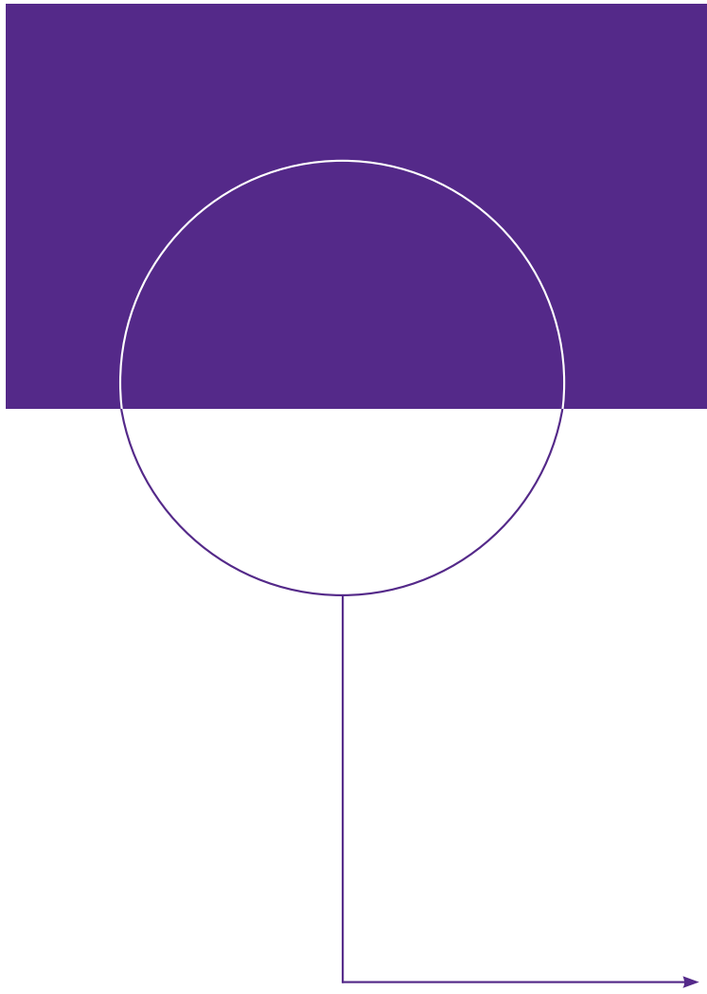


Master's thesis

2020

Marte Løkken Myrvold

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

Master's thesis

Marte Løkken Myrvold

On Estimating Forces and Torques in Manipulators

June 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

On Estimating Forces and Torques in Manipulators

Marte Løkken Myrvold

Cybernetics and Robotics
Submission date: June 2020
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Mathias Hauan Arbo

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

The aim of this project was to study the different state-of-the-art estimation techniques for estimating force-torque in robotic manipulators and to discuss platforms available at the department of Engineering Cybernetics for further experimentation. Historically, industrial manipulators used in manufacturing have been large, heavy machinery. The development of lightweight collaborative robots led to higher safety demands, because of their interaction with, and close proximity to humans. Because of this, the robots need to be able to sense their surroundings. Force sensors, though gradually becoming cheaper, are still expensive tools. Therefore, this project has looked at different sensorless force/torque estimation methods.

Four methods of estimating forces and torques in robots without using external force sensors were discussed: observer-based, least-squares, inverse dynamics and learning-based. Three robot programming middlewares and their applicability to the force estimation situation were considered: ROS, ROS 2 and Orocos. The various robot manipulators considered for use in experiments were UR5, UR3, KUKA's KR16-2 and LBR iiwa, and Franka Emika Panda.

The result of this study shows that several of the estimation methods rely on the availability of an accurate model of the dynamics of the robots. These models can be hard to figure out, and they might be sensitive to deviations and changes.

It was found that by using learning-based estimation techniques it is possible to estimate forces and torques without having to rely on having a correct analytical model, and that it is possible to use such methods to adaptively tune the parameters of the model dynamics. ROS was found to be the robot middleware most suited for this application. All four lightweight robots, that is UR5, UR3, KUKA LBR iiwa and Franka Emika Panda, are applicable.

Sammendrag

Denne avhandlingen skal studere ulike moderne estimeringsmetoder for å estimere kraft og dreiemoment i robotmanipulatorer, samt å diskutere hvilke plattformer som er tilgjengelig for bruk og videre eksperiment ved Institutt for teknisk kybernetikk. Industrielle roboter brukt i produksjon har tidligere bestått av store, tunge maskiner. Utviklingen av lettere roboter designet for å samarbeide med, og jobbe i nærheten av, mennesker har ført til strengere sikkerhetskrav, noe som gjør at de er nødt til å kunne oppfatte sine omgivelser. Ettersom sensorer for å måle kraft fortsatt er kostlige, vil det her bli sett på ulike metoder for å estimere kraft og dreiemoment som ikke baserer seg på eksterne kraftsensorer.

Fire estimeringsmetoder ble diskutert: observerbasert, minste kvadraters metode, invers dynamikk og læringbasert. Videre ble tre ulike middelvarer for robotikk og deres anvendbarhet for kraftestimering evaluert: ROS, ROS 2 og Orocos. De ulike robotmanipulatorene vurdert for bruk var UR5, UR3, KUKA sine KR16-2 og LBR iiwa, samt Franka Emika Panda.

Det viste seg at flere av estimeringsmetodene er avhengige av å ha korrekte modeller av dynamikken til robotene, hvilket kan være vanskelig å finne, samt at de kan være sårbare for avvik og endringer.

Det ble funnet at ved å bruke læringsbaserte metoder er det mulig å estimere kraft og dreiemoment uten å ha en korrekt analytisk modell, og at det er mulig å bruke disse metodene for å adaptivt stille inn parametrene til dynamikken til modellen. ROS viste seg å være middelvaren best egnet for denne applikasjonen, mens alle de fire lettvektrobotene, UR5, UR3, KUKA LBR iiwa og Franka Emika Panda, kan anvendes.

Preface

The plan was to perform experiments on actual robot hardware during the master thesis, but this was made impossible by COVID-19. Instead this thesis changed focus to describing more of the state-of-the-art literature, and evaluating the viability of the research as well as describing future experiments that can be performed.

I wish to express my sincere thanks to both my supervisor Jan Tommy Gravdahl and my co-supervisor Mathias Hauan Arbo for their support. The guidance and in-depth feedback I have received from Mathias during this time has been invaluable for my work, and he helped me push through and finish, even when COVID-19 made things look bleak.

I also want to thank my partner Thomas for his support, even when he himself was working on his master's thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis is an extension of the specialization project completed at the department of Engineering Cybernetics at NTNU, autumn 2019 by the author. Parts of this work are directly taken from the specialization project. The original intent was to perform experiments on robots, but COVID-19 made it difficult to perform experiments, and the project had to be changed accordingly.

1.1 Background

Historically robotic manipulators are large, heavy machinery capable of repeatedly manipulating products and tools at a high positional precision. The strength and precision stems from high mechanical stiffness and little to no backdrivability. In 2008 the first collaborative robots were launched, with the KUKA LBR4 and the Universal Robots, UR5. Because of their interaction and close proximity to humans, safety has become an important issue. Minimizing the risk they pose to humans involves minimizing their kinetic energy, and minimizing the interaction forces they can exert on their environment. Therefore, in addition to the robots being made of lighter construction material and having speed limitations, they also need to be able to sense their surroundings by considering external forces acting on the robot. Thus, it is important to be able to estimate forces and torques at the end-effector, and the torques at the joints.

One way to estimate forces and torques at the end-effector of a robot manipulator is by using external force/torque sensors. These are sensors that are able to feel the forces that are acting on the robot on all three geometric axes

(X-Y-Z), as well as the torques acting on these axes (yaw, pitch and roll). Wrist force sensors and joint torque sensors usually consists of strain gauges, which measure the strain on an object [1]. These gauges are sensitive to changes in temperature, making changes in ambient temperature a common cause of error in the strain measurement. The strain gauge must therefore be able to compensate for this sensitivity, which also requires a lot of calibration. The sensor has a designed rating of forces that it can take, as heavy payloads may lead to permanent deflections and degraded performance. Having a large range of allowable payloads with high noise sensitivity requires advanced material science. Because of these issues force sensors are generally expensive. In addition to this, external sensors contribute to the dynamical model of the system, and must be taken into account. For accurate force control it is also important to note that the sensor's reference frame is not the same as the frame in which the forces occur.

Another way of estimating the forces and torques is using the motor current, and translating this to the joint torques. This method relies on having a mathematical model of the robot and motor dynamics. Figuring out this model, and using this to get the torque at the end-effector is not a simple task. In addition, these models assume that the robot links and joints behave as rigid bodies, which will cause the model to be incorrect as modern robotics systems are highly non-linear, due to variable joint friction as the ball-bearing heats up, potential flexibility in the gearing, and other factors. In the case of lightweight robots, deviations in dynamic parameters, such as link masses, will cause a large percentage of error for model accuracy [2]. This also means that even small changes in the system will have to be taken into consideration when modeling.

1.2 Desired Properties

The goal of this thesis is to lay the groundwork for a possible application for force/torque estimation of the end-effector of robot manipulators. It is desirable for the application to be general in such a way that it will work on different robot platforms, and that it does not rely on having an accurate model of the robot available.

The application should be applicable to collaborative robots, as the motivation for this work stems from the increased safety requirements for robots working in close proximity with humans. In order to be able to estimate forces and torques, some sensor signals are required from the robots. Therefore, robots with internal sensors are preferred.

1.3 Report Structure

This project looks at sensorless force/torque estimation methods with lightweight robotic manipulators. The project gives an overview of state-of-the-art estimation techniques, and discusses platforms available at the department of Engineering Cybernetics for further experimentation. The report is structured as follows: In Chapter 2 a mathematical model for robot dynamics is presented, which is extended upon in order to take flexibility into consideration. Chapter 3 presents the different types of robots. The robots manipulators that were considered for this application are given in Chapter 4, with necessary information regarding them. In Chapter 5 information is given about different robot middleware solutions, and their relevance for this project. The different estimation methods are discussed in Chapter 6, and a table with an overview of the different methods are presented at the end of the chapter. The viability of the different state-of-the-art techniques, with a focus on the robots available at the department, as well as the usefulness of the different robots and middleware solutions are given in Chapter 7. In Chapter 8 further work is presented, and a conclusion is given in Chapter 9.

Chapter 2

Robot Dynamics

This chapter gives a brief introduction to robot kinematics before delving into describing the dynamics of a serially linked manipulator. The purpose of this chapter is to give a common notation when discussing the research on force estimation, and introduce some important core concepts.

2.1 Kinematics

Kinematics is about the motion of a robot manipulator without considering the forces and torques which are the causes of said motion [3]. Kinematics is fundamental for robotics in order to be able to design, analyse, control and simulate. Two of the most important aspects of kinematics are forward and inverse kinematics. Forward kinematics is used in order to determine the position and orientation of the end-effector by using the values of the joint variables, often relative to the base. Inverse kinematics is the inverse of this, namely to use the end-effectors position and orientation to get the values of the joint variables [1].

Coordinate frames are attached to each link in such a way that the coordinate frame $0_i x_i y_i z_i$ is attached to link i. These coordinate frames are necessary for forward kinematics. In Figure 2.1 an elbow manipulator, which is made up of three revolute joints, is shown with coordinate frames attached. The coordinate frame $o_0 x_0 y_0 z_0$ is the inertial frame, and is connected to the base of the robot. In Figure 2.1 coordinate frame $o_3 x_3 y_3 z_3$ is where the end-effector of the robot is, and where a tool can be installed. Coordinate frames are often represented by homogeneous transformation matrices, but can also be expressed

by a Cartesian vector and Quaternion [4], as well as other representations. In forward kinematics, the transformation matrix expresses the position and orientation of a coordinate frame with respect to a reference frame. The position and orientation of the end-effector frame, n, relative to the inertial frame is given by the homogeneous transformation matrix

$$\boldsymbol{T}_n^0 = \begin{bmatrix} \boldsymbol{R}_n^0 & \boldsymbol{o}_n^0 \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix}, \tag{2.1}$$

where $\boldsymbol{R}_n^0$ is the rotation matrix, and $\boldsymbol{o}_n^0$ is the coordinate vector from the origin of the inertia frame to the origin of frame n. The position and orientation of the end-effector frame is thus given by

$$\boldsymbol{T}_n^0 = \boldsymbol{A}_1(\boldsymbol{q}_1) \cdots \boldsymbol{A}_n(\boldsymbol{q}_n), \tag{2.2}$$

with each homogeneous transformation $\boldsymbol{A}_i$ being on the form

$$\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{R}_i^{i-1} & \boldsymbol{o}_i^{i-1} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \tag{2.3}$$

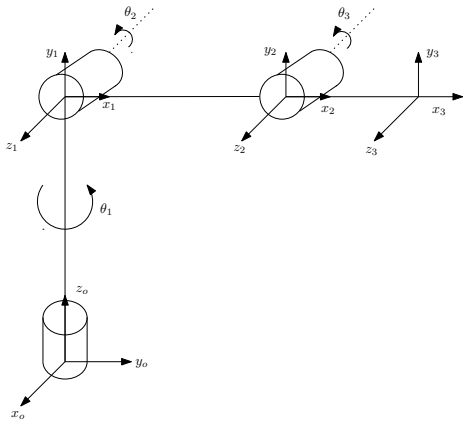
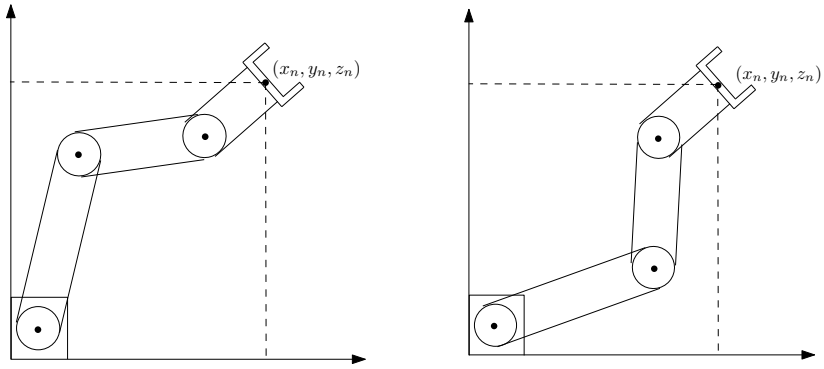


Figure 2.1: Elbow manipulator with coordinate frames attached

As mentioned, the problem of inverse kinematics is to find the values of the joint variables given the position and orientation of the end-effector. In other words, finding the pose of the robot, or how it should move in order to be in the correct pose, by knowing where the end-effector is, or should be. However,

inverse kinematics is not always as straightforward to define, as the problem requires the solution of sets of nonlinear equations. Because of this, it is possible that multiple solutions exists, as well as no solutions [3]. One example where multiple solutions are possible is the elbow manipulator. As seen in Figure 2.2 the manipulator can be in an elbow up configuration as in Figure 2.2a, or in an elbow down configuration as in Figure 2.2b. In both cases, the position and orientation of the end-effector is the same. Thus, both are valid solutions for the inverse kinematics problem of the given end-effector. The case where no solution exists occur when the pose is outside the robot's workspace, where the workspace is defined as the reachable area for the end-effector, and is determined by the geometry of the manipulator and the limits of the joint motions.



(a) Elbow up configuration

(b) Elbow down configuration

Figure 2.2: Elbow up and elbow down configurations

2.2 Dynamics

In contrast to kinematics, dynamics focuses on the forces acting on the robot and explains the relationship between force and motion. Equations of motion are an important part of dynamics, and can be used to simulate the motion of a robot, as well as to design controllers, and forms a basis for several computational algorithms. These algorithms are used for four important computations in dynamics: forward dynamics, inverse dynamics, the joint-space inertia matrix and the operational-space inertia matrix [3].

Forward dynamics is used to determine the joint accelerations by applying joint actuator forces and torques, which is necessary for simulation. Given joint positions, velocities and accelerations, inverse dynamics is used to compute the required joint actuator forces and torques to achieve the desired joint positions, velocities and accelerations. This is used when controlling a robot using a feed-forward controller, and for trajectory planning. The joint-space inertia matrix maps the joint accelerations to the forces and torques, and is used to linearize the dynamics in feedback control, as well as for analysis and forward dynamics. The operational space inertia matrix, which is used in control for the end-effector, maps the accelerations to task forces.

As mentioned, an integral part of dynamics is the equations of motion for a robot. The Euler-Lagrange equation of motion is one such equation, and it is given by

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{2.4}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are n-dimensional vectors representing respectively the joint position, speed, and acceleration, where n denotes the degrees of freedom for the system. If the robot is modelled as a series of rigid bodies, the $\boldsymbol{q}$ is the generalized coordinates of the mechanical system. $\boldsymbol{H}$ is the inertia matrix, which is an $n \times n$ symmetric positive-definite matrix. $\boldsymbol{C}$ is an $n \times n$ matrix such that $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ represents the Coriolis and centrifugal forces, while the vector $\boldsymbol{G}$ is the gravitational forces and $\boldsymbol{\tau}$ is an n-dimensional vector representing the joint torques. If the robot is not experiencing any external forces then with appropriate system identification, and estimation of joint speed and joint acceleration, the joint torques can be calculated by (2.4). A simple robot example experiencing an external force f is shown in Figure 2.3.

2.2.1 End-Effector Forces

The manipulator Jacobian, $\boldsymbol{J}$ is used to relate the linear and angular velocity of the end effector and the joint velocities, as well as between the joint torques and the force at the end-effector. The following derivations for the relationship between wrenches and torques is from page 121 in [5] by Murray et al.

Given $\boldsymbol{F}_t$ as the wrench and letting $\boldsymbol{g}_0(\boldsymbol{\theta}(t))$ represent the motion of the end-effector, the net work performed by the wrench over an integral of time from t_1 to t_2 is

$$W = \int_{t_1}^{t_2} \boldsymbol{v}_0^n \cdot \boldsymbol{F}_t dt, \tag{2.5}$$

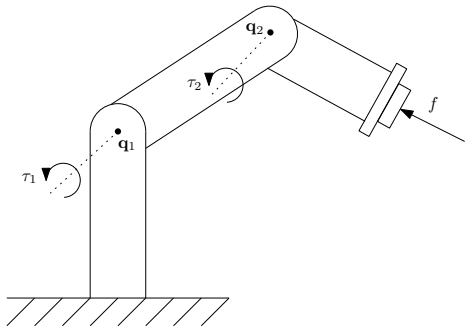


Figure 2.3: Two-link manipulator with two revolute joints. The robot experiences an external force f which results in torques τ_i on the joints.

where $\boldsymbol{v}_0^n$ is the body velocity of the end-effector. As we assume a frictionless system, this work will be the same as the work performed by the joints. Therefore, we get

$$\int_{t_1}^{t_2} \dot{\boldsymbol{\theta}} \cdot \boldsymbol{\tau} dt = W = \int_{t_1}^{t_2} \boldsymbol{v}_0^n \cdot \boldsymbol{F}_t dt. \tag{2.6}$$

This is true for any time interval, and therefore the integrands are equal. The manipulator Jacobian can then be used to relate $\boldsymbol{v}_0^n$ to $\dot{\boldsymbol{\theta}}$, we have

$$\dot{\boldsymbol{\theta}}^{\mathrm{T}} \boldsymbol{\tau} = \dot{\boldsymbol{\theta}}^{\mathrm{T}} (\boldsymbol{J}_0^n)^{\mathrm{T}} \boldsymbol{F}_t. \tag{2.7}$$

Since $\dot{\boldsymbol{\theta}}$ is free, it follows that

$$\boldsymbol{\tau} = \boldsymbol{J}^{\mathrm{T}} \boldsymbol{F} \tag{2.8}$$

2.2.2 Flexible Joints Mapping

The model presented in Figure 2.3 assumes that the robot can be modelled as a set of rigid bodies, and that the motor exerts torques directly on the rigid bodies through the joints. This does not take into account flexibility of the links of the robot or in the gears from motor to the links. Flexible elements introduces challenges with dynamic modeling, trajectory planning, and feedback control. This flexibility can either be in the joints or along the links. In both cases it is necessary to introduce additional generalized coordinates. The following derivation of the robot dynamics of flexible joints is as described in [3] by Siciliano and Khatib. When looking at joint flexibility of a robot with n joints, the generalized coordinates can be given by

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^{2n}, \tag{2.9}$$

where $\boldsymbol{q}$ and $\boldsymbol{\theta}$ are n-dimensional vectors of respectively link and motor positions.

The dynamic model of the system can be found by using the Lagrangian $\mathcal{L} = \mathcal{K} - \mathcal{P}$, where $\mathcal{K}$ is the kinetic energy of the robot and $\mathcal{P}$ is the potential energy. The potential energy of the system comes from the elasticity of the joints and from gravity. The joint elasticity is assumed to be linear, and is given by

$$\mathcal{P}_e = \frac{1}{2}(\boldsymbol{q} - \boldsymbol{\theta})^T \boldsymbol{K}(\boldsymbol{q} - \boldsymbol{\theta}), \tag{2.10}$$

$$\boldsymbol{K} = \text{diag}(K_1, ..., K_N), \tag{2.11}$$

with $K_i(\theta_i - q_i)$ as the torque τ_i that is transmitted to the ith joint.

It is also normal to choose generalized coordinates such that

$$\dot{\boldsymbol{H}}(\boldsymbol{q}) = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}^{\mathrm{T}}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{2.12}$$

which fits well with serially linked manipulators.

By assuming that the center of mass of the actuator's rotors are on the rotation axis and that they are uniformly distributed around the axis, the potential energy caused by gravity will only be dependent on the link positions $\boldsymbol{q}$. Thus, the potential energy caused by gravity is given by

$$\mathcal{P}_g = \mathcal{P}_{g,link}(\boldsymbol{q}) + \mathcal{P}_{g,motor}(\boldsymbol{q}). \tag{2.13}$$

The total potential energy is thus

$$\mathcal{P}(\boldsymbol{\Theta}) = \mathcal{P}_e(\boldsymbol{q} - \boldsymbol{\theta}) + \mathcal{P}_g(\boldsymbol{q}). \tag{2.14}$$

The kinetic energy for the robot is given by the kinetic energy from the links and from the rotors. For the links the energy is

$$\mathcal{K}_{link} = \frac{1}{2}\dot{\boldsymbol{q}}^{\mathbf{T}}\boldsymbol{M_L}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2.15}$$

where $\boldsymbol{M_L}$ is the link inertia matrix. For the rotors, we have

$$\mathcal{K}_{rotor} = \sum_{i=1}^{N} \mathcal{K}_{rotor_i} \tag{2.16}$$

$$= \sum_{i=1}^{N} (\frac{1}{2} m_{r_i} \boldsymbol{v}_{r_i}^{\mathrm{T}} \boldsymbol{v}_{r_i} + \frac{1}{2} {}^{R_i}\boldsymbol{\omega}_{r_i}^{\mathrm{T}} {}^{R_i}\boldsymbol{I}_{r_i} {}^{R_i}\boldsymbol{\omega}_{r_i}). \tag{2.17}$$

Here, the linear velocity of the center of mass and the angular velocity of the i-th rotor and rotor body are given by v_{r_i} and ω_{r_i}. R_i denotes the local rotational frame. The rotor inertia is ${}^{R_i}\boldsymbol{I}_{r_i} = \mathrm{diag}(I_{r_{ixx}}, I_{r_{iyy}}, I_{r_{izz}})$ with $I_{r_{ixx}} = I_{r_{iyy}}$. By using the following expression for the angular velocity

$${}^{R_i}\omega_{r_i} = \sum_{j=i}^{i-1} \boldsymbol{J}_{r_i,j}(\boldsymbol{q})\dot{q}_j + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{m,i} \end{bmatrix}, \tag{2.18}$$

with $\boldsymbol{J}_{r_i,j}(\boldsymbol{q})$ as the j-th column of the Jacobian, the kinetic energy of the rotor can be found as

$$\mathcal{K}_{rotor} = \frac{1}{2}\dot{\boldsymbol{q}}^{\mathrm{T}}[\boldsymbol{M}_R(\boldsymbol{q}) + \boldsymbol{S}(\boldsymbol{q})\boldsymbol{B}^{-1}\boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q})]\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^{\mathrm{T}}\boldsymbol{S}(\boldsymbol{q})\dot{\boldsymbol{\theta}} + \frac{1}{2}\dot{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{B}\dot{\boldsymbol{\theta}}, \tag{2.19}$$

where $\boldsymbol{M}_R(\boldsymbol{q})$ is the rotor masses, $\boldsymbol{S}(\boldsymbol{q})$ is a square matrix representing the inertial couplings between the links and the rotors, and $\boldsymbol{B}$ represents the rotor's inertial components $I_{r_{izz}}$, and is a constant diagonal matrix.

The total kinetic energy is then

$$\mathcal{K} = \frac{1}{2}\dot{\boldsymbol{\Theta}}^{\mathrm{T}}\boldsymbol{\mathcal{M}}(\boldsymbol{\Theta})\dot{\boldsymbol{\Theta}} \tag{2.20}$$

$$= \frac{1}{2}\begin{bmatrix}\dot{\boldsymbol{q}}^{\mathrm{T}} & \dot{\boldsymbol{\theta}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{M}(\boldsymbol{q}) & \boldsymbol{S}(\boldsymbol{q}) \\ \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}) & \boldsymbol{B}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\theta}}\end{bmatrix}, \tag{2.21}$$

with $\boldsymbol{M}(\boldsymbol{q}) = \boldsymbol{M}_L(\boldsymbol{q}) + \boldsymbol{M}_R(\boldsymbol{q}) + \boldsymbol{S}(\boldsymbol{q})\boldsymbol{B}^{-1}\boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q})$.

Now that the kinetic and potential energy of the system is found, the dynamics can be found using the Lagrangian.

$$\begin{bmatrix} \boldsymbol{M}(\boldsymbol{q}) & \boldsymbol{S}(\boldsymbol{q}) \\ \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}) & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{c}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{\theta}}) \\ \boldsymbol{c}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} + \begin{bmatrix} \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{K}(\boldsymbol{q} - \boldsymbol{\theta}) \\ \boldsymbol{K}(\boldsymbol{\theta} - \boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\tau}_{motor} \end{bmatrix} \tag{2.22}$$

It is possible to simplify this by assuming that the angular velocity of the rotors are only due to their own spinning, which reduces the angular velocity from (2.18) to simply

$${}^{R_i}\boldsymbol{\omega}_{r_i} = \begin{bmatrix} 0 & 0 & \dot{\theta}_{m_i} \end{bmatrix}^{\mathrm{T}}, \quad i = 1, ..., N. \tag{2.23}$$

Using this, we get $\boldsymbol{S} \equiv 0$, which means that the total angular kinetic energy of the rotors is reduced to $\frac{1}{2}\dot{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{B}\dot{\boldsymbol{\theta}}$.

Thus, the dynamics becomes

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{K}(\boldsymbol{q} - \boldsymbol{\theta}) = \boldsymbol{0} \tag{2.24}$$

$$\boldsymbol{B}\ddot{\boldsymbol{\theta}} + \boldsymbol{K}(\boldsymbol{\theta} - \boldsymbol{q}) = \boldsymbol{\tau}_{motor}, \tag{2.25}$$

with $\boldsymbol{M}(\boldsymbol{q}) = \boldsymbol{M}_L(\boldsymbol{q}) + \boldsymbol{M}_R(\boldsymbol{q})$. Note that we now have a coupled mechanical system between the motor dynamics and the rigid body dynamics.

2.3 Environment forces

Robot manipulators often perform tasks where they are in contact with the environment. In these cases, it is necessary to take the environmental forces acting on the manipulator into account when modelling. Therefore, (2.4) must be changed in order to account for these forces. This gives

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{J}_{f_{ext}}(\boldsymbol{q})^{\mathrm{T}}\boldsymbol{f}_{ext}, \tag{2.26}$$

where $\boldsymbol{J}_{f_{ext}}$ is the Jacobian to the point of interaction of the external force, and $\boldsymbol{f}_{ext}$ is the external force acting on the robot. Perfectly rigid models of the environment can be used, but it is also possible to model the environment as a mass-spring-damper (MSD) system coupled with the rigid-body dynamics model. By using a MSD system the equation of the environment model is given by

$$\boldsymbol{f}_{ext} = \boldsymbol{K}_e(\boldsymbol{q} - \boldsymbol{q}_e) + \boldsymbol{B}_e\dot{\boldsymbol{q}}, \tag{2.27}$$

where $\boldsymbol{K}_e$ is the environment stiffness matrix, $\boldsymbol{B}_e$ is the damping matrix, and $\boldsymbol{q}_e$ corresponds to the environment position [6].

The robot can also be affected by frictional forces that may occur in the robot joints or in the motor gears. This can affect the dynamics of the system, and thus the accuracy. It it possible to view friction as an external disturbance, which can be reduced by enhancing the robustness of the robot controller. This does, however, not take into account the nonlinear characteristics of the friction, such as hysteresis [7].

Chapter 3

Robots and Sensing

This chapter describes robots in terms of their general categories, and some of the often encountered hardware on them.

3.1 Industrial Robots

Industrial robots are robots used in manufacturing and can perform a number of tasks, such as welding, cutting, handling and more. In order to perform these tasks, the manipulators need to have high positional accuracy, which relies on sensing positioning errors. As there is often no direct measurement of the end-effector from sensors outside the robot, the position of the end-effector must be calculated using position measurements in the joints. This method relies on the geometry of the robot and encoders placed either after the motor, before the gearing mechanism, or at the link side. Because of this, robots are often designed with very high rigidity, in order to minimize errors in accuracy caused by things like computational errors and flexibility [1]. These robots have a high mechanical stiffness, which is important for position-controlled systems, as it maximizes bandwidth [8].

The payload of industrial robots is typically high, and it can be up to several hundred kilogram, such as the KUKA KR 500 FORTEC, which has a payload of 340 - 500 kg [9]. In industrial robots, the motors are usually not positioned at the joint, but they go through gears. As there are more components between the motor and link of an industrial robot, it may be harder to model friction than it is in a lightweight robot. In addition to this the higher gearing ratios

makes it more difficult to notice forces acting on the end-effector as changes in the current applied to the motors. An industrial robot can be seen in Figure 3.1.

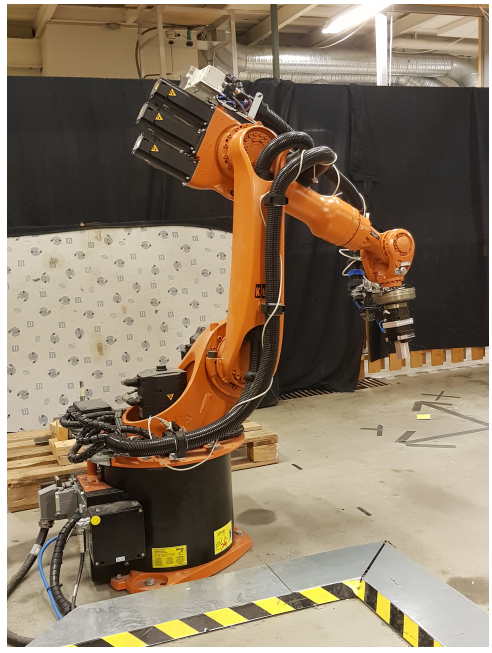


Figure 3.1: Industrial robot

3.2 Lightweight Robots

In 1993, the need for smaller and lighter robots arose, during the ROTEX space shuttle mission [10]. The robot arm was to be used in space, but in order for the astronauts to train for the mission they needed a robot arm on Earth. This robot arm was to have a payload to weight ratio of 1:1, which was not possible with the heavy pre-existing robots. Thus, one saw the need for a lightweight robot.

This led to the development of the DLR LWR at German Aerospace Center (DLR). Because of the desired payload-to-weight ratio, it was decided that the weight of the robot should not exceed 15 kg [11]. The DLR LWR was equipped

with a lightweight harmonic drive in the joints, and motors capable of high power at low speeds, which made it possible to achieve high performance despite the low weight of the robot.

Following the development of the DLR LWR, the first collaborative robots were developed, with both the 7 degrees of freedom (DOF) KUKA LBR4 and the 6 DOF Universal Robots UR5. While traditional industrial robots required workspaces with large safety zones to operate in, collaborative robots are designed to work alongside and interact with humans. This is made possible because of their reduced weight and inertia, as well as speed limitations. By making use of force/torque estimation and control, as well as backdrivability, it is possible to move the robot with physical touch, making it easy to control, and program by physically moving the robot to desired poses. This is not possible with traditional industrial manipulators, as they have a large gearbox which is not backdrivable. A lightweight robot can be seen in Figure 3.2.

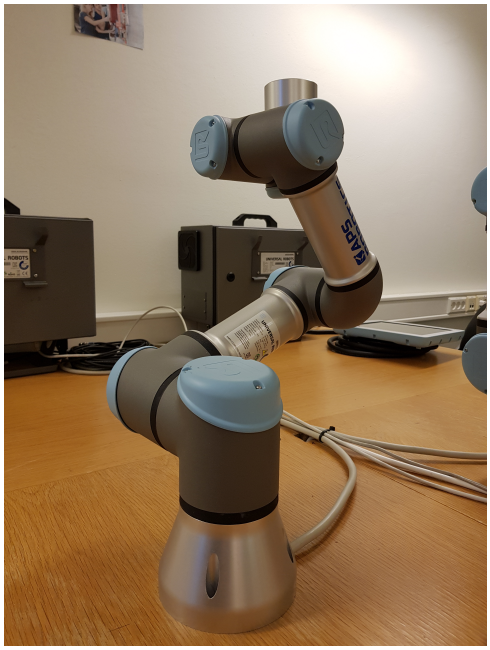


Figure 3.2: Lightweight robot

When it comes to forces and torques in robotic manipulators there are many

different solutions available. Consider for example the direct-drive robot and the SCARA robot. The first direct-drive robot was developed at Carnegie-Mellon University in 1981 [1]. While conventional robot arms consists of gears and gearboxes, such as the harmonic drive, the shafts of the articulated joints in direct-drive robots are directly coupled to the motor rotors [12]. The lack of gears causes direct-drive robots to have no backlash and low friction, as well as high stiffness which in turn leads to higher precision. In addition to this, direct-drive robots are backdrivable [13]. Even so, direct-drive robots are not as popular is conventional robots. The primary reason for this is that designing motors that are both strong enough and small enough is a challenge. In order to get powerful enough robots, the motors need to be rather big, as a reduction in size will make the torque density drop. In addition to this, there is a requirement for a cooling system, such as either fans or water-cooling, as direct drives generate a lot of heat [14].

In 1978 the first prototype of the SCARA (Selective Compliant Articulated Robot for Assembly), a 4 DOF manipulator, was developed by Professor Makino [15]. The SCARA manipulators are designed with three revolute joints and one prismatic, with the z-axes of the joints being mutually parallel [16][1]. The revolute joints are generally actuated with a motor situated in the base of the robot whose power is transmitted using flexible toothed belt, giving it selective compliance in the X-Y directions. This makes them well suited for assembly operations, such as pick and place, as their movements consists of translation along three directions and rotation about a vertical axis. In addition to this, they have a low footprint to workspace area. However, they do not possess the flexibility of 6, or more, DOF serially linked manipulators, which are able to handle objects in arbitrary poses.

3.3 Harmonic Drive

Harmonic drive is a strain wave gear, or harmonic gear, by the Harmonic Drive company, which is commonly used in robotic manipulators [17].

The gear is made up of a wave generator, an actuator shaft, a circular spline and a flex spline. The wave generator is connected to the actuator shaft and produces a torque T_{wg}.

The flex spline is located inside the circular spline and encompasses the wave generator, which has an elliptical disk. It is shaped like a cup [18]. The walls of the flex spline, as well as the open end, are flexible due to how thin the walls are, while the bottom is rigid. The outer wall consists of several teeth. The

flex spline produces the torque T_{fs} about each link. This torque is due to the movement of the wave generator, which pushes on the flex spline and causes a deformation of the flex spline such that it becomes elliptical.

Situated about the flex spline is the circular spline, which is a rigid circle with teeth on the inside. When the flex spline is pushed outwards by the wave generator, its teeth connects with the teeth of the circular spline.

Chapter 4

Robot Manipulators Considered

This chapter describes some of the robot manipulators available at the department of engineering cybernetics, and their specification with regards to their applicability for force estimation. In order to gauge the amount of times different robots presented in this chapter has been used in earlier research, a database search was performed using Scopus, and the number of document results will be presented. This will not give a complete picture of their usage, nor how well they perform in regards to force-torque estimation problems, but it will help to give an overview.

4.1 Universal Robots

Universal Robots was created in 2005 and is an industrial robot manufacturer that specializes in lightweight collaborative robots [19]. The product line consists of seven different robots: UR5, UR5e, UR10, UR10e, UR3, UR3e, and UR16e. Here, the focus will be on the UR5 and the UR3 as they are available at the department.

Both the UR5 and UR3 manipulators have a wide range of sensors available, which are available through the real-time data interface. This makes it possible to get output data from the manipulator over a standard TCP/IP connection.

The UR robots can be controlled by using a teach pendant, which is a panel equipped with a touch screen as well as a power button and an emergency

stop button. The screen runs Polyscope, which is UR's proprietary robot user interface. This makes it possible for users to easily program the robot by using the teach pendant itself, and to run existing programs.

The Universal Robot's UR5 was the first in line of robots that UR launched, and was available on the market in 2008. It is a general purpose robot built for medium-duty applications, with a payload of up to 5kg. It is categorized as a lightweight robot, weighing 18.4 kg. The manipulator has a reach of 850 mm [20]. The range of motion and maximum speed for the joints of the UR5 is given in Table 4.1.

Joint	**Range of motion**	**Maximum speed**
All 6 joints	±360°	180 °/s

Table 4.1: Range of motion and maximum speed for UR5

The UR3 is the smallest of the UR robots, with a weight of 11kg and a payload of 3kg. Its reach is also shorter than that of the UR5, at 500mm. The range of motion and maximum speed for the joints is given in Table 4.2.

Joint	**Range of motion**	**Maximum speed**
Base	±360°	180 °/s
Shoulder	±360°	180 °/s
Elbow	±360°	180 °/s
Wrist 1	±360°	360 °/s
Wrist 2	±360°	360 °/s
Wrist 3	Unlimited	360 °/s

Table 4.2: Range of motion and maximum speed for UR3

4.1.1 Real-Time Data Interface

Using the real-time data interface of the UR robots, it is possible to control and monitor the robots remotely through a TCP/IP connection. This provides the user with data that describes the state of the robot. Thus, it is possible to create a program that reads the data transmitted over this connection. The frequency of the system is 125Hz.

There are several different categories of data, such as robot mode, joint data, tool data, Cartesian info, kinematics info, configuration data and more. For

this project, information about the joints is of relevance, and thus the package containing joint data is of high importance. The different data available for each joint through this interface is listed in Table 4.3.

Name	Datatype	Description
q_actual	double	Actual joint positions
q_target	double	Target joint positions
qd_actual	double	Actual joint velocities
I_actual	float	Actual current
V_actual	float	Actual voltage

Table 4.3: Relevant joint data available from RTDI of UR robots

4.1.2 Applicability

According to the search performed using Scopus, both UR3 and UR5 has been used multiple times in previous research, but the UR5 is the one that stands out, with 153 document results when using the keyword "UR5", compared to "UR3", which gave 55 results. Considering the fact that UR5 was released in 2008, and that it was, together with KUKA LBR 4, one of the first commercially available collaborative robots it makes sense to have a higher amount of usage in previous research than UR3, which was released in 2015. Due to the high number of research using UR5, it is a viable option for future research, as there is much existing work to build from. The UR5 has been used in previous research in [21] by Berger et al. for a learning-based method of force/torque estimation, which will be discussed in Chapter 6.

Both robots from Universal Robots are backdrivable. This makes them well suited for human-robot collaboration, as they can be moved by a human through physical interaction. In addition to this, force/torque estimation is a highly relevant problem, as the robot has to know the amount of force exerted on it in order to know whether the force is above the threshold and therefore it should move, as well as knowing how far and in what direction to move.

From Table 4.3 one can see that information about both actual joint positions and actual joint velocities can be collected from UR3 and UR5 in real-time. As will be shown in Chapter 6, several techniques for estimating forces and torques rely on positions and velocity data from the joints. Therefore, the fact that both robots have internal sensors for this data, as well that it can be obtained and used in real-time with RTDI makes them good candidates for use in force/torque

estimation research.

4.2 KUKA

4.2.1 KUKA KR16-2

The KUKA KR16-2 is a 6 DOF industrial manipulator used to process and transfer components or products, and to handle tools. It has a rated payload of 16 kg, while weighing approximately 245 kg, and a maximum reach of 1612 mm. The arm consists of six components: hollow-shaft wrist, arm, electrical installations, base frame, rotating column and link arm.

The axis data is given in Table 4.4.

Axis	Range of motion, software limited	Speed with rated payload
1	±180°	200 °/s
2	+35° to -155°	200 °/s
3	+154° to -120°	195 °/s
4	±165°	370 °/s
5	±130°	310 °/s
6	Infinitely rotating	610 °/s

Table 4.4: Axis data of KUKA KR16

The robot is controlled using a Kuka Robot Controller (KRC), with KRC4 being the latest iteration and KRC2 being the one available at the department. The KRC includes robot controller, PLC, motion and safety control systems. The KRC4 has a control rate of 250 Hz, whereas the KRC2 has a control rate of approximately 83 Hz.

4.2.2 Robot Sensor Interface

This section is based on information found in [22]. The Robot Sensor Interface (RSI) is an add-on package for KUKA System Software (KSS), which is the OS of the KRC. It can be used for real-time communication between the robot controller and an external system via Ethernet by using the RSI object ST_ETHERNET. If signal processing is activated with ST_ETHERNET, the KRC connects to the external system as a client. The robot controller transfers KRC data to the external system using a cyclical data exchange in the interpolation cycle of 12 ms for the KRC2, and 4 ms for the KRC4. By activating the

internal read function of ST_ETHERNET large data sets can be structured. The function is activated using the keywords in Table 4.5.

Keyword	Description
DEF_RIst	Send the Cartesian actual position
DEF_RSol	Send the Cartesian command position
DEF_AIPos	Send the axis-specific actual position of axes A1 to A6
DEF_ASPos	Send the axis-specific command position of axes A1 to A6
DEF_EIPos	Send the axis-specific actual position of axes E1 to E6
DEF_ESPos	Send the axis-specific command position of axes E1 to E6
DEF_MACur	Send the motor currents of robot axes A1 to A6
DEF_MECur	Send the motor currents of external axes E1 to E6
DEF_Delay	Send the number of late data packets

Table 4.5: Pre-existing RSI objects

4.2.3 Applicability

Firstly, the KUKA KR16-2 is one of the robots with fewest document results on Scopus, with only 21 results for the keyword "KUKA KR16" and 36 results when simply using "KR16". This means there is not as much existing work to build on as some of the other robots, which can prove to be a challenge.

Secondly, unlike the other robots, this one is a heavy-duty industrial robot, and not a lightweight collaborative robot. Because of this, the robot is designed for other requirements. As it is not designed to work in proximity of a human, the need for force/torque estimation is not the same as in the other cases discussed. In addition to this, as explained in Section 3.2, industrial manipulators are not backdrivable, and thus moving the robot by touching it is not possible.

Because of this, KR16-2 is not as suited to the problem of estimating forces and torques when the focus is on robots that are to collaborate with humans. However, it can be interesting to use the robot as a way of evaluating whether the methods created are robust enough to work on an industrial manipulator, and to compare the industrial and lightweight manipulators' viability for force estimation.

4.2.4 KUKA LBR iiwa

The KUKA LBR iiwa is a lightweight robot with 7 DOF, and is designed for human-robot collaboration. Two versions of the robot is available, the LBR iiwa 7 R800 and the LBR iiwa 14 R820. The former has a payload of 7 kg and weighs 23.9 kg, while the latter has a payload of 14 kg and weighs 29.9 kg. The maximum reach of the robots are 800 mm and 820 mm respectively, as denoted by their names [23].

The axis data of the two robots is given in Table 4.6.

Axis	Range of motion	Speed with rated payload
LBR iiwa 7 R800		
1	±170°	98 °/s
2	±120°	98 °/s
3	±170°	100 °/s
4	±120°	130 °/s
5	±170°	140 °/s
6	±120°	180 °/s
7	±175°	180 °/s
LBR iiwa 14 R820		
1	±170°	85 °/s
2	±120°	85 °/s
3	±170°	100 °/s
4	±120°	75 °/s
5	±170°	130 °/s
6	±120°	135 °/s
7	±175°	135 °/s

Table 4.6: Axis data of KUKA LBR iiwa 7 R800 and KUKA LBR iiwa 14 R820

The controller for LBR iiwa is KUKA Sunrise Cabinet which uses the KUKA smartPAD as a user interface for operation. The smartPAD can be used to program the robot to perform a variety of tasks. This can be done using KUKA Robot Language (KRL).

4.2.5 Fast Research Interface

The Fast Research Interface is the interface for remote control of LBR iiwa. The following description of the interface is based on information from [24].

FRI gives access to the KRC at control rates of up to 1 kHz. By using UDP socket communication as communication protocol, the user does not have to use one specific operating system for their application. There are three control modes available for use: joint position control, joint impedance control, and Cartesian impedance control. Some of the data signals transmitted from KRC to remote computer are: measured position, current commanded position, measured torque, estimated external force/torque at the tool center point, current Jacobian, current mass matrix.

4.2.6 Applicability

KUKA LBR iiwa was the robot with the second highest amount of document results, after UR5. Both "KUKA LBR iiwa" and "LBR iiwa" was used as keywords, with the former giving 59 results, and the latter 67. It has previously been used by Bargsten et al. in [25] for a learning-based approach.

As the robot has 7 DOF, as opposed to 6 DOF like UR3 and UR5, it has a greater flexibility with regards to movements and positions. As such, it is able to handle more complicated tasks than one with 6 DOF. One drawback however, is that with an increase in the number of DOF, it becomes what is known as a redundant manipulator (having more than 6 DOF), and techniques for handling the extra degree of freedom must be used.

As signals such as position, torque, and Jacobian is available through the FRI, the LBR iiwa is well suited for the task of estimating force/torque. In addition, the estimated external force/torque from KRC can be used for comparison in experiments regarding the different estimation techniques.

4.3 Franka Emika Panda

Franka Emika Panda is a 7 DOF lightweight robot designed and developed by the German company Franka Emika. The robot weighs 18 kg, with a payload of 3 kg and maximum reach of 855 mm [26]. The axis data of the Panda is given in Table 4.7.

Panda is equipped with more than a hundred sensors, and it has link-side torque sensors in all 7 axes. Thus, it is able to sense its surroundings with high accuracy.

Axis	Range of motion	Speed with rated payload
1	±166°	150 °/s
2	±101°	150 °/s
3	±166°	150 °/s
4	-176° to -4°	150 °/s
5	±166°	180 °/s
6	-1° to 215°	180 °/s
7	±166°	180 °/s

Table 4.7: Axis data of Franka Emika Panda

4.3.1 Franka Control Interface

There are two components by which the Franka Control Interface (FCI) can be accessed, which are libfranka and franka_ros [27]:

- **libfranka** is a C++ library for the client side of the FCI. It provides interfaces for non-realtime commands, realtime commands, robot state and the model library, as well as taking care of network communication with Control. It has a control rate of 1 kHz.
- **franka_ros** is a ROS package which integrates the libranka library into ROS. It contains a description of the robot, based on the URDF format. It has support for both ROS Control and MoveIt!.

Data signals available through the FCI consist of, among others, measured joint position and velocity, torque at link side, estimation of external force/torque.

4.3.2 Applicability

Franka Emika Panda was the robot with the fewest amount of document results, with only 7. As it is the newest robot of the ones discussed, being launched in 2017, it makes sense that it has not been used in as much earlier research as the others. Although UR3 and KUKA LBR iiwa were launched only 2 and 3 years earlier, the fact that both Universal Robots and KUKA already had robots on the market and people familiar with their products might lower the threshold of using their newer robots in research. The fact that the Panda has not been as widely used in research as the other robots may make it more challenging to use, as there is not much work to use as groundwork. However, it means that there might be a lot of unrealized potential.

As the KUKA LBR iiwa, the Panda is a 7 DOF manipulator, and therefore it has the same advantages and disadvantages which that entails, and which is mentioned in Section 4.2.6. Both robots also give access to estimated external force/torque, which can be used in experiments to compare and evaluate the performance of estimators.

Chapter 5

Robot Middleware

This chapter describes some of the most common robot programming middlewares and argues which ones are most applicable to the force estimation situation considered.

Middleware is a software layer between the operating system (OS) and applications running on the OS. It is called a meta-operating system as it is not a real operating system, but is based on one, and thus provides services such as hardware abstraction, low-level device control, package management and more [28].

5.1 No Middleware

Most robots can be controlled from an external computer with a TCP/IP or UDP/IP connection. The Real-Time Data Interface for Universal Robots, as mentioned in Section 4.1.1, is an example of this. There are often existing libraries for communication and control of robot manipulators which are made by researchers or by the robot manufacturers themselves, as is the case with the libfranka library for Franka Control Interface [29] and Stanford's Fast Research Interface Library for the KUKA Robot Controller [30].

These sort of libraries creates dedicated software for one particular experiment or robot. This might make it more approachable and easier to program as no prior, nor new, knowledge of a middleware system is required for the programmer in order to use the libraries. The only knowledge necessary is the programming language they are developing in, as well as ensuring that the

network connections can be established.

5.2 ROS

ROS, or the Robot Operating System, is a popular middleware solution for control and communication with robot manipulators. Software in ROS is organized into loosely connected packages. The goal of these packages is to provide software in a way that is reusable, and they follow the principle that they should contain enough functionality to be useful, but not so much that it becomes heavyweight. A ROS package is simply a library located in the ROS path which can consist of nodes, libraries, and message definitions.

- Nodes in ROS are each responsible for some part of the control of the robot. A system commonly consists of many nodes, which are all programmed to perform one specific task each. The different nodes can communicate peer-to-peer over channels called topics, which the nodes subscribe and publish to. A graph showing what the relationship between nodes and topics is shown in Figure 5.1, where the turtlesim tutorial package is running. The ROS nodes are shown as ellipses.

- Topics are used in order for the nodes to communicate, which utilizes publish and subscribe semantics. A node can publish data to a topic, and all nodes subscribing to said topic will receive the information that was published. In Figure 5.1 the topic, /turtle1/cmd_vel, is shown as a rectangle. Here, /teleop_turtle is functioning as the publisher, and is publishing linear and angular velocity to the topic. Both remaining nodes subscribe to this topic, and are thus receiving messages from the first node. However, those two nodes do not communicate with each other, but only with /teleop_turtle as that is the only node that is publishing to the topic. However, the nodes are unaware of which nodes they communicate with, and focus rather on the data published or received [31].

- Services are another way for the nodes to communicate, which is a request based form of communication. They consist of a service and a client, and the client sends a request message to the service and awaits the reply. This is commonly used in distributed systems [32].

As ROS is organized in such a way that packages should be reusable, there are several existing packages available for use with regards to robot kinematics

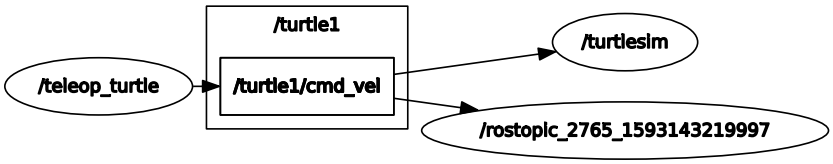


Figure 5.1: ROS nodes and topics

and dynamics. One such package is the Kinematics and Dynamics Library (KDL), which is distributed by the Orocos Project. The KDL package provides functionality for creating kinematic chains, as well as forward kinematics, inverse kinematics and inertia matrix. Using KDL parser, a KDL Tree object can be created from an Unified Robot Description Format (URDF) XML description. URDF is an XML file format that is used to describe a robot by specifying its kinematic and dynamic properties. It consists of links connected through joints, and several types of joints are supported, such as prismatic and revolute, among others. A diagram for visual representation of the URDF description for the UR3 manipulator was created using the graph visualization software rqt_graph, and can be seen in Figure 5.2. It shows a tree of how the links and joints of the robot are connected.

Another package that is available is the ur_modern_driver, which is a driver for Universal Robot's UR3, UR5 and UR10 with controllers CB2 and CB3. It provides nodes for communication with the controllers, which makes it possible to use with the real-time data interface as mentioned in Section 4.1.1. The user is therefore able to get much of the same information as when using the real-time data interface in Section 4.1.1, such as joint data, wrench in the base frame of the robot, and joint temperatures. The information from the robot is published on several topics and can be recorded as a rosbag for later use and replay.

However, as of November 24th 2019 the driver was deprecated in favour of Universal_Robots_ROS_Driver, which was developed in collaboration between Universal Robots and the FZI Research Center for Information Technology [33]. It uses the Real-Time Data Exchange (RTDE) for communication, instead of the earlier realtime data interface.

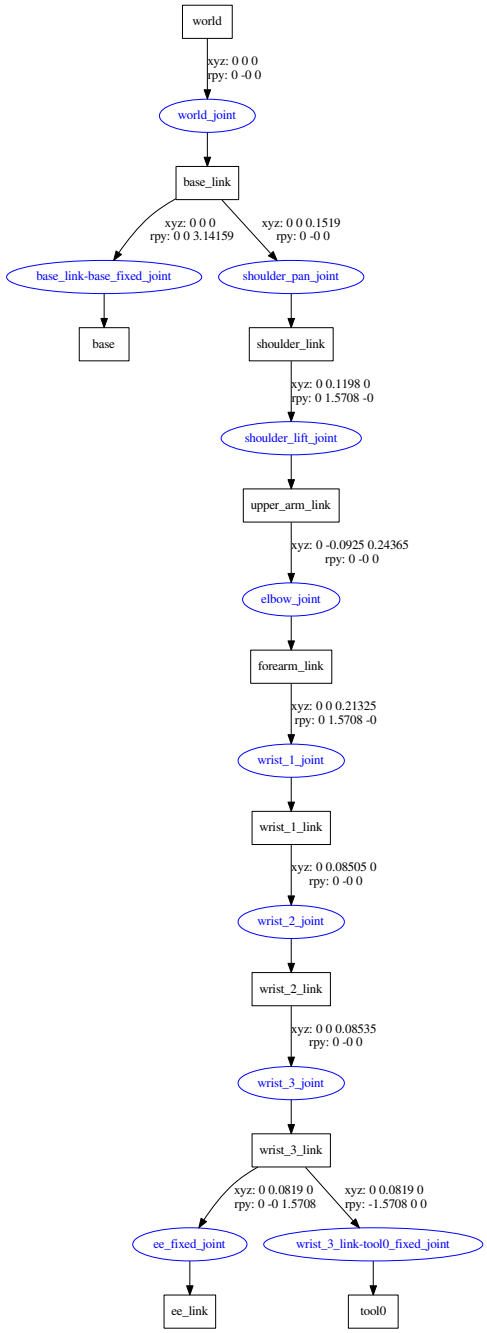


Figure 5.2: Visualization of URDF tree for UR3

5.3 ROS 2

In 2007 work started on developing ROS 2 [34]. When ROS 1 was developed, real-time constraints were not considered, and as such it does not support writing real-time code, but instead relies on external frameworks [35]. Because of this ROS 1 has real-time limitations, while ROS 2 was developed with real-time systems in mind, and thus supports real-time control directly. However, this project looks at the viability of techniques for estimating forces, and thus final integration questions such as real-time viability of middleware system involved is not as big of a problem.

Some of the more important differences between ROS and ROS 2 are shown in Table 5.1, which is based on information from [36] and [37].

5.4 Orocos

OROCOS is another middleware solution that has been widely employed in real-time dependent control situations. The idea of Orocos (Open RObot Control Software) came from Herman Bruyninckx in late 2000, after being disappointed by the commercial robot control software available, and the lack of access to the deepest layers of the hardware control. Thus, an idea for a more open control software for robotics was presented, and in 2001 the project started [38]. Modern Orocos includes the Orocos Toolchain which can be integrated with ROS.

- **The Orocos Toolchain** consists of libraries and programs used to create real-time applications. Two important componants of it are the Real-Time Toolkit and the Orocos Component Library.

- **The Real-Time Toolkit** is a component framework for real-time software and communication [39].

- **The Orocos Component Library** is used for developing interchangeable software components that can be controlled with a lifecycle based on a finite state machine, and deterministically updated [40].

- **Orocos Kinematics and Dynamics Library (KDL)** is a library for computation and modelling of kinematic chains. It provides support for geometric primitives, such as frame, point and twist, as well as kinematic and dynamic solvers, which includes forward and inverse kinematics [41].

Features	ROS	ROS 2
Platforms	Tested on Ubuntu and maintained on other Linux distributions and OS X	Currently tested and supported on Ubuntu Xenial, Windows 10 and OS X El Capitan
C++	C++03	C++11 and parts of C++14
Python	Python 2	Requires at least Python 3.5
Middleware	Custom serialization format and transport protocol	Based on the DDS standard, which uses UDP, and provides several Quality of Service (QoS) policies
Unify duration and time types	Duration and time types are in C++ and Python, and are defined in the client library	Defined as messages, therefore consistent across languages
Components with life cycle	Every node has its own main function	Tools such as roslaunch may be used to start a system composed of many components in a deterministic way
Threading model	Can only choose between single-threaded or multi-threaded execution	More granular execution models available
Multiple nodes	Not possible to create more than one node in a process	Possible to create multiple nodes in a process
Real-time support	Does not support writing real-time code	Possible to write real-time nodes

Table 5.1: Features differences of ROS and ROS 2

- **Orocos Bayesian Filtering Library (BFL)** is a framework for inference in Dynamic Bayesian Networks, and it includes estimation algorithms such as Kalman Filters and Particle Filters [42].

5.5 Comparison of middleware solutions

This section will be used to compare the different middleware solutions and to conclude which is most appropriate for this application, and why that should be pursued.

As discussed in Section 5.1, using no middleware is a possibility. However,

as stated, these libraries are very specific and also lack reusability. If the goal is to create an application which only works on one robot, and one does not want to spend time learning a new middleware, this might be a feasible solution, but for this application it is not.

Three different middlewares were described in the sections above: ROS, ROS 2 and Orocos.

One major advantage of ROS is that it is the most commonly available middleware of the ones presented. ROS has a worldwide user base with over 11 000 unique packages downloaded in July 2019 and more than 100 000 wiki pages [43]. Because one of the defining factors of ROS is that packages should be reusable, this means that there already exists a lot of packages readily available to use, as well as documentation and multiple examples of how to use them. In addition to this, there are ROS packages for all of the robots presented in Chapter 4,

However, the introduction of ROS 2 opens several new possibilities which were not possible with ROS. One of the most important aspects is the support of real-time computing. Another is that it introduces the ability to do composition of nodes, meaning that you can publish and subscribe via shared memory. In regards of this application, this will make a force estimation - force control loop run faster. Another aspect is the Python support. As mentioned in Table 5.1, while ROS 2 requires at least Python 3.5, ROS is still targeting Python 2. However, as of the beginning of 2020, Python 2 is no longer supported by the Python Software Foundation, which means there will be no new changes nor bug fixes. This, combined with the new key features of ROS 2 makes it a more future proof option. Even so, ROS 2 is still in development, and there is still a lack of compatible packages and software stability.

Orocos is primarily used at Katholieke Universiteit Leuven, and does not have the large userbase and the amount of packages ROS does. In addition to this, documentation is sparse, which makes it more difficult to figure out how to use.

In conclusion, the best options are ROS and ROS 2. As ROS 2 is still in the works, ROS is best suited for this application as of now. However, in the future when ROS 2 is more stable and more packages are available, it will be the better option because of the multiple key features which ROS is lacking.

Chapter 6

Estimating Force and Torque

In this chapter, different state-of-the-art estimation techniques for estimating forces and torques in the end-effector are presented, and at the end a table is provided which gives an overview of the different articles and the techniques used.

6.1 State-of-the-Art Estimation Techniques

There is a lot of previous research on how to estimate forces and torques in robot manipulators. A majority of these use sensors for position or motor current as input to the estimator. However, this presents difficulties, such as how to handle large disturbances caused by things like friction. There are several methods to overcome this challenge, such as modeling the disturbance, using an observer, or using adaptive methods. In this section, earlier research on how to estimate forces and torques will be presented, several of which take into consideration how to handle disturbances.

In order to determine which methods and what research to review, a systematic literature review was performed using two different databases and several keywords. The databases used were Scopus and Web of Science. Scopus is an abstract and citation database of peer-reviewed literature for Elsevier, while Web of Science provides access to multiple databases. Both can be used to get comprehensive data about articles and citations, as well as analyses of search results.

The purpose of this review is to evaluate what is existing literature on the

topic, and create an overview of what methods have been used. Particularly if force estimation is to be applied on a variety of robot platforms, it is important to note whether a particular technique, such as observer-based estimation, has been demonstrated by researchers on different robot platforms. Another growing field is machine learning, and as such, another point to evaluate was whether there has been much work on learning-based approaches to force estimation.

Several searches using different keywords were performed. The keyword "force estimation" was included in all of the searches. This was combined with either "robotics" or "robots", where the former resulted in 151 document results while the latter gave 177 document results when using Scopus. Web of Science gave more results overall, but less results when the word "robotics" was used than Scopus did. In total, Web of Science gave 391 document results when using the keyword "force estimation" combined with either "robots" or "robotics", where the former gave 245 results, and the latter gave 146. This shows that the keyword "robots" is more widely used than "robotics", and thus it will be the keyword used for further analysis. The databases also gives an overview of the number of documents published each year, which can be seen in Figure 6.1, where this development is shown for both Scopus and Web of Science, as well as the differences between number of documents containing "robots" or "robotics". From these graphs it can be seen that force estimation for robots is a relatively new research area, with the first result from Scopus being from 1989, while the first document in Web of Sciences databases is from 1996. In addition to this, the figure shows this is a growing area of research with an approximately exponential growth the last few years, which goes to show that this is a highly relevant area for further research.

The analysis of state-of-the art estimation techniques from the specialization project focused on four methods of force estimation: observer-based, least-squares, inverse dynamics and learning-based. Thus, searches with these keywords were also performed. As the keyword "robots" gave more results than "robotics" in previous searches, this was chosen as the keyword to use together with "force estimation" as before, as well as the methods mentioned. By looking at the results from this, one can conclude that observer-based and learning-based are the most commonly used methods, as the former gave 36 results and the latter 9 results when using Scopus, while least-squares and inverse dynamics gave 5 and 3 results respectively. As with the searches where the methods were omitted, Web of Science gave more results than Scopus when including the keywords "observer" and "learning", with 73 results for the first and the latter giving 16 results. In Figures 6.2 and 6.3 the number of documents are shown for each keyword per year for Scopus and Web of Science respectively.

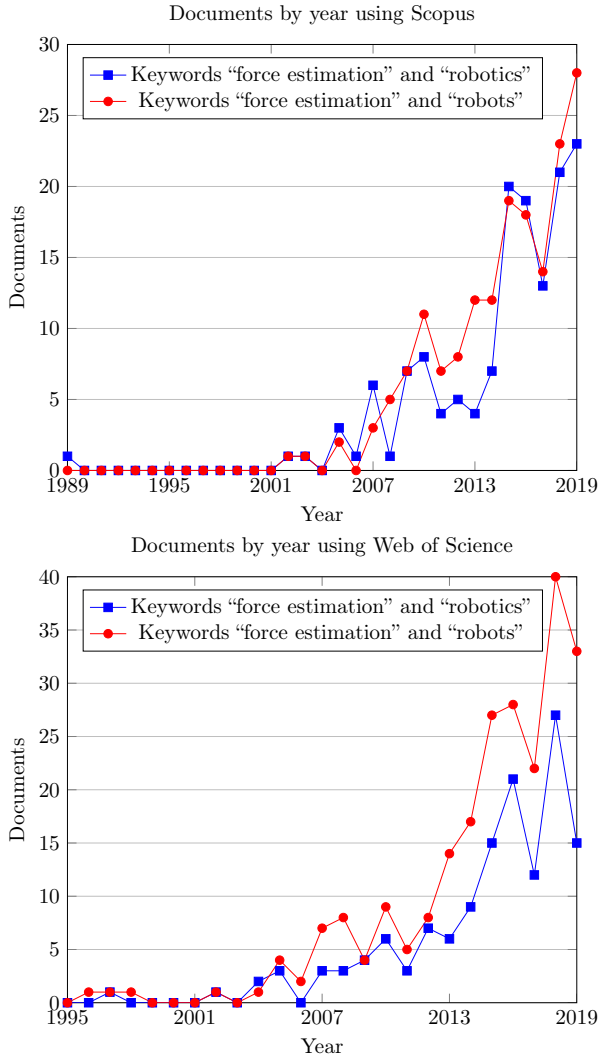


Figure 6.1: Document results by year with the keywords “force estimation”, “robots” and “robotics” for Scopus and Web of Science

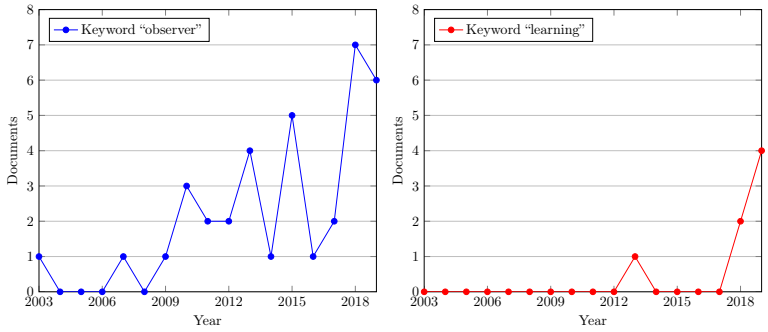


Figure 6.2: Document results of keywords "force estimation" and "robots" combined with either "observer" or "learning", Scopus

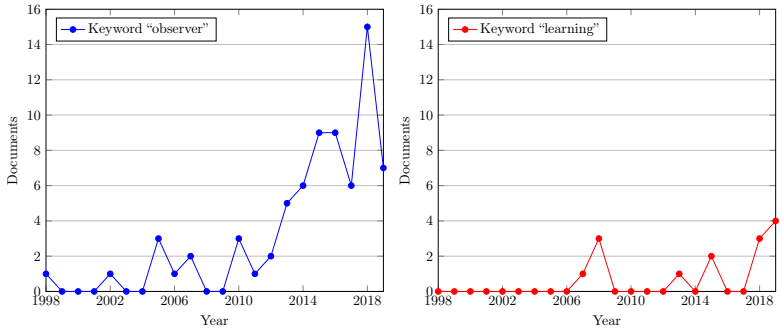


Figure 6.3: Document results of keywords "force estimation" and "robots" combined with either "observer" or "learning", Web of Science

By looking at relevant keywords and number of citations 16 articles were chosen for further review. The number of citations each article has will be presented when their results are discussed. As Figures 6.2 and 6.3 shows observer-based force estimation is more commonly used than learning-based. However, this might be due to the novelty of machine learning and learning-based methods rather than relevance and quality in regards to force estimation. In order to get an overview of how different techniques are applied and their performances all four methods mentioned previously will be discussed. The focus will be on observer-based and learning-based approaches, as the former is the most com-

monly used of the four, and the latter because machine-learning is a growing research field and shows promising results when used in force estimation.

Hacksel and Salcudean proposed to use observers in order to estimate the environment forces and rigid-body velocities of a robot in [44] (67 citations in Scopus and 48 in Web of Science). This was done by thinking of the environment force, or torque, as if it was controlling a damped mass-spring system, which represented the observer error dynamics. One can then measure the displacement of the spring from its equilibrium as if it acts as a stiff spring, which in turn can give an estimate of the environment forces and torques. Starting with a simple state observer for a rigid body, the authors presented the following linear force estimator for the environment forces

$$\boldsymbol{f}_{env} = k_p \tilde{\boldsymbol{x}}, \tag{6.1}$$

where k_p is a positive constant, and $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \hat{\boldsymbol{x}}$ with $\hat{\boldsymbol{x}}$ as the observer values for the position $\boldsymbol{x}$. Hacksel and Salcudean used the velocity observer presented in [45] with modification to account for the force observer. Rotations were parametrized by Euler parameters. In order to estimate environment torque the difference in orientation between the actual body and observed body was used. This resulted in the following error equations

$$\begin{aligned} \dot{\boldsymbol{\mu}} &= \boldsymbol{\tau}_{env} - \frac{1}{2} k_p \boldsymbol{\mathcal{I}}^{-1} \boldsymbol{e} \operatorname{sgn}(e_0) \\ \dot{e}_0 &= -\frac{1}{2} \boldsymbol{e}^{\mathrm{T}} \boldsymbol{\mathcal{I}}^{-1} (\boldsymbol{\mu} - k_v \boldsymbol{e} \operatorname{sgn}(e_o)) \\ \dot{\boldsymbol{e}} &= \frac{1}{2} [e_0 \boldsymbol{I} - (\boldsymbol{e}\boldsymbol{x})] \boldsymbol{\mathcal{I}}^{-1} (\boldsymbol{\mu} - k_v e \operatorname{sgn}(e_0)), \end{aligned} \tag{6.2}$$

where $\mu = \boldsymbol{\mathcal{I}}(\omega - \hat{\omega})$, $\boldsymbol{\mathcal{I}}$ and ω are respectively the inertia matrix and the angular velocity of the body, and e_0 and e are the scalar and vector part of the Euler parameter representation of the attitude error between the actual body and the observed body frames, and k_v is a positive constant. From this, the estimate of the environment torque becomes

$$\boldsymbol{\tau}_{env} = \frac{k_p}{2} \boldsymbol{\mathcal{I}}^{-1} \boldsymbol{e}, \tag{6.3}$$

which was shown to be a good estimation at low frequencies, but deteriorates at larger torques.

This was extended to serial mechanisms, in order to handle a general robot

model as in (2.4). The observer used is given by

$$\begin{aligned}\dot{\hat{\boldsymbol{x}}}_1 &= \hat{\boldsymbol{x}}_2 + k_v\tilde{\boldsymbol{x}}_1 \\ \dot{\hat{\boldsymbol{x}}}_2 &= \boldsymbol{H}^{-1}(\boldsymbol{q})(-\boldsymbol{C}(\boldsymbol{q},\dot{\hat{\boldsymbol{x}}}_1)\dot{\hat{\boldsymbol{x}}}_1 - \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{K}_p\tilde{\boldsymbol{x}}_1 + \boldsymbol{\tau} \\ \tilde{\boldsymbol{x}}_1 &= \boldsymbol{q} - \boldsymbol{x}_1,\end{aligned} \tag{6.4}$$

with the error dynamics

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\tilde{\boldsymbol{x}}}_1 + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\tilde{\boldsymbol{x}}}_1 + C(\boldsymbol{q},\dot{\hat{\boldsymbol{x}}}_1)\dot{\tilde{\boldsymbol{x}}}_1 = -\boldsymbol{K}_p\tilde{\boldsymbol{x}}_1 - k_v\boldsymbol{H}(\boldsymbol{q})\dot{\tilde{\boldsymbol{x}}}_1 + \boldsymbol{\tau}_{env}, \tag{6.5}$$

where $\boldsymbol{\tau}_{env}$ is an environment input added to the motor torques $\boldsymbol{\tau}$.

The experiments were performed on a UBC maglev wrist with 6 DOF. In order to compare between actual torques and the observer-based torque estimates, a JR3 force/torque sensor was used. This showed only a small deviation between the actual and estimated forces and torques, thus it was shown that it is possible to very closely estimate forces and torques using their proposed observer without the need of external sensors.

In [46] (41 citations in Scopus and 34 in Web of Science), Van Damme et al. presented two ways of estimating the force at the end-effector. The first uses a filtered dynamic model and applies a recursive least-squares estimation with exponential forgetting. The second approach uses a generalized momentum-based disturbance observer. The reason for using a filtered dynamic model is to not be dependent on the acceleration $\ddot{\boldsymbol{q}}$, as the acceleration will be very noisy since it is acquired by taking the derivative of the position twice. A first order filter was used to filter each side of the dynamics equations of a serial robot

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{\tau}_f(\dot{\boldsymbol{q}}) = \boldsymbol{K}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\boldsymbol{\theta} = \boldsymbol{\tau}, \tag{6.6}$$

where $\boldsymbol{K}$ is the inverse dynamics separated from the system parameters and $\boldsymbol{\theta}$ is the vector of the system parameters in vector form.

The filter had the following transfer function and impulse response

$$F(s) = \frac{1}{\frac{s}{\omega} + 1} \tag{6.7}$$

$$f(t) = \mathcal{L}^{-1}\{F(s)\} = \omega e^{-\omega t}. \tag{6.8}$$

Applying the filter to (6.6) results in

$$\boldsymbol{K}_f(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{\theta} = \langle\boldsymbol{\tau}\rangle_{F(s)}, \tag{6.9}$$

where $\boldsymbol{K}_f(\boldsymbol{q}, \dot{\boldsymbol{q}})$ has replaced $\boldsymbol{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, and thus it is no longer dependent on the acceleration $\ddot{\boldsymbol{q}}$. Introducing an external force to the system leads to the dynamics as given in (2.26). The filtered equation is given as

$$\langle \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}) \rangle_{F(s)} \boldsymbol{f}_e = \boldsymbol{K}_f(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{\theta} - \langle \boldsymbol{\tau} \rangle_{F(s)} \tag{6.10}$$

where $\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q})$ is the Jacobian to the end-effector, while $\boldsymbol{f}_e$ is the 6×1 end-effector wrench vector of external forces and torques. Using weighted least-squares estimation one can estimate the external wrench as

$$\hat{\boldsymbol{f}}_e = \mathrm{argmin}_{\boldsymbol{f}_e} \sum_{k=1}^{N} \beta(N, k) \cdot (\boldsymbol{y}_f[k] - \boldsymbol{A}_f[k]\boldsymbol{f}_e)^2, \tag{6.11}$$

with N measurements, and with $\boldsymbol{A}_f = \langle \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}) \rangle_{F(s)}$ and $\boldsymbol{y}_f = \boldsymbol{K}_f(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{\theta} - \langle \boldsymbol{\tau} \rangle_{F(s)}$.

The momentum-based observer introduces a disturbance torque $\boldsymbol{\tau}_d$ in the joints. The dynamic equation then becomes

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{\tau}_f(\dot{\boldsymbol{q}}) = \boldsymbol{\tau} + \boldsymbol{\tau}_d. \tag{6.12}$$

Using this, the authors arrived at the following estimator for the external wrench $\boldsymbol{f}_e$

$$\boldsymbol{f}_e = (\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}))^{-1}\boldsymbol{r}, \tag{6.13}$$

with $\boldsymbol{r} = \boldsymbol{K}_I\boldsymbol{e}$, and where $\boldsymbol{K}_I$ is a diagonal gain matrix with positive gains.

Both methods were tested on a pneumatic manipulator arm, which has 2 DOF. An external force sensor was used to measure the force exerted by the actuators. First, simulations were performed. The pneumatic muscle actuators were kept at a constant pressure. The result showed that there was almost no noticeable deviation between applied force and estimated force, both using least-squares and observer. Subsequently, a sensorless admittance controller was implemented. Weights were attached to the end-effector in order to test the accuracy of the force estimation, which showed good results. The result showed that both approaches worked well in order to estimate force, and that they did not differ much, though the observer-based estimator was noisier than the recursive least-squares approach.

A learning-based approach was presented by Colome et al. in [2] (40 citations in Scopus and 27 in Web of Science), used in combination with a disturbance state observer. This eliminates the need for an analytical model of the robot

dynamics. Using inverse kinematics gives

$$\boldsymbol{\tau}_T = \boldsymbol{\tau}_c - \boldsymbol{\tau}_e \tag{6.14}$$

$$\boldsymbol{\tau}_c = \boldsymbol{K}_\theta(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}), \tag{6.15}$$

where $\boldsymbol{\tau}_T$ is the vector of total input forces to the joints, $\boldsymbol{\tau}_c$ is the applied torque commands and $\boldsymbol{\tau}_e$ is the external torque. In order to obtain the function $\boldsymbol{K}_\theta$, meaning to approximate the inverse dynamics of the robot, the authors used the method Locally Weighted Projection Regression (LWPR) [47] and the LWPR open access library [48] for model learning. Using this, one can assume that $\boldsymbol{K}_\theta$ is learned, which follows that $\boldsymbol{\tau}_c$ is known. As acceleration is only available as a second derivative of the joint positions, it is highly susceptible to noise. Thus, when learning a task, they used the desired trajectory and its resultant accelerations instead of the joint position measurements.

In [49] (36 citations in Scopus and 31 in Web of Science) by Zhang et al., a harmonic drive compliance model based on position measurements was presented and used to estimate torque in robotic joints. The wave generator of the harmonic drive was driven using a DC motor which has the ability to measure the angular displacement of the rotor. In order to measure the output angle of the joint, an absolute position encoder was connected to the harmonic drive on the link-side.

In order to test the estimation scheme, an experimental setup with three joints was used. The first joint was the test-joint, that is, the joint with a harmonic drive and link-side encoder. Four experiments were performed: the first one with gradually changing load torque, the second with rapidly changing torque, the third with a sudden payload change, and in the final experiment external static torques were applied. The results were compared to the torque measurements from a force/torque sensor. The maximum differences between the estimated torque and the torque sensor never exceeded 1.9 Nm, and the root mean square error (RMSE) levels of the differences were below 0.5 Nm for all experiments. In order to demonstrate the effectiveness of the joint torque estimation, it was used for motion control. The RMSE values of the tracking showed a 75% improvement when using joint torque feedback with the torque estimate. The torque sensor still proved to perform better, with an 80% improvement.

In [50] (28 citations in Scopus and 18 in Web of Science), Wahrburg et al. used a combination of generalized momentum observer and disturbance observer in order to estimate external wrench. The generalized momentum disturbance

observer is given by

$$\dot{\hat{\boldsymbol{p}}} = \bar{\boldsymbol{\tau}} + \boldsymbol{L}(\boldsymbol{p} - \hat{\boldsymbol{p}}) \tag{6.16}$$

$$\hat{\boldsymbol{f}} = -(\boldsymbol{J}^{\mathrm{T}})^{+} \cdot \boldsymbol{L}(\boldsymbol{p} - \hat{\boldsymbol{p}}), \tag{6.17}$$

with $\bar{\boldsymbol{\tau}} = \boldsymbol{\tau}_{motor} + \boldsymbol{C}^{\mathrm{T}}\dot{\boldsymbol{q}} - \boldsymbol{G} - \boldsymbol{\tau}_{fric}$, and a positive definite matrix $\boldsymbol{L}$.

Generalized momentum does not depend on acceleration $\ddot{\boldsymbol{q}}$, which reduces errors resulting from differentiating position and velocity. It does, however, have its drawbacks, such as assuming ideal friction estimation. In addition, $\boldsymbol{\tau}_{motor}$ must be known, and a good model for $\boldsymbol{C}$ is needed. Therefore, using a Kalman filter approach based on the generalized momentum was suggested, in order to take into account both model uncertainties and disturbances, by combining the aforementioned method with a disturbance observer.

The estimation scheme was tested on a simulated ABB YuMi manipulator, and compared with a generalized momentum observer. The result showed that the proposed scheme had improved accuracy over the standard generalized momentum observer reference because of inaccuracies when estimating joint friction.

Two methods to estimate force was presented by Stolt in [51] (64 citations in Scopus and 49 in Web of Science for [52], which is included in his PhD Thesis), by using the joint position control errors and the joint motor torques. In some industrial robots, the joints are controlled by individual PID controllers with a feed forward based on the robot dynamics. These controllers have an integral action which can be disabled in order for the controllers to act as virtual springs when under static loads. An external force acting on the joints will cause a deviation of the joint angle from its reference point. This deviation will then correspond to the torque of that joint by the virtual spring equations. Thus, one can use the same methods as if each joint had its own torque sensor in order to find the external force that acted upon the manipulator. The joint torques are then given by

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}}\boldsymbol{F} + \boldsymbol{e}, \tag{6.18}$$

where $\boldsymbol{J}(\boldsymbol{q})$ is the Jacobian to the point that the external force wrench acts on the robot, $\boldsymbol{q}$ are the joint positions, $\boldsymbol{F}$ is the end-effector wrench, and $\boldsymbol{e}$ are the disturbance joint torques. The minimum variance estimate of the force is given by $\hat{\boldsymbol{F}} = (\boldsymbol{J}\boldsymbol{R}_e^{-1}\boldsymbol{J}^{\mathrm{T}})^{-1}\boldsymbol{J}\boldsymbol{R}_e^{-1}\boldsymbol{\tau}$. This may be a poor estimate at large disturbances according to Stolt. A solution to this is to use a priori knowledge about the particular assembly operation and adopting a Bayesian approach. Using this, the minimum variance unbiased estimate of $\boldsymbol{F}$ is given by

$$\hat{\boldsymbol{F}} = (\boldsymbol{J}\boldsymbol{R}_e^{-1}\boldsymbol{J}^{\mathrm{T}} + \boldsymbol{R}_F^{-1})^{-1}(\boldsymbol{J}\boldsymbol{R}_e^{-1}\boldsymbol{\tau} + \boldsymbol{R}_F^{-1}\bar{\boldsymbol{F}}), \tag{6.19}$$

where prior knowledge of $\boldsymbol{F}$ is described as $\mathbb{E}[\boldsymbol{F}] = \bar{\boldsymbol{F}}$ and $\mathbb{E}[\boldsymbol{F} - \bar{\boldsymbol{F}}(\boldsymbol{F} - \bar{\boldsymbol{F}})^{\mathrm{T}}] = \boldsymbol{R}_F$, where $\mathbb{E}$ denotes the expected value. The integral part of the controllers highly affected the performance of the estimator. If it was removed completely, problems with the offset caused by gravity and friction arose, while keeping it made it impossible to estimate constant force, because the controller would need to have a stationary error.

In order to test the estimation scheme, the robot performed an assembly task, with a ATI Mini40 force/torque sensor used to get validation data. The force was estimated both with and without apriori information about the low contact torques.

Berger et al. presented a machine learning approach to estimate torque in robot manipulators in [21] (6 citations in Scopus and 5 in Web of Science, chosen because it had "machine learning approaches" as a keyword). This opens the possibility of estimating without any prior knowledge of the kinematics or mass distribution of the robots. All 105 sensors of the UR5, which are provided by the robot's realtime interface, were used in order to identify which sensors were most important for the estimation scheme. In addition to the information from the sensors, the relative time and the preconfigured torque were recorded. 10 different configurations of the torque were used to record training data, as well as virtual training sets that were interpolated by using Dynamic Mode Decomposition (DMD) [53]. After data acquisition, the authors generated a model consisting of the Phase-Feature Space (P-FS) and the Torque-Feature Space (T-FS) that are low-dimensional embeddings of the original high-dimensional data. To do this, and to find which sensors were most important, they looked at how much the preconfigured torque and the relative time influenced the sensors. It was found that the angle sensors were influenced greatly by the preconfigured torque, while current sensors were affected by the relative time. Sensors not influenced by the relative time were not relevant, and thus ignored for phase estimation, while for torque estimation the sensors not influenced by preconfigured torque were ignored.

In [54] (67 citations in Scopus and 57 in Web of Science) by Su et al. the goal was to estimate tactile properties and to detect manipulation events of a robot arm by using contact-based techniques, as well as classifying slip events. While most of the other articles focused on the arm itself, and thus its end-effector, this article focused on the finger forces of the robot, and therefore also introduced a grip force controller, in order to pick up objects without using too

much force on the object itself. BioTac sensors were used as biomimetic sensors in order to estimate forces, and to detect and classify slip events.

In order to estimate finger forces it is important to have a reliable estimation of tri-axial forces (F_x, F_y, F_z). Su et al. used the readings from the BioTac sensor to employ and evaluate four different methods of force estimation, all of which can be characterized as learning-based estimation techniques.

The robot used is a Barrett arm/and system with three fingers, all equipped with BioTac sensors. Each of these sensors consist of an array of 19 electrodes, which can be used to characterize tri-axial forces by looking at their changes of impedance.

The first approach used a weighted sum of the normal vectors of the electrodes, given by $(N_{x,i}, N_{y,i}, N_{z,i})$. By using the impedance changes E_i, and the scaling factors (S_x, S_y, S_z) which convert calculated contact forces into Newtons and are learned by using linear regression and ground truth data, the weights are given by

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \sum_{i=1}^{19} \begin{bmatrix} S_x E_i, N_{x,i} \\ S_y E_i, N_{y,i} \\ S_z E_i, N_{z,i} \end{bmatrix}. \tag{6.20}$$

Locally Weighted Projection Regression (LWPR), which was also used by Colomé et al. in [2], and regression with neural networks were introduced in order to further improve the quality of the force estimation. The second approach used LWPR, where N local linear models $\psi_k(x)$ were used to compute a weighted mean of the values of all local models in order to estimate the function value, given by

$$f(\boldsymbol{x}) = \frac{\sum_{k=1}^{N} w_k(\boldsymbol{x})\psi_k(\boldsymbol{x})}{\sum_{k=1}^{N} w_k(\boldsymbol{x})}, \tag{6.21}$$

where $w_k(x)$ represents the weights which, based on the distance between the estimation point and the function value, are used to determine the influence each local model $\psi_k(x)$ has on the function value. These weights are often modelled using a Gaussian distribution with $\boldsymbol{c}_k$ as the centers of the Gaussians and $\boldsymbol{D}$ as the distance metric, such that

$$w_k(\boldsymbol{x}) = \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_k)\boldsymbol{D}(\boldsymbol{x} - \boldsymbol{c}_k)). \tag{6.22}$$

The weights and the parameters of each local model were learned using locally weighted partial least squares regression.

Neural networks (NN) were used for the third approach, by using a single-hidden-layer consisting of 38 neurons. In the fourth approach a multi-layer NN, which consisted of three hidden layers in addition to input and output, was used in order to learn the mapping from BioTac electrode values to the finger forces. In this NN, each layer consisted of 10 neurons each. For both the single-hidden-layer and multi-layer NN approaches the function for the neurons were the hyperbolic tangent sigmoid transfer function given by

$$a = \frac{2}{1 + \exp(-2n)} - 1. \tag{6.23}$$

A linear transfer function was used for the activation of the output layer, and the NNs were trained with the error back-propagation and Levenberg-Marquardt optimization technique.

A grip controller was designed for testing the force estimation, where the robot arm was to grip and lift an object. Initially, the fingers of the robot are position controlled by using the estimated normal force F_z in order to close on an object until the force went above a certain threshold. Following this, the position controller was stopped and the grip force controller was employed, where the minimal required grip force was estimated using $F_t = \sqrt{F_x^2 + F_y^2}$ for the force tangential to the BioTac sensor, and the grip force F_z was controlled by $F_z = \frac{F_t}{\mu} + \text{safety margin}$, where μ was the current estimation of the friction coefficient.

A data set consisting of raw signals of the 19 electrodes was collected and divided into several data sets containing 30 seconds intervals of readings, which were used for the training and test sets for the NNs in order to evaluate the different force estimation methods, in addition to a validation set to prevent overfitting. Root Mean Squared Error (RMSE) of the force N and Standardized Mean Squared Error (SMSE) was used to evaluate the methods. The three learning-based methods all outperformed the analytical method. Even though a validation set was used, both the LWPR and single-hidden-layer approach overfitted the data, which means that they performed well on the full set containing data they had already been exposed to, but was lacking in performance on sets they had not been exposed to already. However, the 3-layer NN was able to avoid overfitting, as well as performing well on the test set. The best RMSE values for the 3-layer NN on the test set in x, y and z-direction were $0.43N$, $0.53N$ and $0.85N$ respectively, while the SMSE values were 0.08, 0.03 and 0.02. Thus, it can be concluded that the 3-layer NN approach achieved the best performance.

Gupta et al. used a nonlinear disturbance-observer based force estimation in order to estimate contact forces during haptic interactions in [55] (41 citations in Scopus and 28 in Web of Science). Initially, they started with the model of an n-link robot manipulator with external disturbance as in (2.26) with the disturbance given as $\boldsymbol{d}$. Then they defined an auxiliary variable vector

$$\boldsymbol{z} = \hat{\boldsymbol{d}} - \boldsymbol{p}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{6.24}$$

with $\hat{\boldsymbol{d}}$ as the disturbance estimates. The nonlinear function, $\boldsymbol{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ was defined as

$$\boldsymbol{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{p}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\mathrm{d}t}, \tag{6.25}$$

and $\dot{\hat{\boldsymbol{d}}}$ was defined as

$$\dot{\hat{\boldsymbol{d}}} = -\boldsymbol{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})(\hat{\boldsymbol{d}} - \boldsymbol{d}). \tag{6.26}$$

By differentiating (6.24), the error is found to be given by

$$\boldsymbol{e} = \boldsymbol{d} - \hat{\boldsymbol{d}}. \tag{6.27}$$

By assuming that $\dot{\boldsymbol{d}} = 0$ and differentiating (6.27), the authors got the following

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{d}} - \dot{\hat{\boldsymbol{d}}} \tag{6.28}$$

$$= -\boldsymbol{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{e}, \tag{6.29}$$

which was proven to be exponentially stable with the choice of $\boldsymbol{p}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = c\dot{\boldsymbol{q}}$, where c is a positive scalar. By defining a Lyapunov function candidate and differentiating it, it was shown that the rate of convergence is proportional to c.

A custom single degree of freedom haptic interface was used in order to test the disturbance-observer. Two tests were performed: one during free space interaction, and one during virtual wall interaction, which was simulated as a simple spring. Results showed that there was a slight time delay with the disturbance-observers tracking of contact forces, and there is a slight deviation between measured and estimated forces, with maximum near the peaks. The tests done using a virtual wall resulted in larger errors than during free space interaction. The disturbance-observer was then used for a disturbance-observer closed-loop controller, and results indicated that it performed well and could be

used in place of a force sensor. However, the observer was designed for constant disturbances, which may not translate to the real world, such that exponential stability would not necessarily be possible, and issues regarding stability might occur.

Publication year	Authors	Method	Sensor signals	Robot
1994	Hacksel, Salcudean	Angular velocity observer	Position, velocity	UBC maglev wrist [44]
2002	Simpson, Cook, Li	Model-based	Position, motor current	Hirata ARI350 SCARA Robot [56]
2007	Aksman, Carignan, Akin	Learning-based	Position, motor current	2 DOF pitch-roll manipulator [57]
2008	Katsura, Irie, Ohishi	Position-acceleration integrated disturbance observer	Position, acceleration	Rod type linear motor system [58]
2011	Van Damme, Beyl, Vanderborght, Grosu, Van Ham, Vanderniepen, Matthys, Lefeber	Recursive least-squares, generalized momentum observer	Position, velocity	Pleated Pneumatic Artificial Muscles (PPAM) actuated manipulator [46]
2011	Gupta, O'Malley	Disturbance observer	Position, velocity	One DOF custom built haptic device
2013	Colomé, Pardo, Alenyà, Torras	Learning-based, disturbance state observer	Position, (velocity, acceleration)	Barett WAM [2]
2015	Aviles, Alsaleh, Sobrevilla, Casals	Learning-based, neuro-vision-based	Tissue surface displacement,	Stäubli RX60B [59]
2015	Zhang, Ahmad, Liu	Harmonic drive compliance model, position measurements	Position	Experimental setup, 3 joints [49]
2015	Wahrburg, Morara, Cesari, Matthias, Ding	Generalized momentum observer, disturbance observer	Position, velocity, motor torque	ABB YuMi (simulated) [50]
2015	Stolt, Andreas	Based on control errors for the low-level joint control loops	Joint angles	ABB YuMi [51]
2015	Su, Hausman, Chebotar, Molchanov, Loeb, Sukhatme, Schaal	Learning-based	Force	Barrett arms and hands [54]
2016	Bargsten, Fernández, Kassahun	Learning-based	Position, velocity, acceleration	KUKA iiwa R820 [25]
2016	Berger, Grehl, Vogt, Jung, Amor	Learning-based, transfer entropy	Angle, velocity, current	UR5 [21]
2019	Li, Wang, Yang, Zhou	Friction torque model, non-linear least squares regression	Position, motor current	IRIC-I [60]
2019	Gold, Völz, Graichen	Generalized momentum observer	Motor current, position	Simulated [61]

Table 6.1: State-of-the-art estimation techniques

Chapter 7

Discussion

As shown in Chapter 6 and Table 6.1 there are several different state-of-the-art methods of estimating forces and torques in robot manipulators.

Several of these methods rely on the availability of an accurate analytical model of the robot, such as model-based observers, and when using motor current measurements and translating this to the joint torques. This requires information about the robot and motor dynamics of the specific robot used, which might not be easily available. In addition to this, the model will need to be changed, or calibrated, when used with different robots, or when changes are made to the robot, such as attaching different tools at the end-effector. In the case of lightweight robots, such as UR3 or Panda, even small deviations might cause a large percentage of error, and the model might need to be calibrated often.

As mentioned in Section 2.3 the robot will be affected by other forces acting upon it, such as forces caused by contact with the environment and frictional forces that appear in the joints and motor gears of the robot. Some estimation methods rely on having the values of the friction available. These methods can be well-suited for direct-drive mechanisms, as they have low friction because of the lack of gears. However, they are not as well-suited for industrial and lightweight manipulators which have gears and/or gearboxes.

Force and torque signals are widely used for force control of robots. Because of this, the estimation method must be able to estimate forces and torques when the robot is moving. Some of the methods presented assume almost-stationary situations, and experiments are performed on stationary robots. This results in the estimation not being available when the robot moves, which in turn makes

it unusable for force control during operations such as assembly.

Some of the more recent work on estimating forces and torques in robots have used learning-based methods. With these methods, the parameters of the modeled dynamics can be tuned by using adaption, or it can be used to avoid the need of analytical model of joint's friction. If applied to robots with internal sensors, such as the UR3, the method can be used to learn which sensors are relevant for the estimation, and the model can be used to accurately estimate forces and torques at runtime. However, some limitations might appear when using learning-based methods, such as that the configuration of the robot might have to be exactly the same way for the offline training and realtime estimation.

A comparison between an analytical approach and some learning-based methods, which was performed by Su et al. in [54] was discussed in Chapter 6. This showed that all three learning-based methods outperformed the analytical approach. One of the learning-based methods used in earlier research is Locally Weighted Projection Regression, which was used by Colome et al. in [2], as well as by Su et al. However, in both articles, neural networks were introduced in order to improve the quality of the estimators. With machine learning being a growing field of research, and the quality in performance of learning-based estimation techniques shown in the mentioned articles, approaches which make use of learning, and especially neural networks, would be promising methods to pursue.

In Chapter 4 the different robots currently available for the project was presented: Universal Robot's UR3 and UR5, and KUKA's KR16 and LBR iiwa, and Franka Emika Panda.

The KUKA Robot Sensor Interface (RSI) for the KR16 makes it possible to read the forces and torques at the end-effector using a force/torque sensor, and to read the currents supplied to the motors. However, it is unlikely that the forces exerted on the link side will be noticeable at the motor current side for any reasonable amount of external force. In addition to this, because of the 12ms interpolation cycle the update frequency of the interface is limited to 83Hz [62]. This limits the usefulness of the robot in force estimation experiments.

The UR3, UR5, KUKA LBR iiwa and Franka Emika Panda are all lightweight collaborative robots, which makes them more suited for force estimation experiments than the KR16. The UR3 provides a signal of the force exerted at the end-effector in terms of a wrench defined with respect to the base of the robot. The UR3 also provides joint torque measurements. In addition to this it is backdrivable, making it easy to control simply by moving the robot arm itself, as mentioned in Section 3.2. This also applies to LBR iiwa and Panda.

Using ur_modern_driver, or the new Universal_Robots_ROS_Driver, with

ROS makes it is possible to collect internal sensor data for the UR3 and UR5. Thus, it is possible to collect a large amount of training data to be used for a learning-based estimation method.

As both LBR iiwa and Panda have 7 DOF, they are more flexible compared to the other manipulators. Thus, they can be used for more complicated tasks. However, as discussed in Section 4.2.6 this extra DOF introduce the need for techniques to handle it. Therefore, using a 7 DOF manipulator will be a more complicated task than a traditional 6 DOF manipulator. A dexterity analysis was performed in [63] by Kuhlemann using KUKA's LBR iiwa and KR10 R900, where the latter is a 6 DOF manipulator. The purpose was to compare kinematics for 6 and 7 DOF, and to analyze the effects of the additional joint. LBR iiwa was simulated as a 6 DOF manipulator for direct comparison. The results showed that the additional DOF enhanced the dexterity by 16.8%, which was lower than what the authors expected. When comparing the highest average dexterity of the manipulators, KR10 R900 outperformed LBR iiwa by 7.6%, due to larger joint ranges.

As with the UR3, the Panda has a significant amount of internal sensors, with more than 100. Because of this, it is well suited for learning-based methods, where the amount of data is important. One of the methods in Chapter 6 used learning in order to figure out which sensor signals were most important for use in force/torque estimation. This was done on UR3, but with the amount of sensors the Panda offers, the manipulator would be a good choice for such an approach.

As for robot middleware, four alternatives have been discussed: no middleware, ROS, ROS 2 and Orocos.

Because the application should be designed to be general and work on several robot platforms, no middleware would not be a viable solution, as these libraries are usually designed for one specific robot.

One of the main issues with Orocos is the lack of documentation. Because of this, using Orocos as a middleware might be more challenging than some of the other options. In addition, the userbase is not nearly as large as it is for ROS.

The modular design of ROS makes it well suited for the application. Using this as middleware, it would be possible to design a package for the estimator which would work with several different robots. Packages for support of the different robots discussed in Chapter 4 already exists in ROS, and therefore this does not need to be created in order to make it possible to use the robots. In addition to this, packages for robot kinematics and dynamics, among other things, have already been developed, and as such, can be used to simplify the

work significantly.

As of now, ROS 2 is still under development, and is lacking in regards to packages and stability. Therefore, it would not be recommended to use as middleware as of yet. However, when ROS 2 has reached a stage where the software is more stable and it there are more packages available, it would be a possible middleware to use. The feature differences given in Table 5.1 in Section 5.3 make the ROS 2 a more future proof middleware than ROS.

Chapter 8

Further Work

The original plan for this thesis was to implement the different estimation techniques and experiment with them on a robot. Therefore, this will be a natural area to research for further work.

Of the robots discussed, all of the lightweight robots are viable options to use in experiments. However, due to the additional DOF of LBR iiwa and Panda, the best place to start might be with UR5 or UR3.

As for middleware, ROS, and subsequently ROS 2 when it is more developed, are the best solutions for the task. The multiple packages available make it a lot easier, as a lot of the groundwork is already done.

Chapter 9

Conclusion

In this report, several different state-of-the-art methods of estimating forces and torques were presented and discussed. Four robot manipulators were considered, and their applicability was examined. Some of the most common robot programming middlewares were discussed.

It was shown that two promising estimation techniques were observer-based and learning-based methods. Observer-based has been successfully used multiple times in previous research. Learning-based is a rather new approach, but it was shown to give promising results, and the fact that accurate models of the robots are not necessary makes it applicability for an application described as in Section 1.2.

Further, it was shown that the four lightweight robot manipulators, UR5, UR3, KUKA LBR iiwa, and Franka Emika Panda, are all possible to use. UR3, UR5 and Panda are well suited for a learning-based approach, where the amount of data is important, as they provide over 100 different sensors. However, both LBR iiwa and Panda are redundant manipulators, and techniques for handling their additional DOF must be used. Therefore, the UR3 would be the manipulator best suited for the application.

Bibliography

[1] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2006.

[2] A. Colomé, D. Pardo, G. Alenyà, and C. Torras. External force estimation during compliant robot manipulation. In *2013 IEEE International Conference on Robotics and Automation*, pages 3535–3540, May 2013.

[3] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer Publishing Company, Incorporated, 2nd edition, 2016.

[4] Olav Egeland and Jan Tommy Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2003.

[5] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[6] L. F. Baptista, M. A. Botto, and J. S. da Costa. Force control of rigid manipulators: a comparative study for non-rigid environments. In *ISIE '97 Proceeding of the IEEE International Symposium on Industrial Electronics*, pages 878–883 vol.3, July 1997.

[7] Xiao-Feng Liu, Hai-Quan Li, Yi-Jun Chen, and Guo-Ping Cai. Dynamics and control of space robot considering joint friction. *Acta Astronautica*, 111:1 – 18, 2015.

[8] Gill A. Pratt, Matthew M. Williamson, Peter Dillworth, Jerry Pratt, and Anne Wright. Stiffness isn't everything. In Oussama Khatib and J. Kenneth Salisbury, editors, *Experimental Robotics IV*, pages 253–262, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[9] KUKA KR 500 FORTEC, [online]. https://www.kuka.com/en-de/products/robot-systems/industrial-robots/kr-500-fortec. Accessed: 2019-12-11.

[10] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The kuka-dlr lightweight robot arm - a new reference platform for robotics research and manufacturing. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, June 2010.

[11] Alin Albu-Schäeffer, Sami Haddadin, Christian Ott, Andreas Stemmer, Thomas Wimböck, and Gerd Hirzinger. The dlr lightweight robot – design and control concepts for robots in human environments. *INDUSTRIAL ROBOT-AN INTERNATIONAL JOURNAL*, 34:376–385, 08 2007.

[12] H. Asada and K. Youcef-Toumi. Analysis and design of semi-direct-drive robot arms. In *1983 American Control Conference*, pages 757–766, June 1983.

[13] T. Ishida and A. Takanishi. A robot actuator development with high backdrivability. In *2006 IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–6, June 2006.

[14] Erico Guizzo. Startup says direct-drive motors are the future of robot actuators, [online]. https://spectrum.ieee.org/robotics/robotics-hardware/startup-says-directdrive-motors-are-the-future-of-robot-actuators, 2019.

[15] H. Makino. Development of the SCARA. 26:5–8, 02 2014.

[16] J. Angeles, A. Morozov, and O. Navarro. A novel manipulator architecture for the production of scara motions. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2370–2375 vol.3, April 2000.

[17] M. M. Bridges and D. M. Dawson. Redesign of robust controllers for rigid-link flexible-joint robotic manipulators actuated with harmonic drive gearing. *IEE Proceedings - Control Theory and Applications*, 142(5):508–514, Sep. 1995.

[18] M. Hashimoto, Y. Kiyosawa, and R. P. Paul. A torque sensing technique for robots with harmonic drives. *IEEE Transactions on Robotics and Automation*, 9(1):108–116, Feb 1993.

[19] About universal robots, [online]. `https://www.universal-robots.com/about-universal-robots/`. Accessed: 2019-11-02.

[20] UNIVERSAL ROBOT UR5e, [online]. `https://www.universal-robots.com/products/ur5-robot/`. Accessed: 2019-10-02.

[21] E. Berger, S. Grehl, D. Vogt, B. Jung, and H. B. Amor. Experience-based torque estimation for an industrial robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 144–149, May 2016.

[22] KUKA.robotsensorinterface 2.3, 2009.

[23] LBR iiwa, LBR iiwa 7 r800, LBR iiwa 14 r820, specification, 2015.

[24] Günter Schreiber, Andreas Stemmer, and Rainer Bischoff. The fast research interface for the kuka lightweight robot. 05 2010.

[25] V. Bargsten, J. d. Gea Fernandez, and Y. Kassahun. Experimental robot inverse dynamics identification using classical and machine learning techniques. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–6, June 2016.

[26]

[27] Franka control interface documentation, [online]. `https://frankaemika.github.io/docs/index.html`, 2017. Accessed: 2020-06-28.

[28] ROS Introduction, [online]. `http://wiki.ros.org/ROS/Introduction`. Accessed: 2019-11-02.

[29] libfranka, [online]. `https://frankaemika.github.io/docs/libfranka.html`, 2017.

[30]

[31] Ros topics, [online]. `http://wiki.ros.org/Topics`. Accessed: 2019-11-02.

[32] Ros services, [online]. `http://wiki.ros.org/Services`. Accessed: 2019-11-02.

[33] Universal_Robots_ROS_Driver, [online]. https://github.com/UniversalRobots/Universal_Robots_ROS_Driver. Accessed: 2019-12-09.

[34] Brian Gerkey. Why ROS 2? [online]. https://answers.ros.org/question/134551/why-is-ros-not-real-time/. Accessed: 2019-12-16.

[35] Dirk Thomas. Changes between ROS 1 and ROS 2, [online]. http://design.ros2.org/articles/changes.html. Accessed: 2019-12-16.

[36] Vanessa Mazzari. ROS vs ROS2, [online].

[37] Dirk Thomas. Changes between ROS 1 and ROS 2, [online]. http://design.ros2.org/articles/changes.html/. Accessed: 2020-06-27.

[38] Orocos project history, [online]. https://orocos.org/content/history. Accessed: 2020-06-27.

[39] The orocos real-time toolkit, [online]. https://orocos.org/rtt.

[40] Orocos component library, [online]. https://orocos.org/ocl.

[41] Orocos kinematics and dynamics, [online]. https://orocos.org/kdl.

[42] Klaas Gadeyne. BFL: Bayesian Filtering Library. http://www.orocos.org/bfl, 2001. Accessed: 2020-05-12.

[43] Tully Foote. ROS community metrics report, [online]. http://download.ros.org/downloads/metrics/metrics-report-2019-07.pdf, 2019. Accessed: 2020-06-19.

[44] P. J. Hacksel and S. E. Salcudean. Estimation of environment forces and rigid-body velocities using observers. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 931–936 vol.2, May 1994.

[45] S. Salcudean. A globally convergent angular velocity observer for rigid body motion. *IEEE Transactions on Automatic Control*, 36(12):1493–1497, Dec 1991.

[46] M. Van Damme, P. Beyl, B. Vanderborght, V. Grosu, R. Van Ham, I. Vanderniepen, A. Matthys, and D. Lefeber. Estimating robot end-effector force from noisy actuator torque measurements. In *2011 IEEE International Conference on Robotics and Automation*, pages 1108–1113, May 2011.

[47] Stefan Klanke, Sethu Vijayakumar, and Stefan Schaal. A library for locally weighted projection regression. *Journal of Machine Learning Research*, 9:623–626, 04 2008.

[48] Stefan Klanke, Sethu Vijayakumar, and Stefan Schaal. A library for locally weighted projection regression. `http://jmlr.csail.mit.edu/papers/v9/klanke08a.html`.

[49] H. Zhang, S. Ahmad, and G. Liu. Torque estimation for robotic joint with harmonic drive transmission based on position measurements. *IEEE Transactions on Robotics*, 31(2):322–330, April 2015.

[50] A. Wahrburg, E. Morara, G. Cesari, B. Matthias, and H. Ding. Cartesian contact force estimation for robotic manipulators using kalman filters and the generalized momentum. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1230–1235, Aug 2015.

[51] Andreas Stolt. *On Robotic Assembly using Contact Force Control and Estimation*. PhD thesis, Lund University, 2015.

[52] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson. Force controlled robotic assembly without a force sensor. In *2012 IEEE International Conference on Robotics and Automation*, pages 1538–1543, May 2012.

[53] M. R Jovanovi, P. J Schmid, and J. W Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2), 2014.

[54] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal. Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 297–303, 2015.

[55] A. Gupta and M. K. O'Malley. Disturbance-observer-based force estimation for haptic feedback. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 133(1), 2011.

[56] John W.L Simpson, Chris D Cook, and Zheng Li. Sensorless force estimation for robots with friction. 05 2002.

[57] L. M. Aksman, C. R. Carignan, and D. L. Akin. Force estimation based compliance control of harmonically driven manipulators. In *Proceedings*

2007 IEEE International Conference on Robotics and Automation, pages 4208–4213, April 2007.

[58] S. Katsura, K. Irie, and K. Ohishi. Wideband force control by position-acceleration integrated disturbance observer. *IEEE Transactions on Industrial Electronics*, 55(4):1699–1706, April 2008.

[59] A. I. Aviles, S. Alsaleh, P. Sobrevilla, and A. Casals. Sensorless force estimation using a neuro-vision-based approach for robotic-assisted surgery. In *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 86–89, April 2015.

[60] X. Li, Y. Wang, Z. Yang, and H. Zhou. End-effector force estimation for robotic manipulators from motor current measurements. In *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 744–749, 2019.

[61] Tobias Gold, Andreas Völz, and Knut Graichen. External torque estimation for an industrial robot arm using joint torsion and motor current measurements. In *Joint Conference 8th IFAC Symposium on Mechatronic Systems (MECHATRONICS) and 11th IFAC Symposium on Nonlinear Control Systems (NOLCOS)*, pages 879–884, Vienna (Austria), 2019.

[62] Ivar Eriksen. Setup and interfacing of a kuka robotics lab. December 2017.

[63] I. Kuhlemann, P. Jauer, F. Ernst, and A. Schweikard. Robots with seven degrees of freedom: Is the additional dof worth it? In *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, pages 80–84, 2016.