x_2 is obtained after discretising w_a.

	Ground truth		Obtained by vision module		Discrete features	
	l_a	w_a	l_a	w_a	x_1	x_2
Courgette 1	21.5 cm	4.5 cm	20.9 ± 0.3	3.4 ± 0.3 cm	0.00 ± 0.00	0.00 ± 0.00
Courgette 2	23.5 cm	5.0 cm	22.6 ± 0.7	3.8 ± 0.4 cm	0.70 ± 0.66	0.19 ± 0.39
Courgette 3	25.0 cm	5.5 cm	23.5 ± 0.6	4.5 ± 0.3 cm	1.49 ± 0.55	1.61 ± 0.62

Table 8-2: Table showing the percentages of the cases that the discrete feature vector was obtained for the object in [%].

$\boldsymbol{s}_0$	(0,0)	(0,1)	(0,2)	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(2,2)
Courgette 1	100.0	0	0	0	0	0	0	0	0
Courgette 2	37.0	3.7	0	35.2	13.0	0	9.3	1.9	0
Courgette 3	0	0	2.4	2.4	4.9	39.0	4.9	19.5	26.8

Chapter 9

Conclusion and Discussion

In this work, a novel approach for grasping deformable food objects is presented and evaluated. It is a human-inspired grasp approach with learning capabilities. The grasp approach involves a novel reinforcement learning algorithm: Learning of Initial Force and Tuning (LIFT). This algorithm was designed to grasp objects with minimal force in order to prevent that they are damaged during grasping. At the same time, this algorithm was designed to maximise grasp success and minimise slipping effects. Imitated versions of human tactile afferents were used for the state description in combination with visual features. These visual features are obtained with an Red Green Blue - Depth (RGB-D) camera.

9-1 Conclusions

We developed a method for grasping deformable food objects with minimal force. From the experiments can be concluded that the approach results in successful grasping in both simulation and on the real-world setup. As shown in Figures 8-2 and 8-4, the success rate clearly improves over the number of episodes in simulation. A success rate of 96.3 % is obtained in simulation, when calculating the success rate as the moving average of 30 consecutive episodes. On the real-world setup, the success rate improves in a similar fashion, as shown in Figure 8-6b. For the real-world experiments a success rate of 96.7 % is obtained over the last ten episodes.

It is hard to conclude whether the approach is successful in grasping with minimal force, since a minimal force is hard to obtain. This is exactly why the approach is designed in the first place. We can see however, that the return J_{force} does not decrease over the episodes, as shown in Figures 8-1c, 8-3c and 8-5c. This return is higher when lower desired grasp forces are used. It is not expected that this return increases over the episodes together with the success rate. Since in that case, the objects are grasped with lower forces, while also having higher success rates. So, the fact that this part of the return does not decrease too much, while having a success rate that increases, indicates that the approach results in grasping with forces that are nearly minimal.

Next to minimising the force and maximising the success rate, the third goal of the grasp approach is to minimise slipping. In this work, the imitated Fast-Adapting Type I (FA-I) signal is used as indicator for slipping, i.e. the force disturbance $\widetilde{f}_k$. In Figures 8-1d and 8-3d, we can see that $J_{\text{disturbance}}$ does not increase over the episodes. So, in simulation the approach is not successful in reducing the force disturbance. In Figure 8-5c, it seems that $J_{\text{disturbance}}$ stabilises after approximately 30 episodes. However, we cannot conclude whether this is the result of successful adaptation of the desired grasp force, or the result of choosing the right initial desired grasp force.

In the algorithm LIFT, the desired grasp force is initially chosen with a smaller state space and later this force is adapted, while having a larger state space and a smaller action space. We conclude that this two-step approach leads to faster convergence as compared to LIFT without this two-step structure. This conclusion is based on the results from Figures 8-1 and 8-2, which shows that LIFT has superior returns and success rates as compared to Q(λ). Here LIFT is compared to Q(λ), since the algorithm becomes a version of the $Q(\lambda)$ algorithm when omitting the initial step. However, we do not claim that the LIFT algorithm is superior to the Q(λ) algorithm. We only conclude that for this specific problem, with the state consisting of the visual features and the tactile information, that the two-step approach leads to faster convergence.

Finally, we also conclude that incorporating the assumptions on the grasp problem from Chapter 5 in the LIFT algorithm leads to faster convergence in the simulation experiments. These assumptions are incorporated in the LIFT algorithm by using the update rule from Algorithm 1. In Figures 8-3 and 8-4, we can see that the performance of the LIFT algorithm is better when using this update rule, instead of the standard Q-learning update rule. However, the objects in the simulation are not completely realistic, since they are scaled versions of each other. In a real-world setting, there is a much larger variety in object properties. Therefore, we cannot conclude whether the assumptions also lead to improved performance in the real world.

9-2 Applications

The method we propose can be applied in situations where a single type of food object has to grasped. An example where this is the case, is the packaging of bell peppers. Presently, humans put bell peppers into place before they are put into packages of three. A conveyor belt system brings in the bell peppers that are already sorted on color. Humans are needed to place a bell pepper from each colour in line, so that they can be packed. Also, when the bell peppers are packed, the packages have to be put into boxes, which is currently also not automated. For these tasks, the approach that is proposed in this work can provide a solution for the grasping part of these pick-and-place tasks.

9-3 Strengths and Limitations

First of all, a strength of this study is that it introduces the first reinforcement learning approach for grasping deformable food objects with minimal force. A novel reinforcement learning algorithm (LIFT) is introduced for learning grasp forces. Prior knowledge on the

grasp problem is incorporated in the form of a decomposed learning strategy with an inner and outer loop, where the outer loop takes actions based on visual features, while the inner loop adapts this force based on the visual features in combination with tactile information. Prior knowledge is also incorporated in the form of a modified update rule in the outer loop that is based on assumptions on the grasp problem.

Secondly, the method that is proposed is model-free and therefore does not require that a model is made of the objects before grasping. Some model-based methods require computationally expensive Finite Element Method (FEM) models for grasping (Zaidi et al., 2017; Petit et al., 2017). This is not the case for the method that is proposed in this work and therefore it is possible to grasp novel objects quickly. Thirdly, the grasp approach does not require extreme amounts of learning interactions. Whereas some works require tens of thousands learning interactions (Calandra et al., 2017, 2018), our approach is able to converge after tens or hundreds of episodes – depending on the number of features that are used and discretization steps.

Also, this work requires fewer learning interactions than the 525 demonstrations from Misimi et al. (2018). It should be noted however, that the works that are referred to also learn grasp postures and are evaluated on pick-and-place tasks, so it is not possible to compare this work one-to-one with these other studies. Nonetheless, we believe that for many objects it is not necessary to learn grasp postures or solve the motion planning problem with a learning method. In that case, learning pick-and-place tasks with our approach would not require more learning iterations than learning the grasp force.

Finally, another strength of this study is that the evaluation is performed in simulation and on a real-world setup. This allowed us to evaluate the method for more episodes than would be possible on the real-world setup. In this way it was possible to show that the approach is effective with a relatively large state-action space. On the other hand, the experiments on the real-world setup show that the method is successful with real food objects.

There are several limitations to the experimental design of this study. Firstly, the approach is evaluated in experiments in which the end-effector is not moving. In real-world manipulation tasks, the end-effector moves from one place to another while grasping. These movements result in accelerations on the object and therefore influence the grasp forces. These accelerations are not taken into account in our experimental setup.

Furthermore, in practice, the end-effector moves towards the object, while in the experiments the object is moved towards the end-effector. Therefore, it is not possible to conclude from our experiments whether extracting the grasp location from the depth images will result in successful grasps. Also, during the experiments objects rotated inside the hand. These grasps were considered to be successful by the robot, since contact was not lost. However, in practice, this behaviour is undesirable since it is impossible to place the object at its goal position with high precision if the object is moving in the hand. Therefore, if the grasp approach is used for a pick-and-place task, the success of the grasp should depend on whether the object is placed correctly at the goal position.

A limitation of the experiments in simulation is that simulations always contain imperfections. In the case of grasping deformable objects, these imperfections include simulating deformations and contact forces. It is possible that the agent exploits the imperfections in simulation.

Next to limitations to the experiments, there are also limitations to the approach. First of all, the depth information that is provided by the camera module is imperfect. As shown in Table 8-2, sometimes the same feature set was used to describe different objects. If the vision module is not able to discriminate objects in terms of feature values, the visual features no longer provide useful information. Therefore, depth information with higher precision is required in order to make the approach truly effective. Perhaps this problem can be solved by using another depth camera, or by using multiple depth cameras. Another limitation of the vision module is that it has no object recognition capacities. Therefore, the grasp approach is limited such that it can only be used for a single type of object. Finally, another limitation is that the Markov property is violated, since the state transitions are also influenced by past states and actions. For example, when an object slips and ends up on the edge of the finger tips or is rotated, the state does not contain this information, while the probability of failure is higher in that case. If the state representation consists of the pressure values at each texel location for a number of consecutive time steps, the Markov property would be satisfied to a higher degree. However, this would increase the dimensionality of the state-space significantly and therefore increase the number of required learning interactions drastically.

9-4 Future Work

We propose the following directions for future work. First of all, we propose to evaluate the grasp approach on pick-and-place experiments. For these experiments, the setup would consist of a robotic manipulator, a robotic hand with tactile sensors and an RGB-D camera. The task would be to detect an object, grasp it and place it at a goal position. In this way, the approach would be evaluated in a manner that better resembles a practical use case.

Secondly, we propose to extend the vision module with object recognition capacities, in order to allow grasping of different object types. In this way, different Q-functions can be learned for different types of objects. Image recognition could for example be realised by the You Only Look Once (YOLO) algorithm (Redmon et al., 2016; Simony et al., 2018; Tian et al., 2019). The vision module can also be improved by having a more sophisticated method for object orientation detection, since our approach is only viable for oblong objects. A procedure based on frustum pointnets by Qi et al. (2018) is a possible candidate for improving the object (orientation) detection from RGB-D data.

The control module can be improved by having a control structure with direct torque control for tracking of the desired grasp force, instead of having a low-level Proportional Integral Derivative (PID) controller that tracks desired joint velocities and accelerations. This will simplify the control structure and allows for control with higher sampling frequencies. Also, we propose to study adaptive grasp control gains, since we assume that more accurate force tracking will be realised if the grasp control gains are object dependent.

As a direction for future work related to the learning module, we propose to study a black-box optimization strategy for choosing the initial desired grasp force. Choosing the initial desired grasp force is based on a static state that consists of visual features. Therefore, we suspect that a gradient-based optimization strategy, such as Covariance Matrix Adaptation (CMA) (Hansen, 2006) might result in better performance as compared to the Q-learning strategy that is used in the LIFT algorithm.

Appendix A

Background of the Grasp Controller

The sense of touch is vital for grasping objects dexterously. You might have already come to realise this on a cold winter's day when trying to grab your keys to open your front door. The cold reduces the functioning of the somatosensory system and all of a sudden this simple task becomes difficult. This phenomenon was also studied by the scientific community. Experiments showed that anaesthetising the hand (in order to disrupt the tactile information) leads to a considerable drop in the stability of resulting grasps (Westling and Johansson, 1984). As described by Dahiya et al. (2009), the sense of touch can be divided into two modalities based on the origin of the sensory receptors. Firstly, there is the cutaneous sense which comprises the sensory input that comes from receptors in human skin. Secondly, the kinaesthetic sense encompasses the sensory information that comes from receptors that can be found in muscles, tendons and joints. The cutaneous system provides sensory information that is acquired through physical contact with the skin. The kinaesthetic system on the other hand is responsible for proprioceptive information (relative orientation and position of the body parts). This information results from combining afferent (incoming) signals with efference copy, which is a copy of efferent (outgoing) signals that can be used together with an internal model to compare the desired movements to the actual movements resulting from the signal. Next to the cutaneous and kinaesthetic modalities, the notion of a haptic sensory system is also a widely used concept, which is a combination of sensory information from both the kinaesthetic and the cutaneous system (Loomis and Lederman, 1986; Klatzky and Lederman, 2003).

A-1 Tactile Afferents

The cutaneous sensory system is based on tactile information that comes from receptors in the human skin. For grasping, the cutaneous sensory information from the hand provides the most important tactile input. Inside the skin of the human hand there are four different kinds of tactile afferents, where each has its own functions and characteristics. These afferents are Slow-Adapting type I (SA-I), Slow-Adapting type II (SA-II), Fast-Adapting Type I (FA-I)

and Fast-Adapting Type II (FA-II). The difference between fast-adapting and slow-adapting afferents is that fast-adapting afferents send bursts of action potentials when the stimulus changes, while slow-adapting afferents are also active when a stimulus remains constant. The SA-I and FA-I afferents have their nerve endings superficially in the skin. The SA-I afferents terminate in the Merkel cells, while the FA-I afferents end in the Meissner's corpuscles. The SA-I afferents are responsible for detection of low frequency skin deformations, while the FA-II afferents are excited by high frequency deformations. The SA-II and FA-II afferents terminate deeper in the skin and their nerve endings are called the Ruffini and Pacinian corpuscles, respectively. The SA-II afferents are excited by lateral stretching of the skin (Johansson, 1978; Knibestöl, 1975) and tangential shear strain that results from manipulating objects (Westling and Johansson, 1987). On the other hand, the FA-II afferents are sensitive to transient mechanical events, such as the making and breaking of contact with objects (Westling and Johansson, 1987). An overview of the characteristics of the four different types of afferents is shown in Figure A-1.

A-2 Action-Phases in Human Grasping

Johansson and Flanagan (2009) identified that there are distinct phases during a pick-and-place task, i.e. lifting an object from a surface and replacing it after some time. These distinct phases are separated by contact events, such as making and breaking of contact. The action-phases that are identified by Johansson and Flanagan (2009) are: reaching, loading, lifting, holding, replacing and unloading. According to Johansson and Flanagan (2009) each action-phase has its own action-phase controller that makes predictions of sensory information during that phase. In this way, the human is able to make corrective actions in case of mismatches between the predicted sensory signal and the actual sensory signal. Each phase is characterised by specific afferent *"signatures"*, which are neural afferent signals that indicate a transition from one action-phase to another. Examples of these signatures are bursts of SA-I and FA-I afferent neurons when contact is made with an object and when contact is lost. Also, when lifting an object from a surface, this mechanical transient event activates FA-II afferent neurons.

Inspired by these findings from research in neurology, Romano et al. (2011) proposed a human-inspired grasp controller. A flow diagram of this controller is shown in Figure A-2. In this flow diagram we can recognise the action-phases from Johansson and Flanagan (2009).

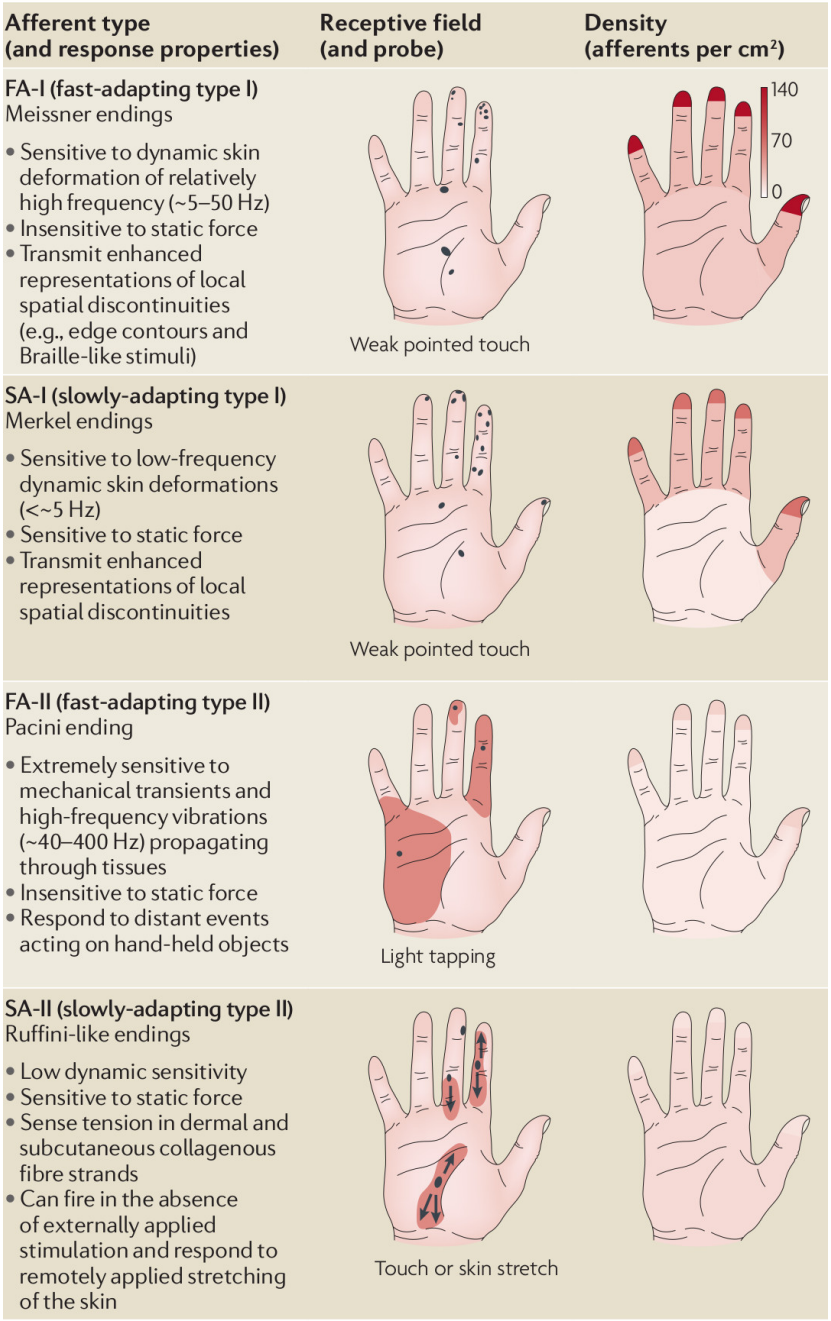


Figure A-1: A summary of the properties of the tactile afferent signals, their receptive field and density. Image adopted from Johansson and Flanagan (2009).

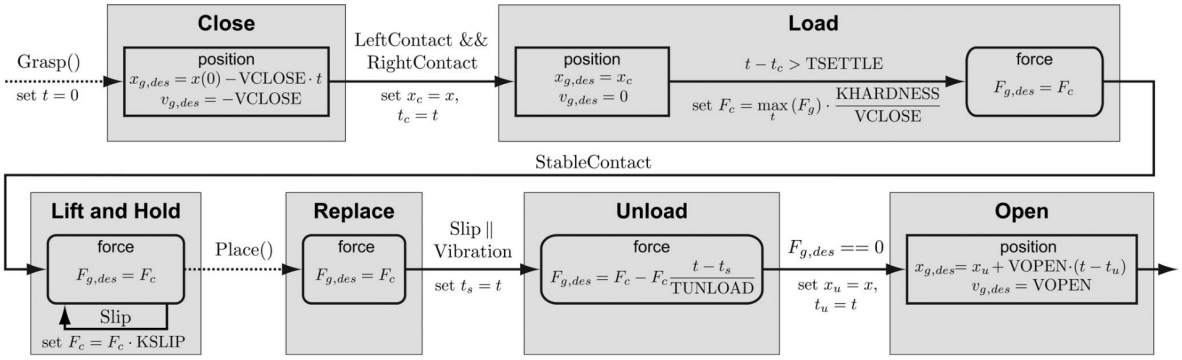
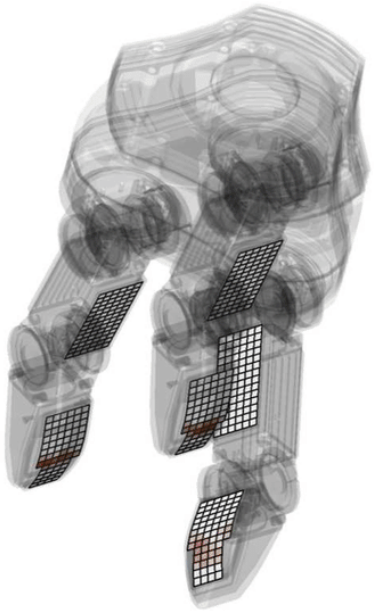


Figure A-2: Flow diagram from Romano et al. (2011). This control scheme is based on the work by Johansson and Flanagan (2009). The capitalised notions refer to control parameters. Furthermore, t refers to time in seconds, x refers to the end-effector's position, $x_{g,des}$ is the desired joint position, $v_{g,des}$ is the desired joint velocity, F_g is the mean finger tip force and $F_{g,des}$ is the desired mean finger tip force.

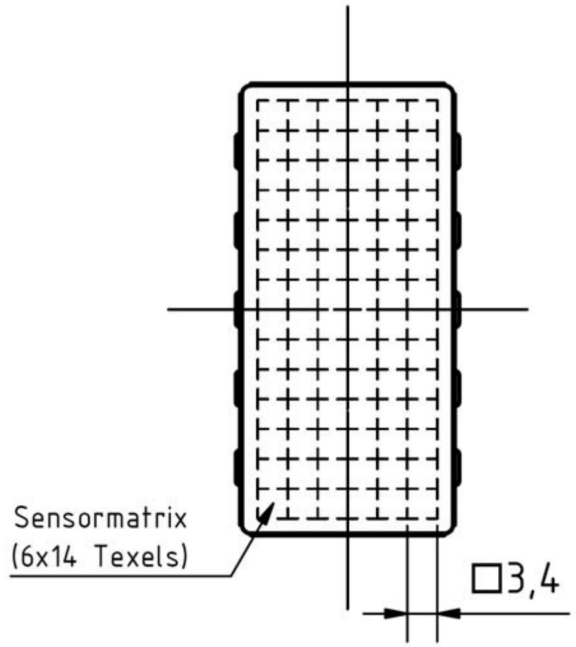
Appendix B

Simulated Tactile Sensors

Since the grasp controller is tactile based, the simulation should also incorporate tactile information. For simulating the tactile sensors, a Robot Operating System (ROS) package was used that was provided by Fraunhofer IPA (nd). This package was modified by the author, since it was not functioning out-of-the-box, contained inefficient code and did not send the same type of messages as the real tactile sensors. The tactile sensors of the Schunk Dexterous Hand (SDH) robotic hand are simulated as follows. Each link of the fingers contains a bumper sensor in the Gazebo simulation. This bumper sensor sends a *ContactState* message every time the corresponding link makes contact with another object. This message contains the wrenches and locations of the contacts at that moment. First we have to check which contact points are actually located at a tactile sensor, since the real robotic finger has a tactile sensor at one side of each finger link. To do this, the contact positions from the *ContactState* message are evaluated. These positions are given in the global world frame of the simulation environment, so these have to be transformed to the local frame of the corresponding finger link in order to check on which side of the link the contact takes place. If the contact takes place at the side of the tactile sensor, the contact location is translated to the corresponding texel location in the sensor matrix, since the real sensors consist of 6×14 texels (see Figure B-1). Next, the force at that tactile location is defined as the force component that is perpendicular to the tactile sensor. The force values at each texel location are stored in the sensor matrix. Since the contact locations are points in the Gazebo simulation, a moving average is applied to the sensor matrix to simulate contact surfaces. Also, a time average is run over sequential sensor matrices, since contacts can be quite capricious in Gazebo. In our simulations the sampling rate of the simulated tactile sensors was 50 Hz and the time average was calculated over 5 consecutive sensor matrices.



(a) Image from Krug et al. (2016) showing the tactile pads on the SDH robotic hand.



(b) Image from the product sheet of the DSA 9205 tactile sensors (Weiss Robotics, 2008).

Figure B-1: The SDH robotic hand is equipped with 6 DSA 9205 tactile sensors, i.e. 2 on each finger. These sensors consist of 6×14 texels for the pads on the proximal phalanges and 6×13 on the distal phalanges. The output of the sensor is a matrix containing a voltage value for each texel. A linear relation is used to convert these voltages to pressure values.

Bibliography

Alderman, L., Eddy, M., and Tsang, A. (2020). Migrant Farmworkers Whose Harvests Feed Europe Are Blocked at Borders. *The New York Times*.

Arriola-Rios, V. E. and Wyatt, J. L. (2017). A multimodal model of object deformation under robotic pushing. *IEEE Transactions on Cognitive and Developmental Systems*, 9(2):153–169.

Asus (n.d.). Xtion PRO LIVE. https://www.asus.com/us/3D-Sensor/Xtion_PRO_LIVE/overview/. Accessed: 15-07-2020.

Bicchi, A. and Kumar, V. (2000). Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE.

Bohlin, R. and Kavraki, L. E. (2000). Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Caccamo, S., Güler, P., Kjellström, H., and Kragic, D. (2016). Active perception and modeling of deformable surfaces using Gaussian processes and position-based dynamics. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 530–537. IEEE.

Calandra, R., Owens, A., Jayaraman, D., Lin, J., Yuan, W., Malik, J., Adelson, E. H., and Levine, S. (2018). More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307.

Calandra, R., Owens, A., Upadhyaya, M., Yuan, W., Lin, J., Adelson, E. H., and Levine, S. (2017). The feeling of success: Does touch sensing help predict grasp outcomes? *arXiv preprint arXiv:1710.05512*.

Chitta, S., Sucan, I., and Cousins, S. (2012). Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19.

Cretu, A.-M., Payeur, P., and Petriu, E. M. (2011). Soft object deformation monitoring and learning for model-based robotic hand manipulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):740–753.

Dahiya, R. S., Metta, G., Valle, M., and Sandini, G. (2009). Tactile sensing - from humans to humanoids. *IEEE transactions on robotics*, 26(1):1–20.

Delgado, A., Corrales, J. A., Mezouar, Y., Lequievre, L., Jara, C., and Torres, F. (2017a). Tactile control based on Gaussian images and its application in bi-manual manipulation of deformable objects. *Robotics and Autonomous systems*, 94:148–161.

Delgado, A., Jara, C. A., and Torres, F. (2017b). Adaptive tactile control for in-hand manipulation tasks of deformable objects. *The International Journal of Advanced Manufacturing Technology*, 91(9-12):4127–4140.

Dickerson, C. and Jordan, M. (2020). South Dakota Meat Plant Is Now Country's Biggest Coronavirus Hot Spot. *The New York Times*.

Flanagan, J. R. and Johansson, R. S. (2003). Action plans used in action observation. *Nature*, 424(6950):769–771.

Fraunhofer IPA (n.d.). schunk_modular_robotics. https://github.com/ipa320/schunk_modular_robotics. Accessed: 15-07-2020.

free3d (2019). Lettuce Romaine 3D Model. https://free3d.com/3d-model/lettuceromaine-v1--988064.html. Accessed: 15-07-2020.

Furukawa, S., Xu, L., and Middlebrooks, J. C. (2000). Coding of sound-source location by ensembles of cortical neurons. *Journal of Neuroscience*, 20(3):1216–1228.

Gil, P., Mateo, C. M., Delgado, A., and Torres, F. (2016). Visual/Tactile sensing to monitor grasps with robot-hand for planar elastic objects. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–7. VDE.

Hansen, N. (2006). The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer.

Intel (n.d.). ROS Wrapper for Intel RealSense Devices. https://github.com/IntelRealSense/realsense-ros. Accessed: 15-07-2020.

Jia, Y.-B., Guo, F., and Lin, H. (2014). Grasping deformable planar objects: Squeeze, stick/slip analysis, and energy-based optimalities. *The International Journal of Robotics Research*, 33(6):866–897.

Johansson, R. S. (1978). Tactile sensibility in the human hand: receptive field characteristics of mechanoreceptive units in the glabrous skin area. *The Journal of physiology*, 281(1):101–125.

Johansson, R. S. and Flanagan, J. R. (2009). Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580.

Kim, D., Maeda, Y., and Komiyama, S. (2019). Caging-based grasping of deformable objects for geometry-based robotic manipulation. *ROBOMECH Journal*, 6(1):3.

Klatzky, R. L. and Lederman, S. J. (2003). Touch. *Handbook of psychology*, pages 147–176.

Knibestöl, M. (1975). Stimulus-response functions of slowly adapting mechanoreceptors in the human glabrous skin area. *the Journal of Physiology*, 245(1):63–80.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE.

Krug, R., Lilienthal, A. J., Kragic, D., and Bekiroglu, Y. (2016). Analytic grasp success prediction with tactile feedback. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 165–171. IEEE.

Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE.

Land, M. F. and Furneaux, S. (1997). The knowledge base of the oculomotor system. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1358):1231–1239.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400.

Lee, M. A., Zhu, Y., Srinivasan, K., Shah, P., Savarese, S., Fei-Fei, L., Garg, A., and Bohg, J. (2019). Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950. IEEE.

Lin, H., Guo, F., Wang, F., and Jia, Y.-B. (2015). Picking up a soft 3D object by "feeling" the grip. *The International Journal of Robotics Research*, 34(11):1361–1384.

Loomis, J. M. and Lederman, S. J. (1986). Tactual perception. *Handbook of perception and human performances*, 2(2):2.

Miller, A. T. and Allen, P. K. (2004). Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122.

Misimi, E., Olofsson, A., Eilertsen, A., Øye, E. R., and Mathiassen, J. R. (2018). Robotic Handling of Compliant Food Objects by Robust Learning from Demonstration. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6972–6979. IEEE.

Nadon, F., Valencia, A., and Payeur, P. (2018). Multi-Modal Sensing and Robotic Manipulation of Non-Rigid Objects: A Survey. *Robotics*, 7(4):74.

Nelken, I., Chechik, G., Mrsic-Flogel, T. D., King, A. J., and Schnupp, J. W. (2005). Encoding stimulus information by spike numbers and mean response time in primary auditory cortex. *Journal of computational neuroscience*, 19(2):199–221.

Nguyen, V.-D. (1988). Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16.

Petit, A., Ficuciello, F., Fontanelli, G. A., Villani, L., and Siciliano, B. (2017). Using Physical Modeling and RGB-D Registration for Contact Force Sensing on Deformable Objects. In ScitePress and Springer, editors, *ICINCO 2017 - 14th International Conference on Informatics in Control, Automation and Robotics*, volume 2 of *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*, pages 24–33, Madrid, Spain. IEEE, ScitePress.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Rohmer, E., Singh, S. P., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE.

Romano, J. M., Hsiao, K., Niemeyer, G., Chitta, S., and Kuchenbecker, K. J. (2011). Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 27(6):1067–1079.

Sanchez, J., Corrales, J.-A., Bouzgarrou, B.-C., and Mezouar, Y. (2018). Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37(7):688–716.

Schneider, A., Sturm, J., Stachniss, C., Reisert, M., Burkhardt, H., and Burgard, W. (2009). Object identification with tactile sensors using bag-of-features. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 243–248. IEEE.

SCHUNK Robotics (2008). Kinematic parameter of SDH2. SDH2 Kinematics.

Simony, M., Milzy, S., Amendey, K., and Gross, H.-M. (2018). Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0.

Spall, J. C. (2005). *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons.

Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tian, Y., Yang, G., Wang, Z., Wang, H., Li, E., and Liang, Z. (2019). Apple detection during different growth stages in orchards using the improved yolo-v3 model. *Computers and electronics in agriculture*, 157:417–426.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Weiß, K. and Worn, H. (2005). The working principle of resistive tactile sensor cells. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 1, pages 471–476. IEEE.

Weiss Robotics (2008). Taktiler Meßwandler für Handlingaufgaben. Product Sheet DSA 9205.

Westling, G. and Johansson, R. (1984). Factors influencing the force control during precision grip. *Experimental brain research*, 53(2):277–284.

Westling, G. and Johansson, R. S. (1987). Responses in glabrous skin mechanoreceptors during precision grip in humans. *Experimental brain research*, 66(1):128–140.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Yi, W. and Marshall, S. (2000). Principal component analysis in application to object orientation. *Geo-spatial Information Science*, 3(3):76–78.

Zaidi, L., Corrales, J. A., Bouzgarrou, B. C., Mezouar, Y., and Sabourin, L. (2017). Model-based strategy for grasping 3D deformable objects using a multi-fingered robotic hand. *Robotics and Autonomous Systems*, 95:196–206.

Glossary

List of Acronyms

CMA	Covariance Matrix Adaptation
DOF	Degrees Of Freedom
FA-I	Fast-Adapting Type I
FA-II	Fast-Adapting Type II
FEM	Finite Element Method
LIFT	Learning of Initial Force and Tuning
OMPL	Open Motion Planning Library
PCA	Principle Component Analysis
PID	Proportional Integral Derivative
PRM	Probabilistic Roadmap Planners
RGB-D	Red Green Blue - Depth
ROS	Robot Operating System
RRT	Rapidly-exploring Random Trees
SA-I	Slow-Adapting type I
SA-II	Slow-Adapting type II
SDH	Schunk Dexterous Hand
YOLO	You Only Look Once