MASTER

Modelling and feedback control of a pneumatic planar soft robotic actuator

Berkers, H.W.M.

Award date:
2019

Modelling and Feedback Control of a Pneumatic Planar Soft Robotic Actuator

H.W.M. Berkers
0803490

Supervisors:
ir. B.J. Caasenbrood
dr. A.Y. Pogromsky
prof.dr. H. Nijmeijer

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics & Control

Eindhoven, August 5, 2019
DC 2019.064

Abstract

Soft robots are robots made from compliant materials, and often inspired by nature. The emerging field of soft robotics can solve issues that are present in rigid robotics: soft robots are safe to operate around humans, and they are flexible and adaptable. As a drawback, sensing for soft robotics is much more complicated and the dynamical behaviour is highly non-linear.

For this research project, a planar soft robotic actuator is used that is driven via two diaphragm air pumps. A test setup has been built that employs a camera system and inertial measurement unit to obtain position and orientation data of this actuator. Using a constant curvature approach, the full position and orientation of the actuator can be estimated with this sensor data.

A lumped mass dynamical model of a 2D soft robotic actuator has been adapted from a 3D model. Damping and stiffness parameters are estimated from measurements. Due to a poor fit of this model to experimental data, a creep model is added to obtain better accuracy. A pump model is developed to model the pump dynamics.

With the test setup the position and orientation of the actuator are controlled with linear feedback control techniques and feedforward. For simple setpoints, these controllers lead to good tracking results. Tracking performance becomes worse for more complex setpoints. The same control techniques used in the experiments are then also applied in simulations for a further validation of the model. For simple setpoints there is a decent match between the model and experiments. More complex setpoints give rise to bigger discrepancies between the simulations and experiments.

Contents

Nomenclature

Notation	Meaning
$\boldsymbol{X}$	Matrix
$\boldsymbol{x}$	Vector
x	Scalar

Symbol	Description
a	acceleration
$\boldsymbol{A}$	Homography of camera mapping
c	Scaling factor damping and stiffness
$\boldsymbol{C}$	Coriolis matrix
$e, \boldsymbol{e}$	Error
f	Focal length
F	Force
$\boldsymbol{g}$	Gravitational acceleration vector
$\mathbf{H}$	Homogeneous transformation matrix
I	Integral term of the MM filter
$\boldsymbol{I}$	Identity matrix
J	Inertia
k	Stiffness function
K	Gain
$\boldsymbol{K}$	Camera intrinsic matrix
$\boldsymbol{k}$	Configuration states
l	Pixel size
ℓ	Actuator backbone length
M	Rotational spring momentum
$\boldsymbol{M}$	Mass matrix
$\boldsymbol{N}$	Input-independent force vector
$\boldsymbol{P}$	Point
$\boldsymbol{p}$	Bellow pressure
Q	Generalized forces
$\boldsymbol{q}$	Actuator states & gen. coordinates
r	Scalar distance
$\mathbf{R}$	Rotation matrix
$\boldsymbol{r}$	Position vector
$\mathcal{R}$	Coordinate frame
s	Scaling factor camera mapping
t	Time
T	Kinetic energy
$\boldsymbol{t}$	Translation vector
U	Potential
u, v	Pixel coordinates
x	Task space coordinate in x-direction
X	x-location measured with the Pixy2
$\boldsymbol{x}$	Task space coordinates
Y	y-location measured with the Pixy2
y	Task space coordinate in y-direction
$\boldsymbol{z}$	Creeping strain

Symbol	Description
α, β	Ellipse parameters
γ	Skewness parameter
δ	Logarithmic decrement
ζ	Damping ratio
κ	Actuator curvature
λ	Creep parameter
μ, ν	Pixel coordinates
σ	Scale factor in image axis
$\boldsymbol{\tau}$	Input-dependent force vector
φ	Orientation of actuator head
$\hat{\varphi}$	Estimated actuator orientation
ψ	Skewness angle
ω	Rotational velocity

Index	Superscript and subscript description
0	Initial condition or origin
a	Active
acc	Accelerometer data
b	Actuator base frame
B	Bellow
dmp	Damping
exp	Experiment
gyr	Gyroscope data
h	Actuator head frame
i	Integral
i	Image frame
k	Timestep
MM	Mahony-Madgwick
p	Proportional
p	Passive
ref	Reference signal
S	Actuator center of mass
sim	Simulation
spr	Spring
w	World frame
x	x-direction
y	y-direction
μ, ν	Pixel coordinates

Abbreviation	Meaning
IMU	Inertial Measurement Unit
MM	Mahony-Madgwick
PWM	Pulse-Width Modulation
SRM	Soft Robot Manipulator

Chapter 1

Introduction

This thesis will present simulations and experiments with a soft robotic actuator. To begin with, a general introduction to the field of soft robotics will be given. We will introduce modelling methods for soft robotics that can be used for simulations. Sensing methods will be described to enable position and orientation sensing for experiments and control purposes. A small section will discuss the manufacturing aspects of a soft robotic actuator. Finally, the research outline of this thesis will be presented.

1.1 Background

1.1.1 Introduction to soft robotics

Traditional rigid robots are highly developed and prevalent in commercial applications, such as in manufacturing automation. They are accurate, and excel in performing repetitive tasks. However they also have their drawbacks. Due to their usually rigid and heavy construction, it is not safe to operate these robots in the presence of humans. Besides, their degrees of freedom are limited, so it is difficult to avoid obstacles.

The emerging field of soft robotics can solve these issues. Soft robots take inspiration from nature, and are often based on trunks or tentacles. They are made of compliant materials and have muscle-like actuation. By contracting or expanding parts of the robot, motion can be realized. Due to the compliant construction, operation is safe around humans which opens up a whole lot of new possibilities in the cooperation between humans and robots. Because of the continuously deformable structure, the number of degrees of freedom (DOF) is theoretically infinite. As a result soft robots are flexible and adaptable, and thus obstacle avoidance becomes much easier. Often, soft robots consist out of multiple independently actuated segments which greatly enhances the flexibility and adaptability. Nonetheless, soft robots have disadvantages as well. The flexible structure makes it difficult to model and control soft robots. Non-linear material behaviour makes it difficult to obtain model parameters. Often these robots are under-actuated [1] and low pressure actuation can cause inability to compensate for gravitational loading [2].

Soft robots can be actuated in several ways. The most widely used actuation method is pneumatically [3]. Pneumatic soft robots usually consist out of multiple pneumatic chambers. These chambers can be pressurized with pumps independently, such that bending is possible in any direction. Another common actuation method is the application of tension cables. Multiple cables

are routed along the actuator, and pulling these cables will also result in curvature of the actuator. Cables have the disadvantage that they require more moving parts and they are more difficult to implement. In this report the focus will be on pneumatically actuated soft robots.

Most soft robots developed so far have been experimental or a showcase. Commercial applications are rare, and when they exist, it is mostly in the form of grippers as seen in Figure 1.1a. Well-known examples of pneumatic, multi-section, soft robotic manipulators are the Bionic Handling Assistant by Festo [4] in Figure 1.1b and the OctArm Continuum Manipulator [5] in Figure 1.1c.

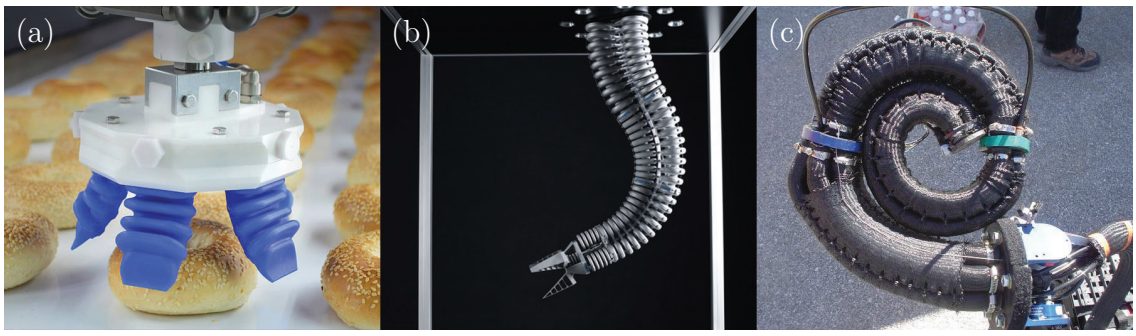


Figure 1.1: a) Soft robot gripper (Source: Soft Robotics Inc.). b) Festo's Bionic Handling Assistant, a multi-section bellows-actuated continuum manipulator. (Source: Festo). c) OctArm V, a multi-section continuum manipulator using air muscle actuators [5].

Soft robots go by a variety of names, such as soft robot manipulator (SRM) and soft robot actuator. These names are used interchangeably in this report. Also, in the remainder of this report, soft robots will refer to pneumatically actuated soft robots, instead of other actuation methods.

1.1.2 Modelling and control for soft robotics

The dynamics of a soft robot are highly non-linear, due to geometry and material properties. This makes modelling these robots a complicated task. Besides, there is a large variety in design and actuation of SRMs that makes it difficult to formulate standard kinematics and dynamics applicable to a broad range of soft robotics.

The most elaborate models to accurately determine end effector position, or any other point on a soft robot, combine forces and moments that are applied to the soft robot by both the applied pressure as well as the environment. These models are geometrically exact [6], but they do not result in closed-form kinematics. They are numerically expensive and therefore not suitable to apply in real-time, model-based control [7].

A conventional approach to simplify the kinematics is to assume that the arc of the SRM has a constant curvature, i.e. when the soft robot bends its backbone has a constant radius. Generally, SRMs do not exactly have a constant curvature, but it is a good approximation [8]. The constant curvature forward kinematics can be obtained with a variety of methods such as Denavit-Hartenberg parameters, Frenet-Serret frames and integral representation. However, all these methods ultimately achieve the same result [7]. A commonly utilized frame convention is shown in Figure 1.2. The length and curvature of the actuator, together with the angle of the curvature plane, can express the position of the end-effector of a continuum robot in Cartesian coordinates.

A common approach to derive the equations of motion is to utilize the Euler-Lagrange equations [2], [9], [10]. The derivation of the forces and energies can be made in different ways. The work by

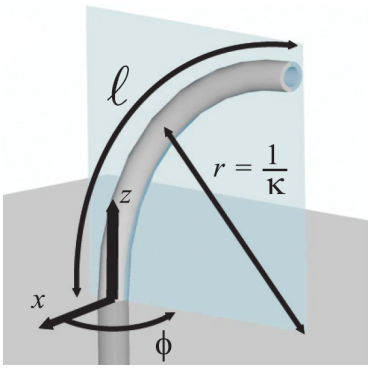


Figure 1.2: Commonly used frame convention in literature. With ℓ the length of the actuator, κ the curvature of the actuator, and ϕ the angle of the curvature plane [7].

[9] divides the actuator body into slices of infinitesimal thickness. Each slice has its own energy that can be calculated with mass and stiffness matrices, and integrating over the length of the actuator results in the total energy.

A classical modelling concept in rigid robots is the usage of lumped models, i.e. the robot is decomposed into discrete elements such as point masses, springs, and dampers. Lumped models are relatively simple and computationally cheap. Another advantage is that it is relatively easy to apply advanced non-linear control methods developed for rigid lumped models, such as inverse dynamics control. The work by [10] presents a lumped model for a bellows-actuated continuum robot, with each bellow represented by a single spring and damper.

When a model has been derived from first principles, model parameters have to be identified. Geometric parameters are easy to identify from CAD data, and masses can simply be measured. Acquiring the stiffness and damping parameters is much more difficult. It is possible to approximate them with FEM simulations [9], or with curve fitting techniques on experimental data [2].

Control of pneumatic soft robots has mostly been done using model-free open-loop control [2]. This open-loop control is often in the form of valve sequencing. A valve is turned on until a certain pressure level inside the actuator is reached, measured with a pressure sensor. The valve is then turned off to either hold or relieve the pressure [2], [3]. Unless a very accurate model is used that translates pressure to position, which would be difficult to obtain and computationally expensive, no proper information on end-effector position is known with this open-loop strategy.

This open-loop control remains popular, as it is much easier to implement than closed-loop control, and, for some applications (such as grippers), it is sufficient. But to achieve accurate position tracking, feedback control is necessary. Model-based control for soft continuum robots is a very recent research topic [2]. While it is possible to stabilize and control a system with simple PID control [11], non-linear model-based control can lead to better performance. A variety of modelling approaches can be used to obtain the models required for these model-based controllers, as described above. Examples of model-based control are computed-torque control and passivity-based adaptive control, as demonstrated by [9] on a soft robotic system in simulations. Computed-torque control employs state-feedback linearisation such that linear control techniques can be implemented. Passivity-based adaptive control aims to make the closed-loops system passive, and it has as an advantage that is is robust with respect to system parameter uncertainties. These controllers showed promising results in numerical simulations, but there is no experimental validation. Feedback linearisation control has been realized experimentally on a bellows-actuated soft robot by [12] with good results. Others have used dynamic models together with trajectory-optimization algorithms to find locally-optimal manoeuvres, and iterative learning control is applied to execute

these manoeuvres [2]. Nevertheless, sensors are required to implement closed-loop controllers. Unlike rigid robotics, sensing in soft robotics is a difficult task. The next section gives an overview on possible sensing methods.

1.1.3 Sensing for soft robotics

One of the main difficulties in soft robotics is position and orientation sensing for closed loop control. In case of a rigid robot arm, it is possible to simply measure angular and linear joint displacements, and combined with the constant dimensions of the arm, the position of the entire arm and end effector is known. This does not extend to the compliant, flexible structure of soft robots, due to the lack of (discrete) joints and variable length. Therefore, a number of possible position and orientation sensing methods for soft robots are described below. Position sensing for soft robotics can be divided into five groups [13]: external localization, inertial measurement, rigid sensors and curvature sensors.

External localization

External localization systems generally involve multiple sensors placed externally, at a fixed place, to create a reference frame. These sensors measure the location of some kind of tracker, and by combining data from multiple sensors the position with respect to the reference frame is known. A prevalent external localization method is the application of camera systems. These systems have been used in closed loop control for soft robots by tracking reflective or coloured markers placed at the actuator [11], [14]. A downside is that good visibility is required. Obstacles or bad lighting can disturb the measurements. Also, the camera can detect undesired objects, disturbing the measurements as well.

An external localization system that resolves the visibility requirement would be an electromagnetic tracking system. A electromagnetic field is generated by an external source, and an electromagnetic sensor is placed on the SRM. The technique is commonplace in the medical field [15] and it has also been used for feedback control in a soft actuator [16]. However, a drawback of this sensing method is that ferromagnetic materials can distort the magnetic field which may disturb position measurements.

Inertial measurement

Inertial measurement units (IMUs) employ accelerometers, gyroscopes and magnetometers to measure linear acceleration, rotational velocity and the magnetic field, respectively. They are often installed in aircraft to measure orientation, by combining the accelerometer, gyroscope and magnetometer data. The gyroscope data has to be integrated, which leads to drift, so sensor fusion algorithms are regularly introduced to improve the orientation data.

Rigid sensors

It is possible to attach soft robotic actuators to a frame consisting of rigid links and joints, and to use measure the position and orientation of the soft robot with encoders in the joints [17], [18]. One could argue if these system can be classified as soft robotic systems, since they have the same limitations as rigid robotic systems that we are trying to avoid by using soft, compliant robotics.

Another form of rigid sensing is the utilization of cables under tension. Multiple cables can be guided along the manipulator, and the displacement of these cables can be measured with wire cable potentiometers, which is then translated to SRM position [12]. It is not preferred to place

these cables on the outside of the SRM, since hitting an obstacle would disturb the cables and therefore the position measurement. Besides, having these cables in place it would be beneficial to also use them to drive the manipulator, which actually is a common method of actuation of SRMs [3]. Attaching multiple sections of manipulators to each other, using these cable sensors, results in a cumbersome soft robot since one potentiometer per wire is required.

Curvature sensors

Most sensors previously discussed are able to measure the position and orientation of a single point of the manipulator, or multiple points depending on the amount of installed sensors. But since the manipulators considered here consist of bending segments, it could also be interesting to measure the curvature in a segment. Curvature sensors exists in different forms: they either measure curvature directly, or they measure the change in length of an actuation channel and combining data from multiple sensors results in curvature.

Pure curvature sensors often consist of elastomers with an embedded material whose electrical resistance changes due to strain [19]. The biggest advantage of a sensor like this is that they are often completely soft, which fits the spirit of soft robotics. They can be embedded in the body of a SRM. A problem with pure curvature sensors is that often they can only measure curvature, and not change in length. This is problematic since the length of SRMs generally changes when actuated. Using a sensor that measures the change in length of the actuation channels resolves this issue. A bigger problem is that they can only measure the curvature in one direction.

Optical curvature sensors exist in the form of fibre optics cables with a light transmitter and receiver on one end, and a reflector on the other end of the cable. The light intensity measured by the receiver changes due to displacement of the reflector, and therefore can be related to channel length. Combining three of these sensors gives curvature data [20]. Another similar method would be to use a stretchable optical waveguide, with a reflective layer that does not stretch. Instead, small cracks occur in the reflective layer that allow light to escape the waveguide, leading to a drop in light intensity [21]. This can again be related to channel length. An optical curvature sensor like this has been demonstrated in a soft robotic end-effector resembling a human hand [22].

Another interesting sensing method is the use of inductance. A wire coil can be wound around the bellows of a SRM. When the SRM bends, the orientation of the coils changes, and therefore the self inductance of the coils as well. This change in inductance can be translated to the curvature of a SRM [23], [24]. By splitting the coils into multiple circuits, the curvature measurements become more accurate if the curvature is non-uniform [23].

1.1.4 Manufacturing of soft robots

Soft robots are made of flexible and/or stretchable materials, often an elastomer. Elastomers are visco-elastic polymers, with a low Young's modulus and able to sustain a large amount of strain. Relatively low pressures are required to actuate elastomer soft robots. The manufacturing of soft robots has another advantage over rigid robots: no accurate and expensive parts are needed. Making prototypes is cheap and easy due to additive manufacturing. Either the actuator itself can directly be 3D printed [4], or moulds can be printed in which the actuator is cast [9].

So far, the materials of soft robots have been simple, as experimental soft robots are often made with commercially available materials. It is expected that material science will play a major role in the development of the soft robotics field. For example, the development of composite soft materials is promising. These materials can lead to hard-soft materials, combining the advantages of both hard and soft robots [25]. Another possibility is the use of tissue engineering and synthetic

biology to create biohybrid soft robots, which can reduce the weight of SRMs [26].

1.2 Thesis outline and contribution

This thesis is an extension of the work by Caasenbrood et al. [9], which presents a dynamical model and control simulations for a pneumatic, three-bellows soft robot, as well as experimental pressure-based control. With pressure control it is extremely difficult to achieve accurate position and orientation tracking, which is desired in most robotic applications. The model presented in [9] is computationally expensive, and has not been subjected to experimental validation. This thesis tries to overcome these problems. The goal of this thesis is as follows. We want to find a modelling approach that is computationally inexpensive, and we want to test the model accuracy using an experimental setup. This setup should be able to measure the position and orientation of a soft robotic actuator. With this experimental setup we also want to investigate feedback control of a soft robotic actuator using pneumatic actuation.

The organisation of this thesis is as follows. Chapter 2 presents an experimental setup that will be used for position and orientation feedback control. With this setup it is also possible to experimentally validate dynamical models. The main problem regarding this setup lies in position and orientation sensing. Section 1.1.3 outlines a plethora of possible sensing methods. Most of these sensors either have many disadvantages, or the technology is quite immature. One sensing method that seems to be appealing is inertial measurement. IMUs are prevalent in commercial applications and literature. In order to make position and orientation sensing manageable, we limit ourselves two planar motion with the two-bellows planar pneumatic actuator in Figure 1.3, which will be actuated with air pumps. Planar motion should be well suited for camera-based position sensing. Thus, by using only one camera and an IMU, we can obtain position and orientation data of the soft robot. The calibration of the sensors will be discussed in detail in Chapter 2 as well. The actuator will be manufactured using additive manufacturing, as this is the easiest and least restrictive method. No research on novel materials will be done, as that is outside the scope for this project.

Figure 1.3: Picture of the two-bellows soft robotic actuator.

Next, in Chapter 3 the kinematics and dynamics for the planar soft robot will be formulated. The kinematics of soft robots has been studied extensively, and this can be borrowed from earlier work [7]. We are interested in formulating a computationally inexpensive model that enables real time simulation, which is valuable for future application, for example in model-based control. Hence, this thesis will also look into a lumped mass model, generally based on the model by [10]. Model parameters are estimated from measurements, and the estimation is improved with optimization techniques. Simulation results with this model will be compared to experimental results. A visco-elastic creep model is added to improve the model accuracy.

Finally, Chapter 4 describes how the experimental setup is utilized for feedback position and orientation control with the IMU and camera data. To improve performance, feedforward is introduced. The experimental results will be compared to the dynamical model, with an added pump model to facilitate actuation in the simulations as well. The control techniques are limited to linear proportional, integral and derivative controllers, and there is no intricate tuning of the controller performance. The controllers are merely implemented to gain a better understanding of the actuator behaviour.

To summarize, this thesis comprises the following:

- Description of an experimental setup containing an IMU and camera system
- IMU and camera calibration
- Formulation of the kinematics and a lumped mass dynamical model
- Parameter estimation
- Model validation with simulations and experimental data
- Feedback control using the experimental setup and in simulations

Chapter 2

Experimental setup

Before delving into feedback control of a soft robotic actuator, the actuator itself and the experimental setup are introduced. The actuator is mounted in a frame with sensors to obtain position and orientation data. The sensors have to be calibrated to acquire accurate data.

2.1 Actuator design and test setup

The actuator presented in Figure 1.3 is a pneumatic planarly actuated soft robot, consisting of two bellows connected to each other in a parallel position with mounting flanges at the top and bottom. One end of a bellow has a hole to connect to pumps, while the other end is closed off. This allows the bellows to be pressurized, and due to the geometry and flexible material the bellows can extend. Due to the parallel positioning of the bellows and an individual pump per bellow, the whole actuator can change in length and orientation. Figure 2.1 shows a section view of the actuator and its dimensions. The actuator has been designed with CAD software. To study the mobility and behaviour of the actuator, during the design process a FEM analysis has been done in Abaqus to see how the actuator deforms when pressurised. The material in the FEM analysis is modelled with a Yeoh strain energy potential. Because the Yeoh material properties of the actual material that will be used are unknown, the material properties of a silicone rubber are taken instead. The Yeoh coefficients implemented in the simulation are $C10 = 0.11$ and $C20 = 0.02$ [27]. Figure 2.2 shows the deformation results when only the bellow on the right side is pressurized. As expected, the right bellow extends, but because it is connected to the unpressurised left bellow, the actuator stands at an angle. Because of the incorrect material parameters, a quantitative analysis of the FEM results is not possible. The results are still useful, as the curvature induced by the pressure is almost constant as shown by the red circular arc. This constant curvature will be exploited to define the kinematics in Section 3.1.

The soft robot has been manufactured with a selective laser sintering (SLS) machine, a method of additive manufacturing. The material of the actuator is a rubber-like thermoplastic elastomer. Two identical rigid connectors are mounted to both ends of the actuator (the green parts in Figures 2.3). One side of a connector contains two nipples which protrude into the bellow holes. An air channel connects these nipples to holes in the side of the connector where the pump hoses are attached.

An experimental setup has been built to enable experiments with the planar soft robot. Figure 2.3 shows a picture of the setup. The base of this setup consists of a uniform triangular prism,

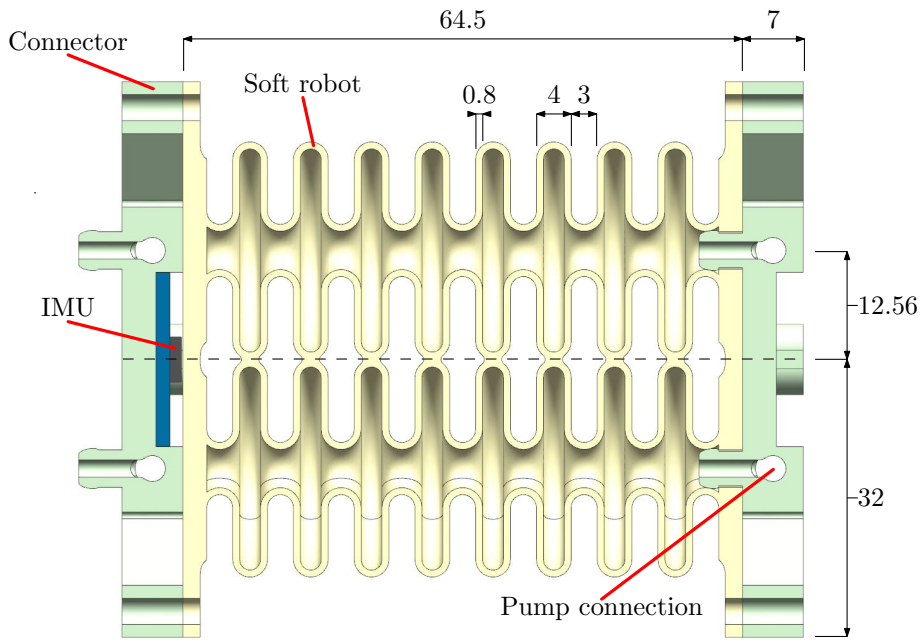


Figure 2.1: Sideways section view of the actuator, showing the bellow structure. Dimensions are in millimetres. The dashed line indicates the symmetry plane of the soft robot and connectors. The right side is mounted to the board in Figure 2.3.

constructed from square aluminium profiles. At the top, a clear plexiglass board is installed on which the actuator can be mounted. The pumps used for actuation are pneumatic diaphragm pumps, which are connected to 12 V DC motors via a housing. The pumps can reach a pressure of up to 80 kPa. Two of these motor-pump configurations can pressurize the bellows of the actuator independently. The pumps are connected to the actuator via small silicone hoses. Between the pump and bellow a small orifice is installed to ensure some leakage in the system. This leakage is beneficial when the bellows need to deflate.

The employed position sensing camera is the Pixy2 camera [28]. The Pixy2 is a low-cost vision system that operates a color-based algorithm to detect objects. It is possible to teach multiple object to the Pixy2, and it can track multiple objects simultaneously. It calculates the hue and saturation of each pixel which is then sent to the object detection algorithm as input. This makes

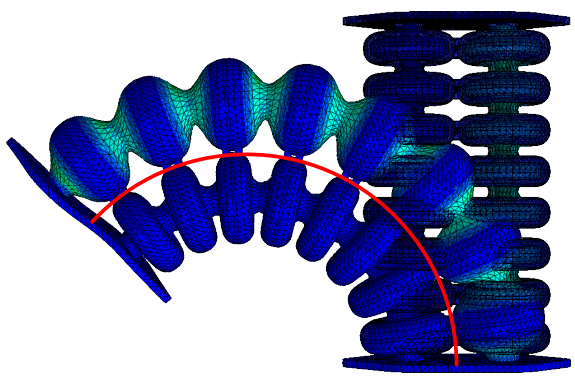


Figure 2.2: Abaqus FEM results when applying a pressure of 40 kPa to the right channel, showing both the deformed and undeformed state. The red line is a circular segment, which shows that the actuator curvature is near constant.

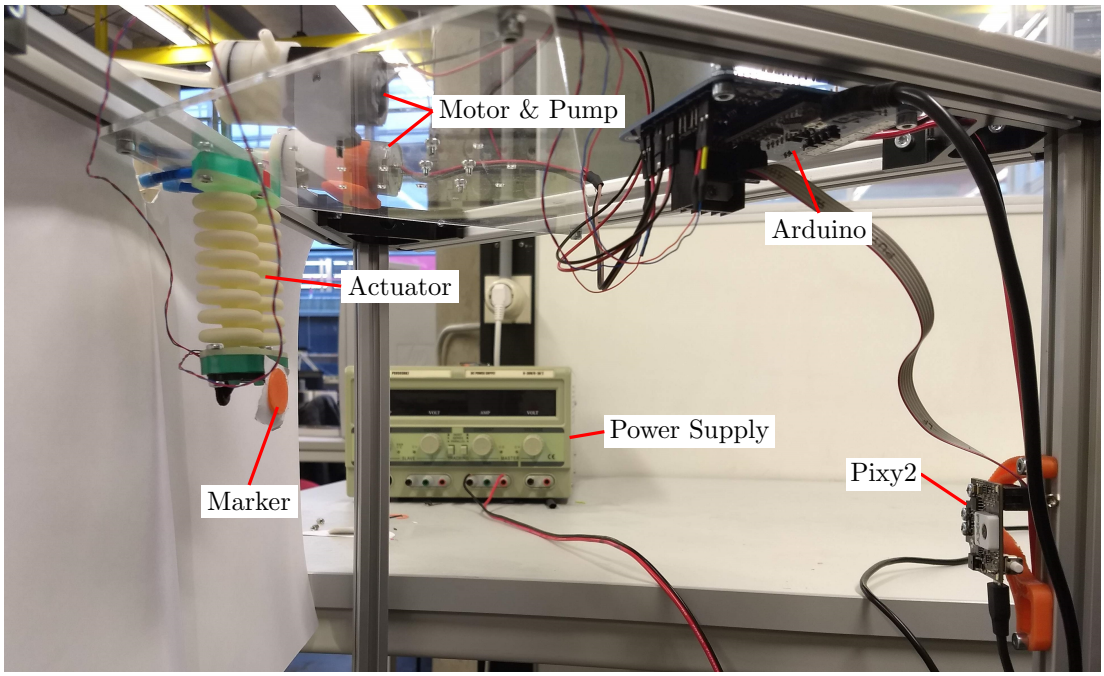


Figure 2.3: Experimental setup showcasing the different components.

the object detection robust with respect to lighting and exposure, since the hue is independent from lightning and exposure values. The Pixy2 has a resolution of 316×208 pixels, and a framerate of 60 frames per second. Object tracking is done real-time and the sample rate is 60 Hz. The Pixy2 communicates with the microcontroller via UART protocol. A circular orange marker is placed at the actuator head, and this marker is trained to the Pixy2 as the object to be tracked. The Pixy2 outputs the location of this marker in pixels.

Furthermore, we employ an inertial measruement unit (IMU), the BNO055 sensor [29]. The BNO055 is an integrated sensor chip containing a triaxial gyroscope, triaxial gyroscope, triaxial magnetometer and an embedded microcontroller. It has a maximum data rate of 100 Hz. The gyroscope can measure a rotational velocity of up to 2000 °/s, with a resolution of 0.0305 °/s. The accelerometer has a range of $+/- \ 4g$, with a resolution of $0.019\,16\,\mathrm{m/s^2}$. The BNO055 has an on-board data fusion algorithm that is able to fuse the data of all three on-board senors to get absolute or relative orientation data. In the remainder of this thesis, the term IMU will be used to refer to the BNO055.

A printed circuit board (PCB) with an Arduino Nano ATmega328P microcontroller is employed to control and read the data from the pumps and sensors, and to send sensor data to a PC. A DC power supply provides 12 V to this PCB. The Arduino Nano requires a pulse-width modulation (PWM) signal to control the pumps in the range 0 V to 12 V.

2.2 Sensor calibration

In order to use the IMU and Pixy2 as sensors for feedback control of the actuator position and orientation, they first have to be calibrated. For the IMU this calibration can be accomplished with a relatively simple filter. The Pixy2 calibration is more extensive and requires a mapping to transform image data into world coordinates that correspond to the actuator position in the real world.

2.2.1 IMU calibration

The IMU has an on-board sensor fusion algorithm that fuses the drift-prone gyroscope data and drift-free, but noise-sensitive, accelerometer data to obtain an orientation estimate that is better than using just the accelerometer data or just the gyroscope data. However, the sensor fusion output is still prone to drift. Figure 2.4 displays five stationary measurements of the roll angle of the IMU that shows a significant, inconsistent, drift rate, with a mean drift rate of 0.145 °/s. A cause for the high drift rate can be bad calibration. The IMU has some automatic calibration procedures, but manual calibration is required to fine-tune these calibration procedures. Since the fusion algorithm is a black box, it is difficult to exactly determine the cause of drifting. Hence, the accelerometer and gyroscope data will be measured directly and converted to orientation data using a transparent algorithm to gain a better insight into the internal mechanics and to reduce the drift.

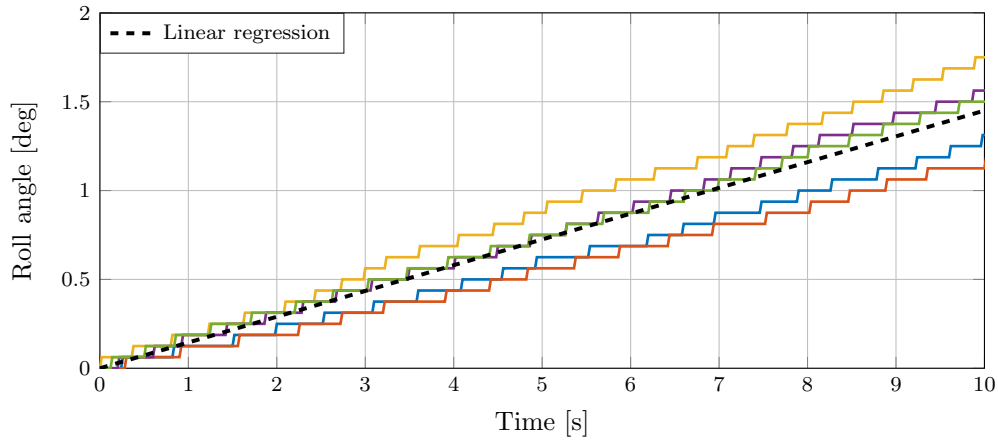


Figure 2.4: Five static measurements with the onboard fusion algorithm of the IMU, showing a significant drift rate in the roll angle.

To filter the accelerometer and gyroscope data a Mahony-Madgwick (MM) filter is proposed [30], [31]. It is assumed that the actuator does not twist along the backbone, and that during actuation the out of plane angle is negligible. Therefore, we are only interested in the in-plane angle. We define the angle φ as the absolute orientation of the actuator head in the measuring plane. This angle corresponds to the roll angle of the IMU.

The MM filter is detailed as follows. First, the accelerometer angle φ_{acc} is calculated with

$$\varphi_{\text{acc}} = \text{atan2}\left(a_y, a_x\right), \tag{2.1}$$

where a_y and a_x are the acceleration values measured by the accelerometer in the IMU's y-axis and x-axis, respectively. The gyroscope data ω_{gyr} is a measurement of the angular rate $\dot{\varphi}$. The filter

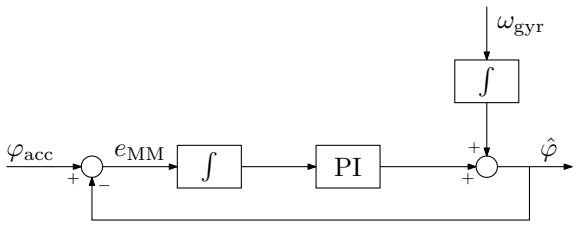


Figure 2.5: Block diagram of the Mahony-Madgwick filter.

uses φ_{acc} as setpoint, controlled with a PI controller. The gyroscope data ω_{gyr} can be perceived as output disturbance. The estimation of the angle φ is denoted $\hat{\varphi}$. A block diagram of the filter is displayed in Figure 2.5. The transfer function corresponding to this block diagram reads

$$\hat{\varphi} = \left(K_{p,\text{MM}} + \frac{1}{s}K_{i,\text{MM}}\right)\frac{1}{s}(\varphi_{\text{acc}} - \hat{\varphi}) + \frac{1}{s}\omega_{\text{gyr}}. \tag{2.2}$$

The sensor update rate is limited so the continuous transfer function has to be discretized to implement it on the Arduino Nano. Defining $e_{\text{MM}} = \varphi_{\text{acc}} - \hat{\varphi}$ and $I = \frac{1}{s}K_{i,\text{MM}}\, e_{\text{MM}}$, and using backward difference (2.2) can be written in discrete time as

$$e_{\text{MM},k} = \varphi_{\text{acc},k} - \hat{\varphi}_{k-1}, \tag{2.3a}$$

$$I_k = I_{k-1} + K_{i,\text{MM}}\, e_{\text{MM},k}\Delta t, \tag{2.3b}$$

$$\hat{\varphi}_k = \hat{\varphi}_{k-1} + (K_{p,\text{MM}}\, e_{\text{MM},k} + I_k + \omega_{\text{gyr},k})\Delta t, \tag{2.3c}$$

where Δt is the sample time, and the subscript k indicates the time step as $\hat{\varphi}_k = \hat{\varphi}(k\Delta t)$, for $k \in \mathbb{N}$. These equations can be directly implemented on the Arduino. Figure 2.6 shows an example measurement of the IMU in motion. The acceleration angle φ_{acc} and the gyroscope angle $\int \omega_{\text{gyr}}\text{d}t$ are shown, as well as the output of the MM filter $\hat{\varphi}$. As expected, the acceleration angle signal is noisy, and the gyroscope angle drifts. The MM filter angle solves both of these problems and gives a good estimate of the orientation. The filter gains $K_{p,\text{MM}}$ and $K_{i,\text{MM}}$ have been chosen iteratively such that the magnitude of the filtered orientation $\hat{\varphi}$ resembles the magnitude of the integrated gyroscope data ω_{gyr}, but shifted to compensate for the drift. The drift rate with the filter, obtained from a static measurement, is only -9.07×10^{-4} °/s which is a very significant improvement over the IMU's own fusion algorithm.

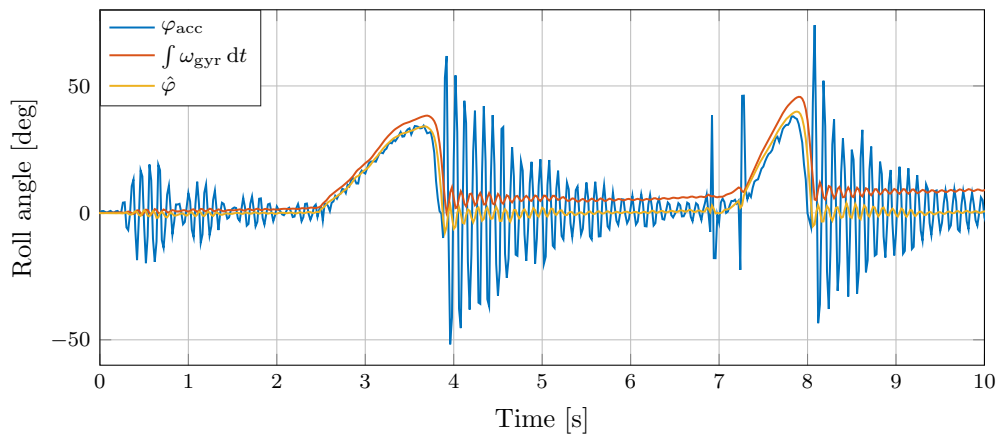


Figure 2.6: Example measurement with the IMU in motion. The MM filter gains are chosen $K_{p,\text{MM}} = 3$ and $K_{i,\text{MM}} = 3$.

2.2.2 Camera calibration

The Pixy2 outputs the location of a marker in pixels. To convert the pixel data to world units we use the pinhole camera model [32]. The pinhole camera model is a common way to describe the relationship between three-dimensional real world coordinates and the image coordinates. A pinhole camera consists of an aperture through which an image is projected onto an image plane. The pinhole camera model assumes an ideal pinhole camera, i.e. the aperture is a single point and there are no lenses to focus. The real camera is of course not ideal and lens distortion decreases the accuracy of this model, which is especially prevalent around the edges of the image. It should be noted that the mapping described here assumes that the world coordinates are restricted to

a single plane. The actuator will not remain in this plane exactly, but it is assumed that during actuation the out-of-plane motion is negligible.

Denote a point in the two-dimensional image plane $\boldsymbol{P}_\mathrm{i} = \begin{bmatrix}\mu & \nu\end{bmatrix}^\top$, where μ and ν are horizontal and vertical pixel coordinates of the image, respectively. A point in the three-dimensional world coordinates is denoted by $\boldsymbol{P}_\mathrm{w} = \begin{bmatrix}X & Y & Z\end{bmatrix}^\top$. By increasing the dimension of these position vectors we obtain the augmented vectors $\tilde{\boldsymbol{P}}_\mathrm{i} = \begin{bmatrix}\mu & \nu & 1\end{bmatrix}^\top$ and $\tilde{\boldsymbol{P}}_\mathrm{w} = \begin{bmatrix}X & Y & Z & 1\end{bmatrix}^\top$. The mapping from the 3D world coordinates to the 2D image coordinates is then defined as [32]

$$\tilde{\boldsymbol{P}}_\mathrm{i} = \frac{1}{s}\boldsymbol{K}\begin{bmatrix}\mathbf{R} & \boldsymbol{t}\end{bmatrix}\tilde{\boldsymbol{P}}_\mathrm{w}, \tag{2.4}$$

with scaling factor s, the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation vector $\boldsymbol{t} \in \mathbb{R}^3$, and the intrinsic matrix $\boldsymbol{K} \in \mathbb{R}^{3\times3}$. The rotation matrix and translation vector represent the rigid transformation from the world frame to the camera frame in world coordinates. The camera frame coincides with the location of the aperture. The intrinsic parameters map the 3D camera frame in real world coordinates to the 2D image plane in pixels. Figure 2.7 displays these two mappings.

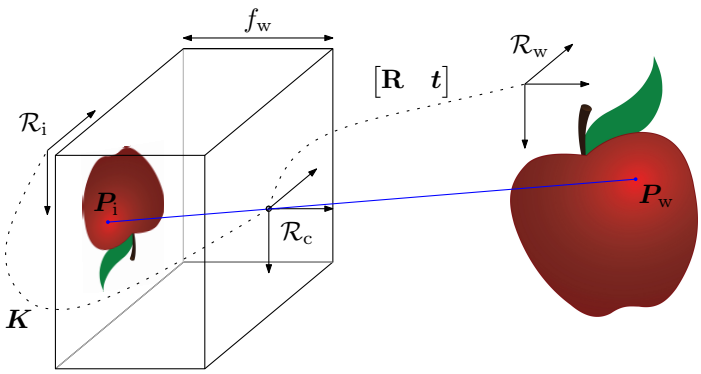


Figure 2.7: The pinhole camera model showcasing the extrinsic and intrinsic mappings.

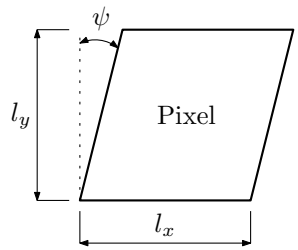


Figure 2.8: Definition of the pixel skew.

The matrix $\boldsymbol{K}$, an intrinsic matrix from the camera, is defined as [32]

$$\boldsymbol{K} = \begin{bmatrix}\sigma_\mu & \gamma & \mu_0 \\ 0 & \sigma_\nu & \nu_0 \\ 0 & 0 & 1\end{bmatrix}, \tag{2.5}$$

with (μ_0, ν_0) the location of the principal point, σ_μ and σ_ν as scale factors in the image axes, respectively, and the skewness parameter γ. The scale factors are defined as

$$\sigma_\mu = \frac{f_\mathrm{w}}{l_x}, \tag{2.6a}$$

$$\sigma_\nu = \frac{f_\mathrm{w}}{l_y}, \tag{2.6b}$$

with f_w the focal length of the camera in world units, and l with $i \in \{x, y\}$ the pixel size in world units. The skewness parameter is defined as

$$\gamma = f_\mathrm{w} \tan\psi, \tag{2.7}$$

with the angle ψ defined in Figure 2.8.

The assumption is made that the SRM undergoes only planar movement. Suppose that the mapping between point $\boldsymbol{P}_{\mathrm{w}}$ that is on the SRM plane and point $\boldsymbol{P}_{\mathrm{i}}$ on the Pixy2 image is of interest. The world coordinate frame can be chosen such that the SRM plane is on $Z = 0$ of the world coordinate system. This simplifies (2.4), using $\mathbf{R}_{\bullet,i}$ as the i-th colum of $\mathbf{R}$, to

$$s \begin{bmatrix} \mu \\ \nu \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \mathbf{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \mathbf{R}_{\bullet,1} & \mathbf{R}_{\bullet,2} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{2.8}$$

By redefining $\tilde{\boldsymbol{P}}_{\mathrm{w}} = \begin{bmatrix} X & Y & 1 \end{bmatrix}^\top$, and defining the homography $\boldsymbol{A} = \begin{bmatrix} \mathbf{R}_{\bullet,1} & \mathbf{R}_{\bullet,2} & \boldsymbol{t} \end{bmatrix}$, the relationship between the point $\boldsymbol{P}_{\mathrm{i}}$ and $\boldsymbol{P}_{\mathrm{w}}$ is given by

$$\tilde{\boldsymbol{P}}_{\mathrm{i}} = \frac{1}{s} \boldsymbol{A} \tilde{\boldsymbol{P}}_{\mathrm{w}}. \tag{2.9}$$

To evaluate this mapping, the extrinsic and intrinsic parameters have yet to be identified. These parameters can be found with Zhang's method [32]. Define a points on a real world plane, of which we take b amount of images. The task is then to minimize [32]

$$\begin{aligned} &\underset{\boldsymbol{K},\mathbf{R}_i,\boldsymbol{t}_i}{\arg\min} \sum_{i=1}^{b} \sum_{j=1}^{a} \left\| \boldsymbol{P}_{\mathrm{i},ij} - \hat{\boldsymbol{P}}_{\mathrm{i},ij} \left(\boldsymbol{K}, \mathbf{R}_i, \boldsymbol{t}_i, \boldsymbol{P}_{\mathrm{w},j} \right) \right\|^2, \\ &\text{s.t. } \boldsymbol{K} \in \mathbb{R}^{3\times 3},\ \mathbf{R}_i \in \mathbb{R}^3,\ \boldsymbol{t}_i \in \mathbb{R}^3, \end{aligned} \tag{2.10}$$

with $\hat{\boldsymbol{P}}_{\mathrm{i},ij} \left(\boldsymbol{K}, \mathbf{R}_i, \boldsymbol{t}_i, \boldsymbol{P}_{\mathrm{w},j} \right)$ the projection of the point $\boldsymbol{P}_{\mathrm{w},j}$ in image i, obtained using (2.9). This minimization problem is implemented in MATLAB's Camera Calibration Toolbox. To this end, 17 pictures have been taken with the Pixy2 of a 6×8 checkerboard pattern with a grid size of 20 mm. These pictures have been taken from various angles and distances. The real world points $\boldsymbol{P}_{\mathrm{w}}$ are defined as the checkerboard corner locations, whose locations are known and constant and which means $a = 35$. With $b = 17$, this amounts to 595 data points. The pictures are imported in the toolbox which is able to detect the checkerboard points in the image frame $\boldsymbol{P}_{\mathrm{i},ij}$ of each image i. With an initial guess for $\boldsymbol{K}$, $\mathbf{R}$, and $\boldsymbol{t}$, the toolbox solves the minimization problem (2.10) with the Levenberg-Marquardt algorithm. The resulting intrinsic and extrinsic parameters lead to an overall mean error, i.e. the overall mean distance between the pixel coordinates and the reprojection of the world coordinates, of 0.31 pixels.

The intrinsic matrix is camera specific, so this matrix does not change when the camera position is changed. The extrinsic matrix depends on the location of the camera with respect to the measuring plane, i.e. the actuator plane in our case. As a result the extrinsic matrix has to be determined once the camera is installed on the frame of the experimental setup. The same checkerboard as used for determining the intrinsic parameters is then placed in the actuator plane at a known location. MATLAB's Camera Calibration toolbox is used to calculate the extrinsic parameters, i.e. the rotation matrix and translation vector, corresponding to that camera position.

The mapping is implemented on the Arduino, so it is possible to calculate the world coordinates of the marker every timestep. If at timestep k the pixel values u_k and v_k have been measured by the Pixy2, using (2.9) these pixel coordinates can be transformed to the marker location in world coordinates at timestep k with

$$\begin{bmatrix} X_k \\ Y_k \\ 1 \end{bmatrix} = s \boldsymbol{A}^{-1} \begin{bmatrix} \mu_k \\ \nu_k \\ 1 \end{bmatrix}. \tag{2.11}$$

To apply this equation, first the scaling factor s has to be recalculated every timestep. It follows from (2.11) that the scale factor at time step k can be calculated with

$$s_k = \left(\left[\boldsymbol{A}^{-1}\right]_{(3,3)} + \left[\boldsymbol{A}^{-1}\right]_{(3,1)} \mu_k + \left[\boldsymbol{A}^{-1}\right]_{(3,2)} \nu_k \right)^{-1}. \tag{2.12}$$

The position of the marker does not coincide with the actuator head, which is the position of interest. The position of the actuator head can simply be obtained by subtracting an offset, as explained in Appendix A.

The calibration process described here does not account for lens distortion. The Pixy2 is not an actual pinhole camera, and the lens of the Pixy2 is not perfect either which gives rise to distortion effects. These distortion effects can be reduced with a distortion model. Distortion parameters can be estimated using a minimization problem similar to (2.10). An analysis of the lens distortion is given in Appendix B. This analysis shows that the distortion of the Pixy2 is negligible in the desired measurement range, and distortion reduction is computationally expensive. Hence, lens distortion is neglected.

Filtering

The Pixy2 data is quantized to the pixel grid, which means that the marker trajectory is limited to this grid as well. Figure 2.9 shows an example of how the real trajectory can differ from the trajectory measured by the Pixy2. This quantization, as well as overall tracking errors, causes the Pixy2 data to be jittery. When this pixel data is mapped to real world coordinates, these coordinates become jittery as well. Therefore, a lowpass filter is proposed to obtain smooth location data. The blue line in Figure 2.10 shows an example measurement of the x-location of the marker where the unfiltered Pixy2 data is used. A lowpass filter with a cutoff frequency of 1 Hz is introduced. The cutoff frequency can be chosen this low because of the relatively slow motion of the soft robot. The filter is implemented as an infinite impulse response filter such that it is able to filter the data in real-time. The orange line in Figure 2.10 shows the filtered x-location corresponding to the same measurement as the blue line. As a property of this lowpass filter, a small delay is introduced. As illustrated by the figure the filter is able to smooth out the quantization attribute, but it still captures the motion of the marker properly (such as the spike around the 4-second mark).

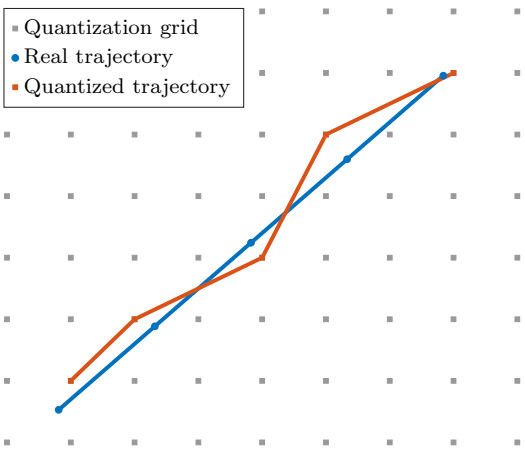


Figure 2.9: Example showing how the true trajectory of the marker is quantized to the pixel grid.

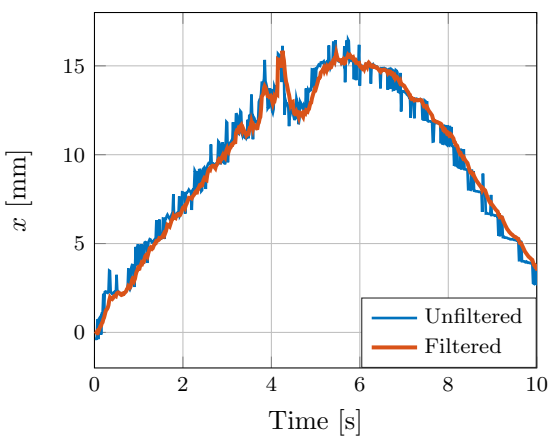


Figure 2.10: Unfiltered and filtered Pixy2 data converted to x-position.

2.3 Dynamics of the pumps

The Arduino microcontroller powers the DC motors with a digital PWM signal, which is regulated with a PWM value in the range 0 to 255. The signal consists of a series of 12 V pulses, and the PWM value determines the duty cycle of these pulses. By limiting the pulses at a high switching frequency, this PWM signal can simulate an analog voltage in the range 0 V to 12 V. So, while the motor input voltage is technically fixed at 12 V, in the remainder of this report the motor input voltage is defined as the simulated analog voltage.

Figure 2.11 shows the steady state elongation of the actuator for a range of input voltages. These results are obtained by applying an equal voltage to both motors, waiting for the actuator to stop moving, and subsequently measuring the actuator length with the Pixy2. Three measurements have been done per voltage level. As seen in the graph, the pumps generate enough pressure to cause a displacement if the motor input is greater than approximately 4.7 V. Together with the maximum motor input of 12 V, this means that we have a saturated system. If this system is to be controlled, this saturation can cause inability to reach a certain state when the controller output is outside the 4.7 V to 12 V range. Control of the system will be discussed in more detail in Chapter 4. The regression in Figure 2.11 is a piecewise function, containing a deadzone and a third order polynomial. Defining $\hat{\varepsilon}$ as the measured elongation, it is governed by the equation

$$\hat{\varepsilon} = \hat{\ell}_{\text{ss}} - \ell_0 = \begin{cases} 0 & 0\,\text{V} \leq u < 4.7\,\text{V}, \\ (-0.1493u^3 + 3.376u^2 - 25.87u + 54.13) \times 10^{-3} & 4.7\,\text{V} \leq u \leq 12\,\text{V}, \end{cases} \tag{2.13}$$

where $\hat{\ell}_{\text{ss}}$ is the measured steady state length of the actuator in meters and u the motor input in volt.

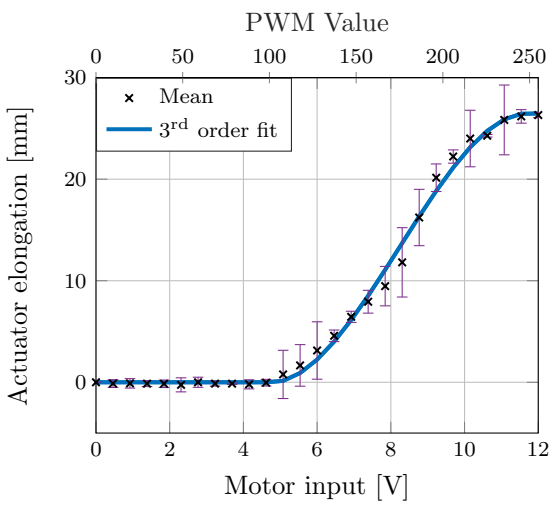


Figure 2.11: Measurements of steady state elongation of the actuator as a function of input voltage. The error bars represent a 95% confidence interval.

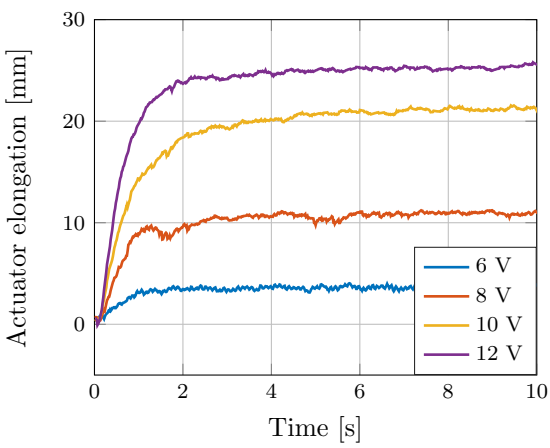


Figure 2.12: Step response of the actuator elongation for various motor input voltages.

Figure 2.12 shows the elongation step response of the actuator for four different input voltages. The input voltages for both motors are chosen identical in order to achieve only elongation and no curvature. Due to imperfections, the motors and pumps are not fully identical, so some curvature is observed in the actuator. The step response profile is indicative of a first order system. The settling times of the four cases seem to be in the same range. It seems that (especially for the higher voltage levels) there is a mismatch between the steady state elongation and the step response

elongation. But the higher voltage step responses in Figure 2.12 have not reached a steady state yet, their elongation is still converging towards the steady state level in Figure 2.11.

Summary

This chapter describes a planar soft robotic system that is actuated by pumps. This actuator is mounted in a frame, along with an IMU and camera system. The IMU data is filtered with a Mahony-Madgwick filter, and the position data is obtained with the pinhole camera model, to obtain discrete position and orientation data X_k, Y_k, and $\hat{\varphi}_k$. This can be used for closed loop control, but first, as will be explained in the next chapter, we explore the orientation and position kinematics of the actuator.

Chapter 3

Modelling of the actuator

Dynamic models have a range of applications, such as design optimisation, predicting dynamic behaviour, and model-based control. Therefore, this chapter discusses the derivation of a dynamic model for a planar soft robotic actuator. Using kinematic equations, the full pose of the actuator can be obtained from the IMU and camera data from the previous chapter. The equations of motion are derived with the Euler-Lagrange method. The lumped mass model presented here is based on work by [10], adapted to model a soft robot actuator in 2D. Stiffness and damping parameters are estimated from measurement data. A creep model is proposed to enhance the model fit. Finally, a model is developed that enables simulations including the air pumps.

3.1 Kinematics of the actuator

The actuator presented in this thesis consists of two parallel bellows. To describe the relative and absolute positions of the actuators a kinematic model is required. To this end, three coordinate frames are introduced, namely the actuator space $\boldsymbol{q}$, configuration space $\boldsymbol{k}$, and task space $\boldsymbol{x}$. The actuator space consists of the actuator states

$$\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 \end{bmatrix}^\top , \tag{3.1}$$

which are defined as the length of each bellows, with $q_i \in \mathbb{R}_{>0}$. The reason the bellow lengths are taken as actuator states is that in our case these are directly actuated with pumps, and it makes it easier to model the forces exerted by the bellows. The notation q_i with index $i \in \{1, 2\}$ is used to denote the bellow lengths generally.

Geometrically exact models exist to describe the kinematics of soft robots, but these are complicated and computationally expensive. A more simple and widely employed method to describe the configuration states is the constant curvature approach [7]. This kinematic model will not describe the behaviour of the SRM exactly, but arguably, as we have a lightweight and compact actuator, the model will be close [8]. For the 2D case, this constant curvature approach requires only the backbone length ℓ and curvature κ of each section i to obtain the configuration states

$$\boldsymbol{k} = \begin{bmatrix} \kappa & \ell \end{bmatrix}^\top , \tag{3.2}$$

to fully describe the position and orientation of the SRM. The inverse of the curvature is equal to the (non-constant) radius of the circular arc of the backbone. From circular geometry, it follows that the bending angle of the actuator φ is equal to the product of backbone length and curvature,

i.e. $\varphi = \kappa\ell$ (see Figure 3.1).

Figure 3.1 shows a visualisation of the definition he actuator states and configuration states. The mapping f_{specific} can be used to obtain the configuration states as a function of the actuator states, i.e. $\boldsymbol{k}(\boldsymbol{q})$. This mapping is called specific since it changes for different actuators and it depends on the definition of the kinematics. For our actuator, with the constant curvature approach and circle geometry the mapping is defined as

$$\boldsymbol{k}(\boldsymbol{q}) = \begin{bmatrix} \kappa(\boldsymbol{q}) \\ \ell(\boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} \dfrac{q_1 - q_2}{(q_1 + q_2) r_{\text{B}}} \\ \frac{1}{2}(q_1 + q_2) \end{bmatrix}, \tag{3.3}$$

where r_{B} is the distance between the backbone and bellows, see Figure 3.1.

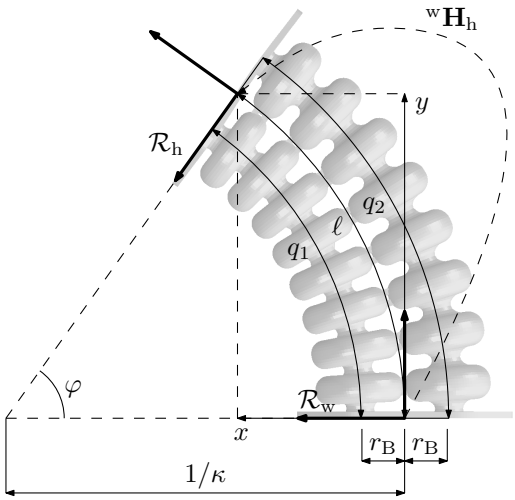


Figure 3.1: Definition of the actuator states $\boldsymbol{q}$, configuration states $\boldsymbol{k}$, and task states $\boldsymbol{x}$ for a planar soft robot. The definitions of the world frame $\mathcal{R}_{\text{w}}$ (equal to the base frame $\mathcal{R}_{\text{B}}$) and head frame $\mathcal{R}_{\text{h}}$ are shown as well.

For practical purposes, the end-effector location is often described in Cartesian coordinates with respect to a (fixed) world reference frame. For a 2D actuator, the task space coordinates are defined as

$$\boldsymbol{x} = \begin{bmatrix} x & y & \varphi \end{bmatrix}^\top, \tag{3.4}$$

with position coordinates x and y and orientation φ. The mapping $\boldsymbol{f}_{\text{general}}$ maps the configuration states to the task space with the homogeneous transformation matrix $\mathbf{H} \in \mathbb{R}^{3\times 3}$

$${}^{j}\mathbf{H}_i = \begin{bmatrix} {}^{j}\mathbf{R}_i & {}^{j}\boldsymbol{t}_i \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix}, \tag{3.5}$$

with rotation matrix $\mathbf{R} \in \mathbb{R}^{2\times 2}$ and translation vector $\boldsymbol{t} \in \mathbb{R}^{2\times 1}$. The homogeneous transformation matrix maps the relative position and orientation from coordinate frame i to coordinate frame j. The mapping from the base of the actuator $\mathcal{R}_{\text{b}}$ to the head of the actuator $\mathcal{R}_{\text{h}}$ can be described

with the homogeneous transformation matrix (3.5) to obtain [10].

$$
{}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}} = \begin{bmatrix} \cos\kappa\ell & -\sin\kappa\ell & \dfrac{1-\cos\kappa\ell}{\kappa} \\ \sin\kappa\ell & \cos\kappa\ell & \dfrac{\sin\kappa\ell}{\kappa} \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.6}
$$

The coordinates of the actuator head in the world reference frame $\mathcal{R}_{\mathrm{w}}$ are obtained by multiplying the local transformation matrix with the transformation matrix between the world reference frame and base frame, ${}^{\mathrm{w}}\mathbf{H}_{\mathrm{b}}$, to obtain

$$
{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}} = {}^{\mathrm{w}}\mathbf{H}_{\mathrm{b}}\,{}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}. \tag{3.7}
$$

To simplify this, we take the world reference frame coinciding with the base frame, i.e. ${}^{\mathrm{w}}\mathbf{H}_{\mathrm{b}} = \boldsymbol{I}_3$. This reduces (3.7) to

$$
{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}} = {}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}. \tag{3.8}
$$

As one can see, when the curvature of the actuator is zero, i.e. $\kappa = 0$, numerical singularities can arise in the $(1,3)^{\mathrm{th}}$ and $(2,3)^{\mathrm{th}}$ entries of the matrix ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}$. Physically, zero or negative curvature is perfectly feasible. Mathematically, these entries are well-defined, as they are differentiable and their limits, derived with L'Hôpital's rule,

$$
\lim_{\kappa\to 0} \frac{1-\cos(\kappa\ell)}{\kappa} = 0, \tag{3.9a}
$$

$$
\lim_{\kappa\to 0} \frac{\sin(\kappa\ell)}{\kappa} = \ell, \tag{3.9b}
$$

exist as well. However, numerically ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}$ can become ill-conditioned when κ approaches 0. To circumvent this issue, two different transformation matrices can be used with a switching function depending on the value of κ. However, switching can cause numerical instabilities. Another way to mitigate these numerical singularities is by using a Taylor series. The $(1,3)^{\mathrm{th}}$ and $(2,3)^{\mathrm{th}}$ entries of (3.6), that we define as H_{13} and H_{23}, respectively, can be written with a Taylor series around $\kappa = 0$ as

$$
H_{13} = \frac{1-\cos\kappa\ell}{\kappa} = \ell\sum_{n=0}^{\infty}\frac{(\kappa\ell)^{2n+1}}{(2n+2)!}(-1)^n = \frac{\kappa\ell^2}{2!} - \frac{\kappa^3\ell^4}{4!} + \frac{\kappa^5\ell^6}{6!} - \dots \quad , \tag{3.10a}
$$

$$
H_{23} = \frac{\sin\kappa\ell}{\kappa} = \ell\sum_{n=0}^{\infty}\frac{(\kappa\ell)^{2n}}{(2n+1)!}(-1)^n = \ell - \frac{\kappa^2\ell^3}{3!} + \frac{\kappa^4\ell^5}{5!} - \frac{\kappa^6\ell^7}{7!} + \dots \quad . \tag{3.10b}
$$

By plugging (3.10a) and (3.10b) into (3.6), and limiting the order of the Taylor series, an approximation of ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}$ can be calculated that we define as ${}^{\mathrm{b}}\hat{\mathbf{H}}_{\mathrm{h}}$. The accuracy of this approximation depends on the truncation order. Figure 3.2 shows the difference between H_{13} and the approximation $\hat{H}_{13}$ for various truncation orders. Figure 3.3 shows the corresponding error norm. From truncation order 7 and higher, the error is barely noticeable. Therefore, the matrix ${}^{\mathrm{b}}\hat{\mathbf{H}}_{\mathrm{h}}$ up to $O(\kappa^7)$ will be used hereafter in simulations in place of the original matrix ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}$.

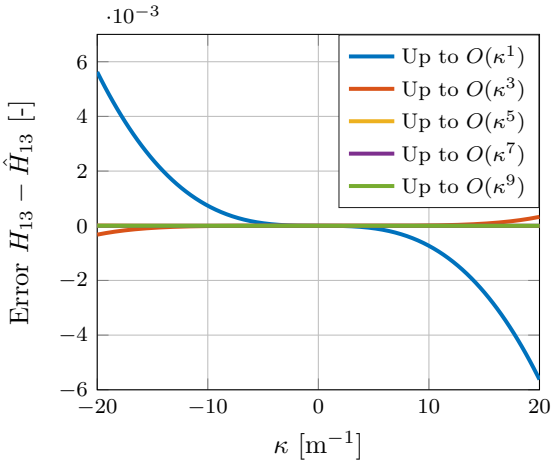


Figure 3.2: Error between exact H_{13} and approximated $\hat{H}_{13}$, as a function of κ for various truncation orders, and with $\ell = 0.065$.

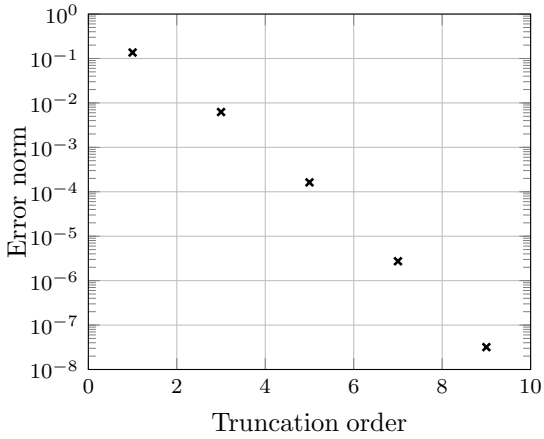


Figure 3.3: Error norm corresponding to Figure 3.2.

The sensors measure the coordinates of the actuator in the task space (3.4). The inverse kinematics have to be formulated to express the actuator space in task space coordinates, which is convenient for control purposes. We can describe the curvature κ and length ℓ of the actuator in the actuator space and task space coordinates separately. The curvature and length as a function of actuator states are given in (3.3). The curvature and length as a function of task space coordinates can be derived with circular segment geometry to find

$$\boldsymbol{k}(\boldsymbol{x}) = \begin{bmatrix} \kappa(\boldsymbol{x}) \\ \ell(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \dfrac{2\sin\left(\frac{1}{2}\varphi\right)}{\sqrt{x^2+y^2}} \\ \dfrac{\varphi\sqrt{x^2+y^2}}{2\sin\left(\frac{1}{2}\varphi\right)} \end{bmatrix}. \tag{3.11}$$

By equating (3.3) and (3.11) and solving for $\boldsymbol{q}$ we obtain an expression for $\boldsymbol{q}(\boldsymbol{x})$:

$$q_1 = r_B\varphi + \frac{\varphi\sqrt{x^2+y^2}}{2\sin\left(\frac{1}{2}\varphi\right)}, \tag{3.12a}$$

$$q_2 = -r_B\varphi + \frac{\varphi\sqrt{x^2+y^2}}{2\sin\left(\frac{1}{2}\varphi\right)}. \tag{3.12b}$$

As can be seen, when the orientation angle is zero, the denominators in (3.12) are equal to zero. This is similar to the numerical issues with ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}$. Define

$$f(\varphi) = \frac{\varphi}{\sin\left(\frac{1}{2}\varphi\right)}, \tag{3.13}$$

which is differentiable and $\lim_{\varphi\to 0} f(\varphi) = 2$ exists. Therefore, $f(\varphi)$ is mathematically well-defined, but numerical singularities can arise. To prevent this numerical singularity a Taylor series around $\varphi = 0$ is considered

$$\hat{f}(\varphi) = 2 + \frac{\varphi^2}{12} + \frac{7\varphi^4}{2880} + \frac{31\varphi^6}{483840} + O(\varphi^8). \tag{3.14}$$

By using the Taylor series only up to a sufficient truncation order, an approximation of $f(\varphi)$, defined as $\hat{f}(\varphi)$, can be substituted in the inverse kinematics equations (3.12) to obtain an ap-

proximation of the inverse kinematics $\hat{\boldsymbol{q}}(\boldsymbol{X})$. This approximation contains no singularities, but as a downside the error of the approximation increases as φ diverges from zero. The truncation order has a big influence on the accuracy of the approximation. Figure 3.4 shows $f(\varphi)$ and $\hat{f}(\varphi)$ up to several orders. The error $f(\varphi) - \hat{f}(\varphi)$ can be seen in Figure 3.5. It is expected that the actuator will not bend further than $\pi/2\,\text{rad}$. For $\varphi = \pi/2\,\text{rad}$ rad, the error $f(\varphi) - \hat{f}(\varphi)$ up to $O(\varphi^6)$ is only 4.45×10^{-5} rad. Therefore the approximation $\hat{\boldsymbol{q}}(\boldsymbol{X})$ with $\hat{f}(\varphi)$ up to $O(\varphi^6)$ is sufficient and will be utilized for control in the experimental implementation (see Chapter 4).

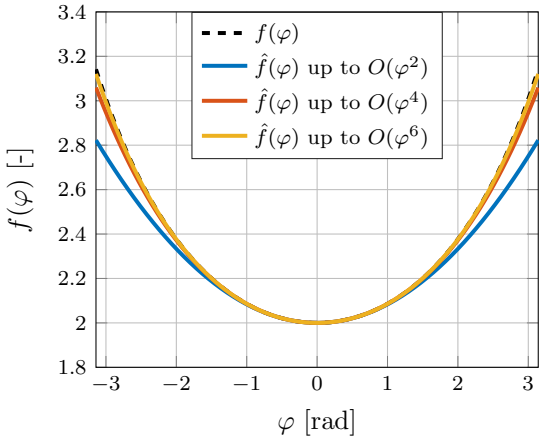


Figure 3.4: Function $f(\varphi)$ and its Taylor series for several truncation orders.

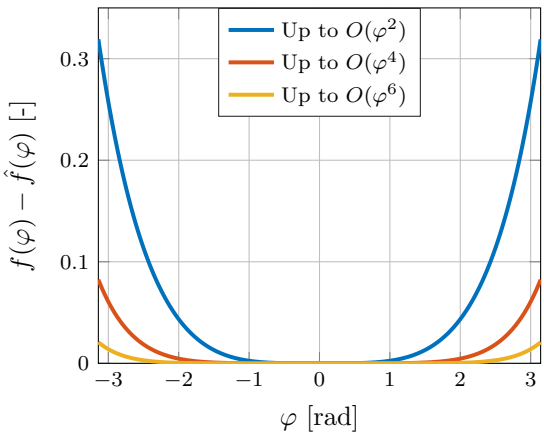


Figure 3.5: Error between $f(\varphi)$ and its Taylor series corresponding to Figure 3.4.

3.2 Dynamics of the actuator

In order to achieve a computationally efficient model, the mass will be discretized. This results in a lumped mass model that contains a discretized mass, connected to a fixed base via springs and dampers. The equations of motion of the lumped mass model are derived with the Euler-Lagrange formalism

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} = Q_i^{\mathrm{nc}}, \tag{3.15}$$

where the generalized coordinates q_i are defined as the actuator states (3.1). The Euler-Lagrange equations include the the kinetic energy $T(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}_+$, the velocity-independent potential energy $U(\boldsymbol{q}) \in \mathbb{R}$, and the generalized non-conservative forces $Q_i^{\mathrm{nc}}(\boldsymbol{q}) \in \mathbb{R}$ corresponding to q_i.

To discretize the distributed mass, the following assumptions are made:

Assumption 3.1. *The mass m of the actuator is concentrated at a fixed position in the actuator head frame $\mathcal{R}_h$, with the center of mass located at ${}^{h}\boldsymbol{r}_{\mathrm{S}}$ (which in our case coincides with the location of $\mathcal{R}_h$).*

Assumption 3.2. *The moment of inertia at the center of mass ${}^{\mathrm{h}}J_{\mathrm{S}} \in \mathbb{R}$ is constant.*

Figure 3.6 illustrates this lumped model. The grey bar represents the lumped mass with mass m and inertia ${}^{\mathrm{h}}J_{\mathrm{S}}$, with the center of mass coinciding with location of frame $\mathcal{R}_{\mathrm{h}}$. Following Assumption 3.1, the position of the center of mass in the head frame ${}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}$ is constant. For constructing the dynamic model, we use the position of the center of mass in the world frame, denoted by ${}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}}$.

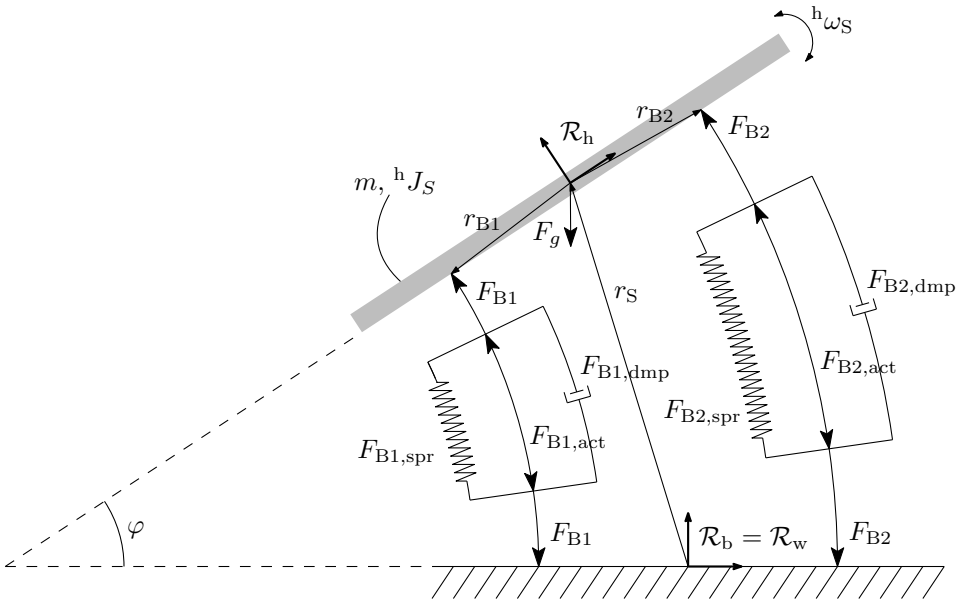


Figure 3.6: Lumped mass model of the actuator, containing springs, dampers, and pressure forces.

By using the global homogeneous transformation (3.7), this position vector can be written as

$$ {}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}} = {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}. \tag{3.16} $$

Since this is a 2D model, the position vectors are 2-dimensional. However, the transformation matrix ${}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}$ has dimension 3×3. Hence, a position vector has to be extended to $\begin{bmatrix}\boldsymbol{r}^\top & 1\end{bmatrix}^\top$ to apply this transformation. When multiplying forces with the transformation matrix the force vector has to be extended to $\begin{bmatrix}\boldsymbol{F}^\top & 0\end{bmatrix}^\top$, which we will need later on.

The dynamic model also requires the derivative of ${}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}}$ with respect the bellow lengths q_i, and the time derivative. The partial derivative of the position vector ${}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}}$ with respect to the generalized coordinate q_i gives

$$ \frac{\partial {}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}}}{\partial q_i} = \frac{\partial {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_i}\,{}^{i\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} + {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}\underbrace{\frac{\partial {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}}{\partial q_i}}_{=0}. \tag{3.17} $$

The position vector of the center of mass in $\mathcal{R}_{\mathrm{h}}$, ${}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}$, is independent of $\boldsymbol{q}$. Therefore, the second term in (3.17) is equal to zero. Using the chain rule, the time derivative of ${}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}}$ is equal to

$$ {}^{\mathrm{w}}\dot{\boldsymbol{r}}_{\mathrm{S}} = {}^{\mathrm{w}}\dot{\mathbf{H}}_{\mathrm{h}}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} + {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}\underbrace{{}^{\mathrm{h}}\dot{\boldsymbol{r}}_{\mathrm{S}}}_{=0}. \tag{3.18} $$

The position vector ${}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}$ is constant, thus the second term in (3.18) is equal to zero.

Now that we have a definition for the mass, inertia, and position of the center of mass in the world frame and its derivatives, the kinetic energy can be defined. The kinetic energy of the system is governed by the sum of the translational and rotational kinetic energy

$$ T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \underbrace{\frac{1}{2} m\, {}^{\mathrm{w}}\dot{\boldsymbol{r}}_{\mathrm{S}}^{\top}\, {}^{\mathrm{w}}\dot{\boldsymbol{r}}_{\mathrm{S}}}_{\text{Translational}} + \underbrace{\frac{1}{2}\, {}^{\mathrm{w}}J_{\mathrm{S}}\, {}^{\mathrm{w}}\omega_{\mathrm{S}}^2}_{\text{Rotational}}. \tag{3.19} $$

Where ${}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}}$ is the position vector of the center of mass in $\mathcal{R}_{\mathrm{w}}$, and ${}^{\mathrm{w}}\dot{r}_{\mathrm{S}}$ its time derivative, and ${}^{\mathrm{w}}\omega_{\mathrm{S}}$ the rotational velocity of the lumped mass (see Figure 3.6). The moment of inertia and the rotational velocities are scalar values because the model is planar. This also means that the inertia and rotational velocity in the local frame and world frame are equal, i.e. ${}^{\mathrm{w}}J_S = {}^{\mathrm{h}}J_S$ and ${}^{\mathrm{w}}\omega_{\mathrm{S}} = {}^{\mathrm{h}}\omega_{\mathrm{S}}$. The rotational velocity is equal to the rate of change in orientation of the actuator

$$ {}^{\mathrm{w}}\omega_{\mathrm{S}} = \dot{\varphi}(\dot{\boldsymbol{q}}). \tag{3.20} $$

The rotational velocity is only dependent on $\dot{\boldsymbol{q}}$ and not on $\boldsymbol{q}$, because the orientation φ is a linear function in q_1 and q_2. Using (3.16) and (3.20) the expression for the kinetic energy becomes [10]

$$ T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2} m \left({}^{\mathrm{w}}\dot{\mathbf{H}}_{\mathrm{h}} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} \right)^{\top} \left({}^{\mathrm{w}}\dot{\mathbf{H}}_{\mathrm{h}} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} \right) + \frac{1}{2} {}^{\mathrm{w}}J_{\mathrm{S}} \dot{\varphi}(\dot{\boldsymbol{q}})^2. \tag{3.21} $$

The Euler-Lagrange equation requires derivatives of the kinetic energy. The derivative with respect to the generalized coordinate q_i is straightforward, and can be written as [10]

$$ \frac{\mathrm{d}T}{\mathrm{d}q_i} = m \left({}^{\mathrm{w}}\dot{\mathbf{H}}_{\mathrm{h}} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} \right)^{\top} \left(\frac{\partial {}^{\mathrm{w}}\dot{\mathbf{H}}_{\mathrm{h}}}{\partial q_i} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} \right). \tag{3.22} $$

The rotational part disappears, as $\dot{\varphi}$ is only a function of $\dot{\boldsymbol{q}}$. For the derivative of the kinetic energy with respect to $\dot{q}_i$ and time, we use the identity [10]

$$ \frac{\partial {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_i} = \frac{\partial {}^{\mathrm{w}}\dot{\mathbf{H}}_{\mathrm{h}}}{\partial \dot{q}_i}, \tag{3.23} $$

for which a proof is given in Appendix C. The derivative of the kinetic energy then becomes

$$ \frac{\partial}{\partial t} \frac{\partial T}{\partial \dot{q}_i} = m \left({}^{\mathrm{w}}\ddot{\mathbf{H}}_{\mathrm{h}} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} \right)^{\top} \left(\frac{\partial {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_i} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} \right) + \frac{1}{2} {}^{\mathrm{w}}J_{\mathrm{S}} \frac{\partial}{\partial t} \frac{\partial \dot{\varphi}^2}{\partial \dot{q}_i} + \frac{\partial T}{\partial q_i}. \tag{3.24} $$

The second term in (3.24) can be expressed, with $\varphi = \kappa \ell$ and (3.3), as

$$ \frac{1}{2} {}^{\mathrm{w}}J_{\mathrm{S}} \frac{\partial}{\partial t} \frac{\partial \dot{\varphi}^2}{\partial \dot{q}_i} = \frac{{}^{\mathrm{w}}J_{\mathrm{S}}}{4 r_{\mathrm{B}}^2} \left(\ddot{q}_i - \ddot{q}_j \right) \quad \text{with } \left\{ j \in \{1, 2\} \mid j \neq i \right\}. \tag{3.25} $$

To determine an expression for the potential and non-conservative forces, we first need to define the forces acting on the soft robot. These forces can be divided into bellow forces F_{B}, gravity force F_g and external forces F_{ext}. The bellow forces corresponding to bellow i are defined as $F_{\mathrm{B}i}$, and contain a spring force $F_{\mathrm{B}i,\mathrm{spr}}(q_i)$, damper force $F_{\mathrm{B}i,\mathrm{dmp}}(q_i, \dot{q}_i)$, and actuation force $F_{\mathrm{B}i,\mathrm{act}}(p_i)$. The bellow forces are also illustrated in Figure 3.6. The spring and damper forces and their estimation will be discussed in Section 3.3. The actuation force is the force exerted by the internal pressure of the bellows, defined as

$$ F_{\mathrm{B}i,\mathrm{act}}(p_i) = A_{\mathrm{p}} \left(p_i - p_{\mathrm{amb}} \right), \tag{3.26} $$

with p_i the internal pressure of bellow i, p_{amb} the ambient pressure, and A_{p} the effective surface that the pressure works against. In vector format, the internal pressures of the bellows are defined as $\boldsymbol{p} = \begin{bmatrix} p_1 & p_2 \end{bmatrix}^{\top}$. Despite the dependence of the actuation force on the internal pressure in reality, a pressure-free model of the acutation force will be discussed in Section 3.5. To include the bellow forces in the model, they have to be represented in the homogeneous notation in the world frame first. In the frame $\mathcal{R}_{\mathrm{h}}$, the bellow forces can be written as

$$ {}^{\mathrm{h}}\boldsymbol{F}_{\mathrm{B}i,\bullet} = \begin{bmatrix} 0 & F_{\mathrm{B}i,\bullet} & 0 \end{bmatrix}^{\top}, \tag{3.27} $$

where $\bullet$ can be substituted for spr, dmp, and act (denoting the spring, damper, and actuation forces, respectively). Because the bellows are always perpendicular to both the head and base frame, the force in homogeneous notation only has a term in the y-direction. The bellow forces in the world frame are then defined as

$$ {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}i,\bullet} = {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}} \begin{bmatrix} 0 & F_{\mathrm{B}i,\bullet} & 0 \end{bmatrix}^{\top}. \tag{3.28} $$

The gravity force applies to the center of mass ${}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}}$, and in homoegeneous notation can be written as

$$ {}^{\mathrm{w}}\boldsymbol{F}_{g} = -m\,{}^{\mathrm{w}}\boldsymbol{g}, \tag{3.29} $$

with $\boldsymbol{g}$ the gravitational acceleration vector. Besides the bellow and gravity forces, external forces can apply to the soft robot such as contact forces. For the remainder of this report, external forces are not considered so $F_{\mathrm{ext}} = 0$.

All remaining forces can be divided into conservative and non-conservative forces, with the spring and gravity force being conservative, and the damper and actuation forces being non-conservative. The potential function $U(\boldsymbol{q})$ is a summation of the projections of the conservative forces, and can be written as

$$ U = {}^{\mathrm{w}}\boldsymbol{F}_{g}^{\top}\,{}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}} + \sum_{\chi=1}^{2} {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{spr}}^{\top}\,{}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{B}\chi}. \tag{3.30} $$

The model in [10] also includes a rotational spring momentum in the potential function, dependent on the bending angle φ. The function of this momentum function is to stabilize the actuator to stand straight when no external forces are present. The rotational spring momentum is omitted from the model presented here because the spring and damper forces should achieve the same goal of straightening the actuator when no external forces apply. The partial derivative of U with respect to the actuator state q_i is [10]

$$ \frac{\partial U}{\partial q_i} = -m\,{}^{\mathrm{w}}\boldsymbol{g}^{\top}\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_i}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} + \sum_{\chi=1}^{2} {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{spr}}^{\top}\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_i}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{B}\chi}, \tag{3.31} $$

which is substituted in the Euler-Lagrange equation (3.15).

The generalized forces Q_i contain the projections of the damping and actuation forces. The generalized force corresponding to actuator state q_{ik} can now be written as [10]

$$ Q_i = \sum_{\chi=1}^{2}\left({}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{act}}^{\top}\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_i}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{B}\chi} - {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{dmp}}^{\top}\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_i}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{B}\chi}\right). \tag{3.32} $$

When (3.22), (3.24), (3.32), and (3.31) are substituted into the Euler-Lagrange equation (3.15), we obtain two coupled second-order differential equations. These differential equations can be rewritten in the form of [10]

$$ \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{\tau}(\boldsymbol{p}(\boldsymbol{t})). \tag{3.33} $$

with the mass matrix $\boldsymbol{M} \in \mathbb{R}^{2\times 2}$, Coriolis matrix $\boldsymbol{C} \in \mathbb{R}^{2\times 2}$, input-independent force vector $\boldsymbol{N} \in \mathbb{R}^{2}$ and input force vector $\boldsymbol{\tau} \in \mathbb{R}^{2}$. The mass matrix can be calculated with the entrywise equation [10]

$$ \boldsymbol{M}_{(r,c)} = m\left(\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_c}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}\right)^{\top}\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_r}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} + \frac{{}^{\mathrm{w}}J}{4r_{\mathrm{B}}^{2}}(-1)^{(r+c)}, \tag{3.34} $$

where r and c are the matrix row and column, respectively. The mass matrix contains the inertial terms of the equations of motion. The entrywise Coriolis matrix, which contains the Coriolis and centrifugal force terms of the equations of motion, is defined as [10]

$$\boldsymbol{C}_{(r,c)} = m \left(\frac{\partial {}^{\mathrm{w}}\dot{\mathbf{H}}_{\mathrm{h}}}{\partial q_c} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} \right)^{\top} \frac{\partial {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_r} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}. \tag{3.35}$$

The input-independent force vector $\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ contains stiffness, damping, gravitational and external forces. Its entrywise equation is defined as [10]

$$\boldsymbol{N}_{(r)} = -m\, {}^{\mathrm{w}}\boldsymbol{g}^{\top} \tfrac{\partial {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_r} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} + \sum_{\chi=1}^{2} \left({}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{spr}}^{\top} + {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{dmp}}^{\top} \right) \tfrac{\partial {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_r} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{B}\chi}. \tag{3.36}$$

The input-dependent force vector $\boldsymbol{\tau}(\boldsymbol{p})$ is contains the controllable actuation forces as a function of the internal bellows pressure $\boldsymbol{p}$. The entrywise equation for this vector is defined as [10]

$$\boldsymbol{\tau}_{(r)} = \sum_{\chi=1}^{2} {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{act}}^{\top} \frac{\partial {}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_r} {}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{B}\chi}. \tag{3.37}$$

The model presented here describes the dynamics of a single soft robotic actuator section. Soft robots often consist of serially stacked sections, such as in Figure 1.1b, that are independently actuated. The model can be efficiently extended to multiple sections. The actuator states $\boldsymbol{q}$ and configuration states $\boldsymbol{k}$ can be defined for each section separately, to obtain $\boldsymbol{q} \in \mathbb{R}_{+}^{2n}$ and $\boldsymbol{k} \in \mathbb{R}^{2n}$ with n the amount of sections. By defining local coordinate frames at the heads of each section, a homogeneous transformation matrix ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{h}}$ can be defined that maps coordinates in the base frame of the bottom section to the head frame of the topmost section. The mass of each section can be discretized, and the gravity and bellow forces can be calculated for each section. This leads to $2n$ coupled equations which constitute the equations of motion.

3.3 Parameter estimation of the actuator

In order to use the model in simulations, some parameters need to be identified first. Parameters such as mass, nominal lengths and rotational inertia are assumed to be constant and these can be measured from the 3D-printed actuator and the CAD model. The measured values are listed in Table 3.1. For the mass only half the actuator is considered, due to the discretization of the mass. Therefore the mass in the model consists only of the mass of one connector (Figure 2.1) and half of the soft robot. Similarly, the rotational inertia consists of the rotational inertia of one connector and half of the soft robot. The mass and inertia of the cables are negligible. The position of the center of mass coincides with the location of the head frame $\mathcal{R}_{\mathrm{h}}$, therefore the position vector ${}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}}$ is equal to zero, i.e. ${}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\top}$ (the 1 is due to the augmentation).

Table 3.1: Constant model parameters.

Parameter	Value	Unit
Mass m	0.0166	[kg]
Inertia ${}^{\mathrm{w}}J$	$1.27 \cdot 10^{-5}$	$[\mathrm{kg} \cdot \mathrm{m}^2]$
Nominal length ℓ_0	0.065	[m]
Distance r_{B}	0.0126	[m]

Research on other soft robotic systems shows that the rotational kinetic energy is often much smaller than the translational kinetic energy, and the rotational kinetic energy can be neglected in some cases [10], [35]. To investigate the influence of the kinetic energy on our system, the kinetic energy (3.21) has been calculated from experimental data of a lateral swing of the actuator. The translational and rotational parts of this kinetic energy are shown separately in Figure 3.7. As expected, the rotational part is lower than the translational part, but the rotational part is still significant. This is further illustrated in Figure 3.8, which shows the share of the rotational kinetic energy with respect to the total kinetic energy. This share is around 20%, which is much larger than the approximately 8% observed in [35]. Arguably, it could be said that for smaller soft robotic systems, the rotational kinetic energy has a bigger influence than for bigger soft robots. Another interesting aspect is that the profile of both the translational and kinetic energy is very similar, and that the share in Figure 3.8 is near constant when the total kinetic energy becomes lower. If the computation time has to be reduced one could also approximate the rotational kinetic energy by multiplying the translational kinetic energy with a constant.

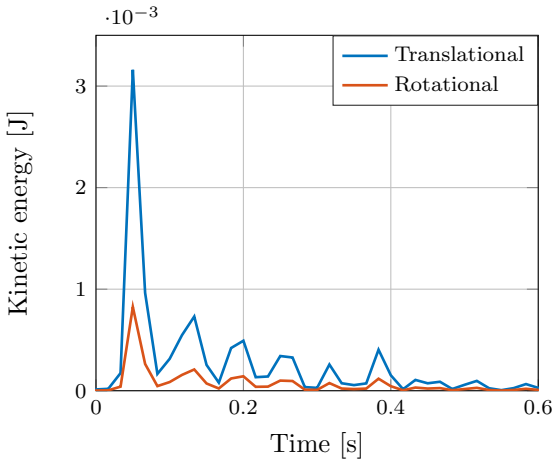


Figure 3.7: Comparison of translational and rotational kinetic energy, calculated with experimental data.

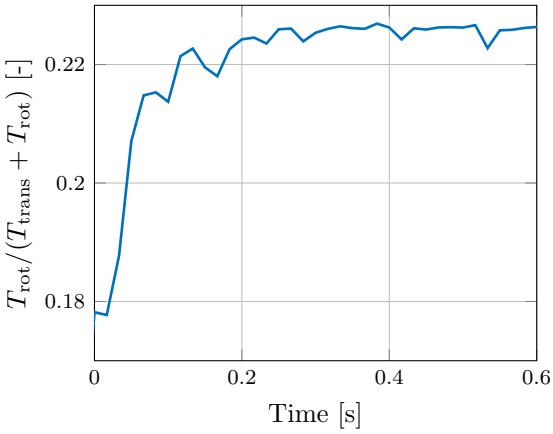


Figure 3.8: Share of the rotational kinetic energy with respect to the total kinetic energy, corresponding to Figure 3.7.

The longitudinal stiffness of the actuator has been identified by suspending weights from the bottom of the actuator and measuring the corresponding displacement. The resulting displacement measurements are shown in Figure 3.9, as well as a linear fit and two nonlinear polynomial fits. The curve fitting is done with least-squares regression. The measurements imply that the stiffness is nonlinear in nature. By adding the constraint that zero displacement equals zero force you can prevent a jump in spring force when the displacement crosses zero. The stiffness curves corresponding to the curve fits in Figure 3.9 are shown in Figure 3.10. This graph shows a significant difference between the linear and nonlinear stiffness models, which illustrates that a constant linear stiffness is a poor approximation for the longitudinal stiffness of a soft robotic system.

It should be noted that the forces and stiffnesses presented here are for the full actuator. For the stiffness of one bellow the values are simply divided by two. The stiffness as a function of bellow length $k(q_{ik})$ can be included in the model in the bellow spring force as

$$F_{\text{B}i,\text{spr}}(q_i) = k(q_i)\,(\ell_0 - q_i)\,. \tag{3.38}$$

Linear damping parameters can be identified by keeping the actuator at an angle and subsequently releasing it. This lateral swing will decay, and from this decay damping parameters can be ex-

tracted. By assuming the damping is viscous and linear, and because the system is underdamped, it is possible to obtain the damping ratio ζ with the logarithmic decrement δ. The logarithmic decrement is the natural logarithm of the ratio of the amplitudes of consecutive peaks. For the length of bellow q_1, the logarithmic decrement can be defined as [36]

$$\delta = \frac{1}{N} \ln \frac{q_1(t)}{q_1(t + NT)}, \tag{3.39}$$

where N is the amount of periods between peaks and T is the period time. From δ the damping ratio can then be determined by

$$\zeta = \frac{\delta}{\sqrt{\delta^2 + 4\pi^2}}. \tag{3.40}$$

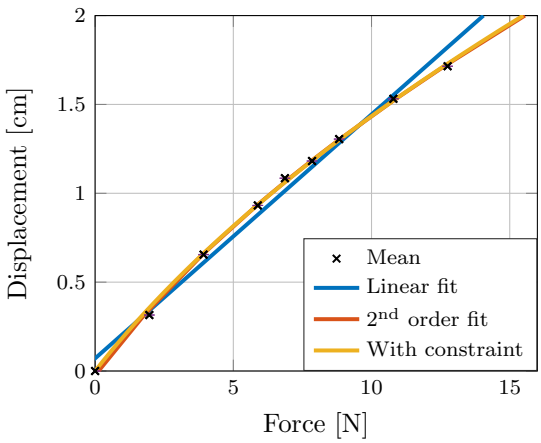


Figure 3.9: Measurement points and curve fits corresponding to force and displacement. Error bars represent a 95% confidence interval.

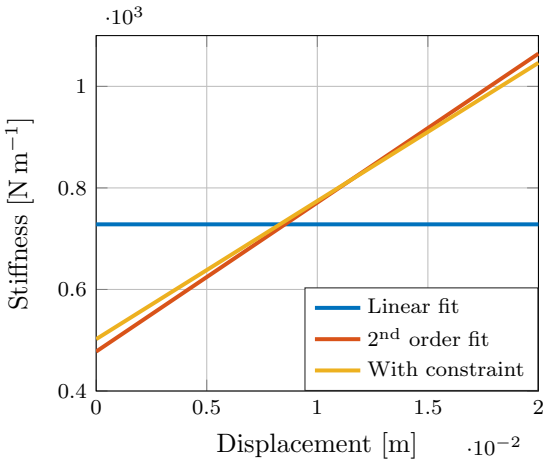


Figure 3.10: Stiffness as a function of displacement.

The lateral swing measurements are carried out by pulling the actuator head to the side, and abruptly releasing it. The pumps are disconnected in this experiment. The angle and bellow length $q_1(t)$ of one such measurement are shown in Figures 3.11 and 3.12. Three of these measurements have been done, and the decrement is calculated multiple times. The damping ratio depends on the timing of the data, therefore the non-filtered camera data is used (as discussed in Section 2.2.2) which does not have the induced delay due to the lowpass filter. As a downside, this camera data is quite rough and contains outliers, visible in Figure 3.12. For a more accurate estimation of the damping, these outliers are rejected when calculating the decrement. For example, only the peaks denoted by the orange markers in Figure 3.12 are used. Using the average of all the calculated decrements in equation (3.40) results in a damping ratio $\zeta = 0.0513$. The damping force can then be calculated with

$$F_{\mathrm{B}i,\mathrm{dmp}}(q_i, \dot{q}_i) = 2\zeta\dot{q}_i\sqrt{\frac{1}{2}mk(q_i)}, \tag{3.41}$$

with the stiffness function $k(q_i)$ as derived from Figure 3.10.

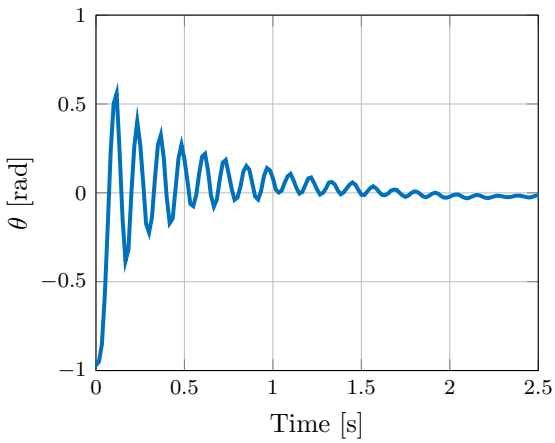


Figure 3.11: Actuator angle response after release, unactuated.

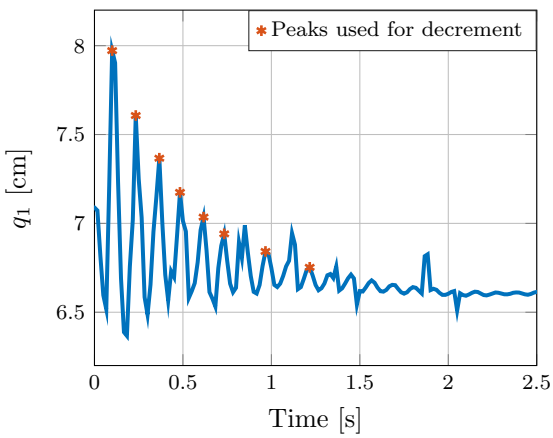


Figure 3.12: Bellow q_1 length after release, unactuated. The peaks used to determine the decrement are denoted by the orange markers.

Now that the parameters have been estimated, simulations can be performed with the equations of motion. The model has been implemented in MATLAB. The equations of motion are derived with the symbolic toolbox. These symbolic equations are then converted into a function file. The solver `ode45` utilizes this function file to obtain simulation results. Solver settings are default, with relative tolerance 10^{-3} and absolute tolerance 10^{-6}.

In order to determine the accuracy of the model, simulations will be compared to experimental data. To this end, experimental data of a lateral swing is used similar to the experiment of Figures 3.11 and 3.12, so the pumps are disconnected and there are no motor inputs. To make a fair comparison, the initial conditions for the simulations are taken equal to the initial conditions in the experiment. The lateral swing experiment is unactuated, so the force input $\boldsymbol{\tau}(\boldsymbol{p}(t)) = \mathbf{0}$. This yields a computationally efficient, open loop simulation with zero input that gives the bellow lengths $\boldsymbol{q}$ (and its derivatives $\dot{\boldsymbol{q}}$) as output.

For one such lateral swing experiment, a comparison between the experimental and simulation results of the bellow length q_1 is shown in Figure 3.13, the comparison for q_2 is shown in Figure 3.14. The curvature angle is shown in Figure 3.15. These graphs show a big discrepancy between the experiment and simulation. First, the oscillation frequency of the simulation is higher. Also, the simulated bellow lengths q_1 and q_2 show some dynamics that are not present in the experiment, and the amplitude is higher in the simulation. The error in the bellow lengths varies from approximately -1 to 1 cm. Considering that the range of motion in the experiments is in the 1 cm range, this error is substantial. If we would look at a longer time scale, when the steady state equilibrium position is reached, a small static error is observed in the range of 0.5 mm. This error is insignificant compared to the error during motion, and is arguably caused by sensor inaccuracies.

Another interesting anomaly is that in the experiment the oscillations do not occur around the nominal bellow length and the zero degree point, but instead there is an offset in the oscillations which does converge to the nominal values after around 1 second. Multiple similar experiments have been conducted to investigate if this behaviour is always present, which it is. Likely reasons for this discrepancy are the discretized mass or an inaccurate material model. It is anticipated that the oscillation offset is caused by visco-elastic material behaviour which is not incorporated in the model. Also, it has been assumed that the actuator stays in plane during motion. From angle measurements it can be concluded that the lateral swing also induces a small motion out of plane which is not accounted for in the model.

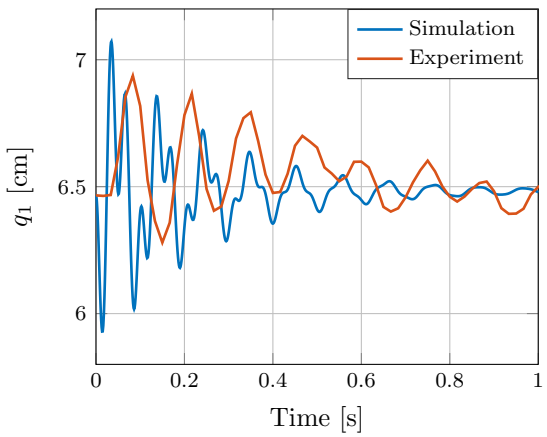


Figure 3.13: Comparison between simulation results and experimental data of the bellow length q_1.

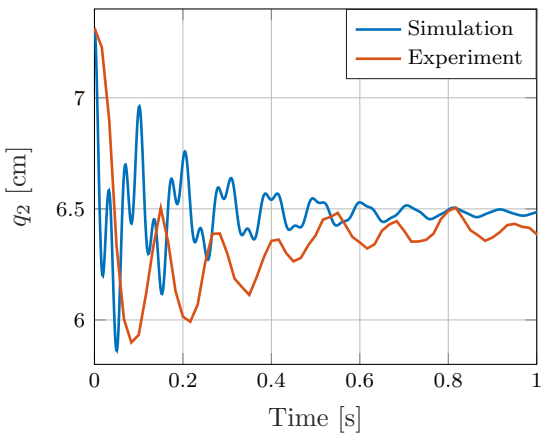


Figure 3.14: Comparison between simulation results and experimental data of the bellow length q_2.

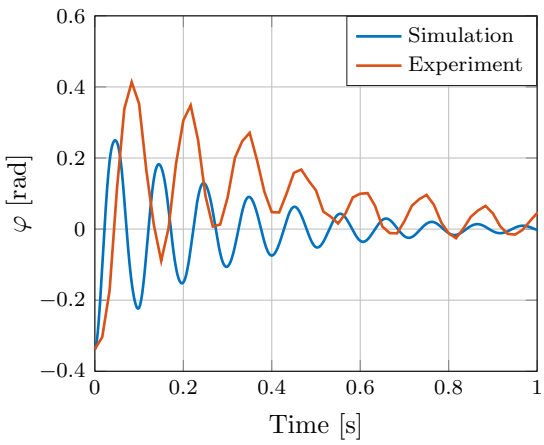


Figure 3.15: Comparison between simulation results and experimental data of the curvature angle φ.

3.4 Model improvements

As Figures 3.13-3.15 indicate, the model does not match experimental results well. Therefore the model is expanded with the rotational spring moment and visco-elastic creep.

As a first addition, the rotational spring momentum that has been omitted earlier is added to the potential energy to obtain [10]

$$U = {}^{\mathrm{w}}\boldsymbol{F}_g^{\top}\,{}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{S}} + \sum_{\chi=1}^{2} {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{spr}}^{\top}\,{}^{\mathrm{w}}\boldsymbol{r}_{\mathrm{B}\chi} + M_{\mathrm{bnd}}(\boldsymbol{q}). \tag{3.42}$$

This rotational spring momentum M_{bnd}, a function of the bending angle φ via the actuator states $\boldsymbol{q}$, models the lateral stiffness induced by the width of the bellows. With this new potential equation, the force vector $\boldsymbol{N}$ becomes

$$\boldsymbol{N}_{(r)} = \frac{\partial M_{\mathrm{bnd}}}{\partial q_r} - m\,{}^{\mathrm{w}}\boldsymbol{g}^{\top}\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_r}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{S}} + \sum_{\chi=1}^{2}\left({}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{spr}}^{\top} + {}^{\mathrm{w}}\boldsymbol{F}_{\mathrm{B}\chi,\mathrm{dmp}}^{\top}\right)\frac{\partial\,{}^{\mathrm{w}}\mathbf{H}_{\mathrm{h}}}{\partial q_r}\,{}^{\mathrm{h}}\boldsymbol{r}_{\mathrm{B}\chi}. \tag{3.43}$$

with $r = 1, 2$ the row of $\boldsymbol{N}$. This momentum function M_{bnd} has to be identified first before it can be implemented. To this end, experiments have been conducted to determine the steady state orientation behaviour for various motor voltages, see Figure 3.16. For these experiments the motor input u_1 is equal to zero, and the motor input u_2 is fixed, such that the actuator bends to a steady state orientation. The longitudinal spring force corresponding to the steady state angle in Figure 3.16 calculated with (3.38) is shown in Figure 3.17. It is assumed that the rotational spring momentum force has a similar nonlinear profile. Therefore, the following equation for the derivative of the rotational spring momentum is proposed

$$\frac{\partial M_{\text{bnd}}}{\partial q_i} = k_{\text{bnd}} \text{sgn}\left(q_i - q_j\right) \varphi(\boldsymbol{q})^2 \quad \text{with } \{j \in \{1, 2\} \mid j \neq i\}, \tag{3.44}$$

with k_{bnd} a bending stiffness parameter that has to be estimated.

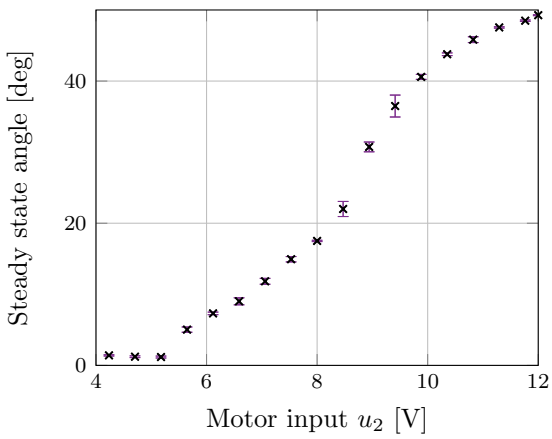


Figure 3.16: Steady state actuator angle measurements versus the motor input voltage. The error bars indicate the 95% confidence interval.

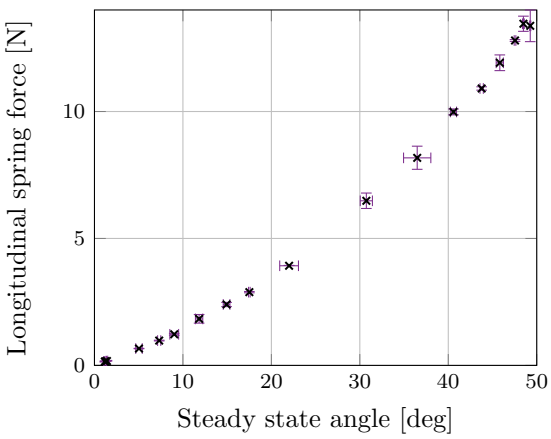


Figure 3.17: Longitudinal stiffness force corresponding to Figure 3.16 if the bending momentum would not exist, calculated with (3.38).

As a second addition, a visco-elastic creep model has been added to the dynamic model. A constant stress applied to a visco-elastic material results in a time-dependent strain. Figure 3.15 shows an example of this effect: the actuator does not oscillate around the zero degrees point directly, but it takes time to get to that point. The response of the creeping strain $\boldsymbol{z}(t)$ can be expressed as

$$\dot{\boldsymbol{z}} = -\lambda_1 \boldsymbol{z} + \lambda_2 \dot{\boldsymbol{q}}, \tag{3.45}$$

with λ_1 the relaxation rate, λ_2 a tuning parameter, and the strain rate of the bellows $\dot{\boldsymbol{q}}$. This expression is added to the dynamic model (3.33) to obtain

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}(\boldsymbol{p}) + \lambda_3 \boldsymbol{z}, \tag{3.46}$$

with λ_3 the creep compliance coefficient. The creep parameters are difficult to measure, so instead they are estimated with Simulink's Parameter Estimation Toolbox, together with the bending stiffness parameter k_{bnd}. During the estimation process, a small discrepancy between the measured longitudinal stiffness and damping parameters has been found. Therefore, the empirical longitudinal stiffness model (3.38) and damping model (3.41) have been updated to

$$F_{\text{B}i,\text{spr}}(q_i) = c_{\text{spr}} k(q_i) \left(\ell_0 - q_i\right), \tag{3.47}$$

$$F_{\mathrm{B}i,\mathrm{dmp}}(q_i, \dot{q}_i) = 2c_{\mathrm{dmp}}\zeta\dot{q}_i\sqrt{\frac{1}{2}mk(q_i)}, \tag{3.48}$$

where c_{spr} and c_{dmp} are scaling parameters that have been used as input in Simulink's Parameter Estimation Toolbox as well.

The parameter estimation process has been carried out as follows. Orientation data from three different experiments is imported into the toolbox. These experiments consist of holding the actuator at an angle and abruptly releasing it, similar to the experiments for determining the linear damping parameters. The initial conditions from the experiments are employed as initial conditions in the simulations. For the first iteration, an initial guess of the optimization parameters has been estimated. The optimization parameters are the creep parameters λ_1, λ_2, and λ_3, the bending stiffness parameter k_{bnd}, and the scaling parameters c_{spr} and c_{dmp}. Every optimization iteration, the model is simulated with the current parameter values. The task is then to minimize the error, in this case the difference between the measured orientation and the simulated orientation, with a non-linear least squares algorithm. The optimization process stops when a local minimum has been found. This process has been carried out iteratively multiple times for different initial guesses for the optimization parameters. The final optimization parameters are presented in Table 3.2. It is interesting to note that the scaling parameters c_{spr} and c_{dmp} are close to 1, indicating that the original longitudinal stiffness and damping estimation measurements are a good estimation.

Table 3.2: Final optimized parameters.

Parameter	Value	Unit
λ_1	-0.1047	$[\mathrm{s}^{-1}]$
λ_2	-14.98	$[\mathrm{kg} \cdot \mathrm{s}^{-2}]$
λ_3	-1.854	$[-]$
k_{bnd}	169.49	$[\mathrm{kg} \cdot \mathrm{m} \cdot \mathrm{s}^{-2}]$
c_{spr}	0.9102	$[-]$
c_{dmp}	1.0369	$[-]$

The experimental data of Figures 3.13-3.15 is again used to compare the model with experimental results, this time with the addition of the rotational bending momentum and creep. The results can be seen in Figures 3.18-3.20.

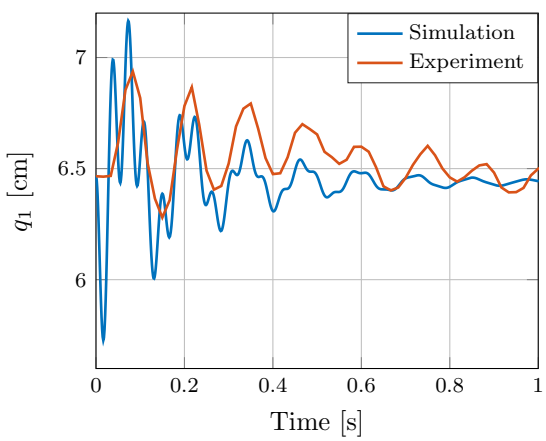


Figure 3.18: Comparison between simulation results and experimental data of the bellow length q_1.

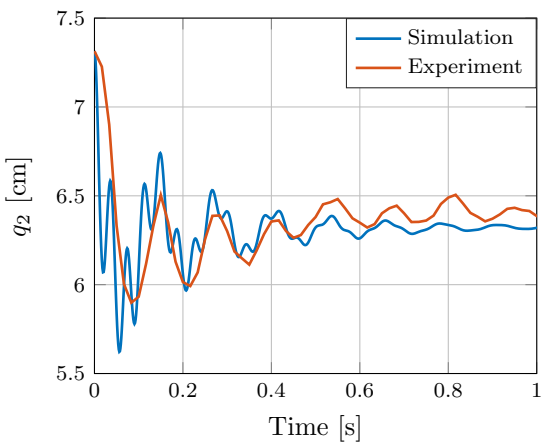


Figure 3.19: Comparison between simulation results and experimental data of the bellow length q_2.

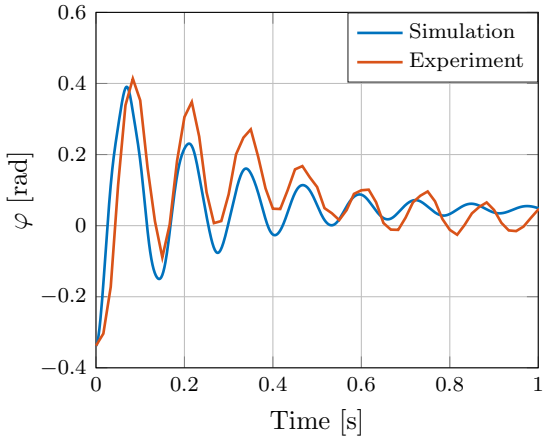


Figure 3.20: Comparison between simulation results and experimental data of the curvature angle φ.

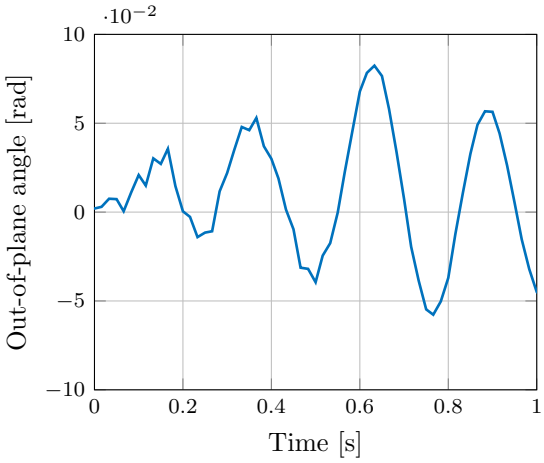


Figure 3.21: Out-of-plane angle, computed with a Mahony-Madgwick filter similar to (2.3), corresponding to the experimental data in Figures 3.18-3.20.

The discrepancy between the updated model and the experiment is smaller than the discrepancy between the original model and the experiment. A comparison between the error norms of the original model (the data in Figures 3.13-3.15) and the improved model (the data in Figures 3.18-3.20) is shown in Table 3.3. These values show a significant improvement, but there are still some considerable discrepancies. The oscillation frequency and decay rate of the model and experiment are very close. The biggest difference that is still observed are the higher frequency oscillations in the bellow lengths q_1 and q_2, that are observed on top of the 'main' oscillation, see Figures 3.18 and 3.19. The orientation, Figure 3.15, does not show these higher frequency oscillations and fits the experiment well. Part of the difference might be explained by the perpendicular motion that is not accounted for in the model. The out-of-plane bending corresponding to the experiments in Figures 3.18-3.20 is shown in Figure 3.21. The actuator oscillates between $-3.3°$ and $4.7°$. Not only does this influence the dynamics, this out-of-plane bending corresponds roughly to a out-of-plane motion of the actuator head, and thus the camera marker, between -1.9 mm and 2.6 mm. Thus, this motion influences both the dynamics and sensor accuracy.

Table 3.3: Error norms of the bellow lengths and orientation for the original model and the improved model, and the difference between them.

Norm	Original	Improved	Diff.
q_1 [mm]	17.8	13.1	26.2%
q_2 [mm]	23.2	15.0	35.3%
φ [rad]	1.40	0.629	55.0%

3.5 Pump model

The simulations so far do not include actuation forces. It is expected that during actuation the movements of the soft robot are much slower, and it will be interesting to see if this influences the model fit. Hence, a pump model is developed to be able to do simulations with actuation. This model is based on the measurements of Section 2.3, where we demonstrate the saturation and first-order system characteristics of the pumps.

If a pressure model is used, the effective area of the bellows needs to be known as well to obtain the force. Therefore, we develop a model to directly convert PWM motor input to the pressure force. The derived pump model is shown in Figure 3.22, with the model being identical for both pumps. The model input is a user generated signal (or the output of a controller as will be discussed in 4). This signal is limited to the range 0-255 to obtain the PWM signal defined as u_i, with i denoting the motor/pump corresponding to bellow q_i. Next, the function $g(u)$ converts PWM to force. This function is derived from the steady state measurements of Figure 2.11 (and the polynomial fit of equation (2.13) as follows. Simulations are performed with a fixed motor input in the range 0-255. This input is multiplied with a gain, and substituted in the model as the actuation forces $F_{\mathrm{B1,act}}$ and $F_{\mathrm{B2,act}}$ (so both bellows have the same actuation force). This gain is iteratively tweaked until the steady state elongation of both the experiment and simulation matches. By doing this for multiple input values, we obtain the function $g(u_i)$ that maps steady state PWM value to steady state actuation force,

$$g(u_i) = -5.686 \cdot 10^{-6} u_i^4 + 2.937 \cdot 10^{-3} u_i^3 - 0.4189 u_i^2 + 18.21 u_i, \tag{3.49}$$

which is a least-squares fourth order polynomial fit. To incorporate the time-dependent, first order system characteristics of the pump the transfer function $\frac{1}{\tau_{\mathrm{act}} s+1}$ is introduced, with τ_{act} a time constant that determines the settling time. The value of τ_{act} is estimated with the data in Figure 2.12 to obtain $\tau_{\mathrm{act}} = 0.7$. To validate the model, simulations are done with the same initial conditions as in Figure 2.12, which produces the simulation results in Figure 3.23. In general, the time dependent behaviour seems to match well. However, there is a discrepancy between the steady state actuator lengths of the experiments and simulations. This is mostly prevalent for the low voltage actuation, i.e. the 6 V (equal to a PWM value of 127.5).

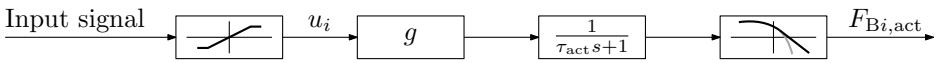


Figure 3.22: Block scheme of the pump model. The fourth block is a representation of the rate limiter.

As discussed in Section 2.1, small leakage orifices are placed between the pumps and the soft robot to ensure the soft robot can deflate. There is no active deflation, thus there is a limit to the negative rate of change of the bellow pressure. This aspect is modelled with a rate limiter which limits the derivative of the force when the force is decreasing (which can be seen as a one-sided saturation of the derivative of the force). The rate limit value has been estimated with an experiment where first the motor input of both motors is 12 V, and at $t = 5$ s the motors are turned off, see Figure 3.24. It takes approximately 5 seconds for the actuator to deflate and return to its nominal length. This is roughly twice as long as it took the actuator to reach its inflated state. Using this data, the lower limit on the derivative $\frac{\mathrm{d}F_{\mathrm{B}i,\mathrm{act}}}{\mathrm{d}t}$ is set at -2 N/s, and the resulting simulation results can also be seen in Figure 3.24. No upper limit on the force derivative is imposed. In the simulation data an overshoot is visible when the actuator returns to its nominal length around $t = 10$ s (the backbone length does converge to the nominal value eventually, but this outside the plot range). This overshoot is arguably caused by the creep model, when running the same simulation without the creep model the overshoot does not occur. This implies that the creep model needs improvement. Nevertheless, the pump model generally corresponds to the experimental data, and we are left with the final estimation of the bellow forces $F_{\mathrm{B}i,\mathrm{act}}$.

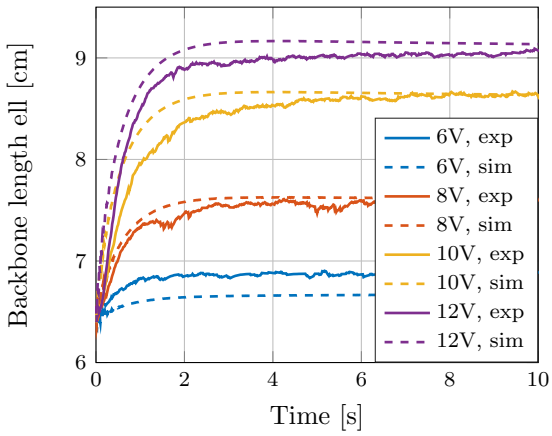


Figure 3.23: Experimental results with various constant motor input levels, and the corresponding simulation results with the pump model of Figure 3.22.

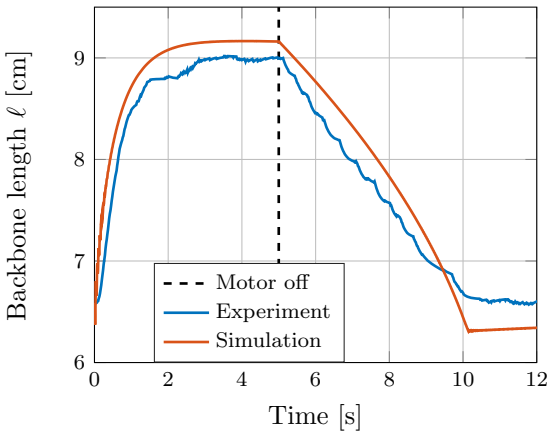


Figure 3.24: Experiment and simulation with a constant, maximum motor input for $0 < t < 5$. At $t = 5$ the motor input is set to zero. Motor input is equal for both motors/pumps.

Summary

Using the constant curvature approach, the kinematics of the actuator are defined. The actuator's equations of motion are derived with a lumped mass model. The spring and damping coefficients of this model are derived from measurement data. With these coefficients, there is a big discrepancy between the model and experimental data. This discrepancy is likely caused by the lumped mass approach and an a material model that does not take visco-elasticity into account. The model fit is improved by taking bending stiffness and creep into account. Due to the lumped mass approach and non-planar movement, there is still a notable discrepancy between the model and experiments. A pump model is introduced so that simulations with actuation are possible as well. This pump model also demonstrates that the creep model has inaccuracies.

Chapter 4

Closed-loop control of the actuator

Feedback control experiments will now be performed on the experimental setup described in Chapter 2. For the control process we first impose a reference signal on the actuator states, i.e. the bellow lengths. The control goal is then to find a motor input such that the actuator state errors converge to zero. First just the orientation of the actuator head will be controlled, and ultimately we will control both the orientation and position of the actuator head. Various control strategies are proposed and investigated. Simulations with the controllers are performed for a further model validation.

4.1 Orientation control of the actuator

Before looking at full position and orientation control, we first limit ourselves to control of the orientation only, because the IMU data is less prone to noise than the camera data and it is not needed to incorporate the inverse kinematics. The control process is ran on-board the Arduino, at the same 60 Hz frequency as the sensor data rate, resulting in a discrete control system. Recalling the definition in Section 2.2.1, discretized states are denoted with the timestep subscript k. The output of the controllers presented in this chapter will be in the PWM value scale, and not the voltage scale, because this way the controller output can directly be used as motor input.

The system we want to control is the soft robot actuator together with the two motor-pump configurations. The pumps directly influence the bellow lengths $\boldsymbol{q}$, and therefore we want to impose a reference signal on $\boldsymbol{q}$. This reference signal, together with the measured bellow lengths, gives rise to a bellow length error signal. The control goal for the system is to find the motor inputs such that the bellow length error converges to zero. The control scheme for orientation control is shown in Figure 4.1. The input reference signal φ_{ref} is a predefined discrete time-dependent or constant signal. The measured deviation from this setpoint is obtained by subtracting the IMU angle $\hat{\varphi}$, computed with (2.3), from the reference angle. The corresponding discrete time control error vector $\boldsymbol{e}_{\varphi}$ at timestep k is defined as

$$\boldsymbol{e}_{\varphi,k} = h\left(\varphi_{\text{ref},k} - \hat{\varphi}_k\right) = \begin{bmatrix} \varphi_{\text{ref},k} - \hat{\varphi}_k \\ -\left(\varphi_{\text{ref},k} - \hat{\varphi}_k\right) \end{bmatrix}, \tag{4.1}$$

with the function $h : \mathbb{R} \to \mathbb{R}^2$ mapping the angle error to the actuator state errors. The function

h is chosen such that when $\hat{\varphi} < \varphi_{\text{ref}}$, the bellow length q_2 has to extend and q_1 has to decrease to reach the setpoint, and vice versa for $\hat{\varphi} > \varphi_{\text{ref}}$. The actual length of the actuator is not of importance for orientation control, only the ratio between q_1 and q_2. The error $\boldsymbol{e}_\varphi$ is then the input for the discrete feedback controller C. The controller output $\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top$ is subjected to a saturation corresponding to the 0 V to 12 V input range of the motors. The saturated controller output $\boldsymbol{u}_{\text{sat}}$ is subsequently used as input for the plant, where the plant comprises the motors, pumps, tubing, and soft robotic actuator. Controller outputs u_1 and u_2 correspond to the motor and pump driving q_1 and q_2, respectively. Finally, the orientation of the actuator is measured with the IMU to obtain the output $\hat{\varphi}$.

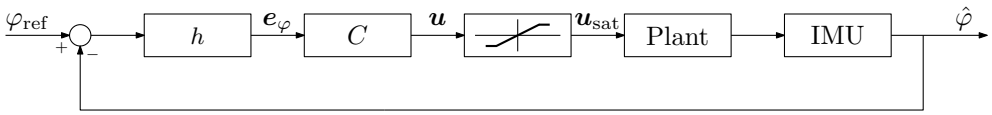


Figure 4.1: Block diagram of the orientation control scheme.

We first investigate the simplest control strategy, that is, a proportional controller. The controller output at timestep k is then equal to

$$\boldsymbol{u}_k = C_{\text{P}}\left(\boldsymbol{e}_{\varphi,k}\right) = K_p \boldsymbol{e}_{\varphi,k}, \tag{4.2}$$

with proportional gain K_p. To test this controller the constant reference signal $\varphi_{\text{ref}} = 0.7\,\text{rad}$ is introduced. The time response of the proportional controller to this fixed setpoint is shown in Figure 4.2 for multiple values of K_p. The actuator is not able to reach the setpoint for any of the applied gain values. A possible explanation for this behaviour might be that the controller output is linearly dependent on the error. When the error decreases, so does the controller output, and therefore the pump pressure decreases as well. This pressure loss then causes the error to increase, and as a result the actuator starts to oscillate. This gives rise to a fluctuation of the observed angle around an angle below the reference angle, with the angle dependent on the gain K_p. As discussed in Section 3.5, a small start-up delay can be observed.

The identical behaviour of the actuator at the start of the measurement in 4.2 is due to the saturation. Figure 4.3 shows the unsaturated motor output u_2. Even though the controller output varies for the different control gains, the motor input is limited to the 255 PWM limit (which corresponds to 12 V).

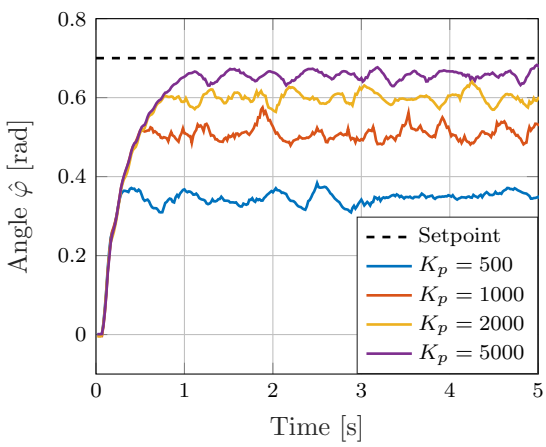


Figure 4.2: Step response with a proportional orientation controller for multiple gain values.

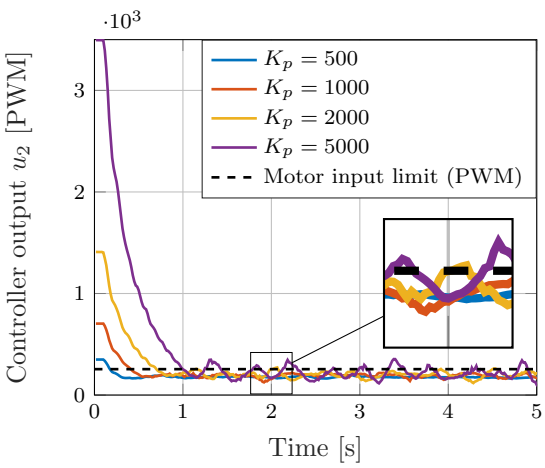


Figure 4.3: Controller output u_2, corresponding to the results

It is clear from Figure 4.2 that a proportional controller is not sufficient to reach the desired control objective. To overcome the steady state error, the controller is replaced with a PID controller. Now, consider the standard continuous PID controller with a continuous error signal $\boldsymbol{e}(t)$

$$\boldsymbol{u}(t) = K_p \boldsymbol{e}(t) + K_i \int_0^t \boldsymbol{e}(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}\boldsymbol{e}(t)}{\mathrm{d}t}, \tag{4.3}$$

where the controller output $\boldsymbol{u}$ depends on the error, the integral of the error and the derivative of the error via the K_p, K_i, and K_d, respectively. To implement this controller on the microcontroller, it has to be discretized first. By using the backward finite difference, and substituting the orientation error $\boldsymbol{e}_\varphi$, the discretized PID controller output becomes (see Appendix D for the derivation) [37]

$$\begin{aligned} \boldsymbol{u}_k &= C_{\mathrm{PID}}(\boldsymbol{u}_{k-1}, \boldsymbol{e}_{\varphi,k}, \boldsymbol{e}_{\varphi,k-1}, \boldsymbol{e}_{\varphi,k-2}) \\ &= \boldsymbol{u}_{k-1} + \left(K_p + \frac{1}{2}K_i\Delta t + \frac{K_d}{\Delta t}\right)\boldsymbol{e}_{\varphi,k} + \left(-K_p + \frac{1}{2}K_i\Delta t - \frac{2K_d}{\Delta t}\right)\boldsymbol{e}_{\varphi,k-1} + \frac{K_d}{\Delta t}\boldsymbol{e}_{\varphi,k-2}. \end{aligned} \tag{4.4}$$

Recalling the 60 Hz update rate of the microcontroller, the time step is equal to $\Delta t = 1/60 \approx 0.0167$ seconds. This PID controller is tested with the same constant reference orientation as used in Figure 4.2. By iteratively tuning the controller gains K_p, K_i, and K_d, the actuator is now able to reach the setpoint as can be seen in Figure 4.4.

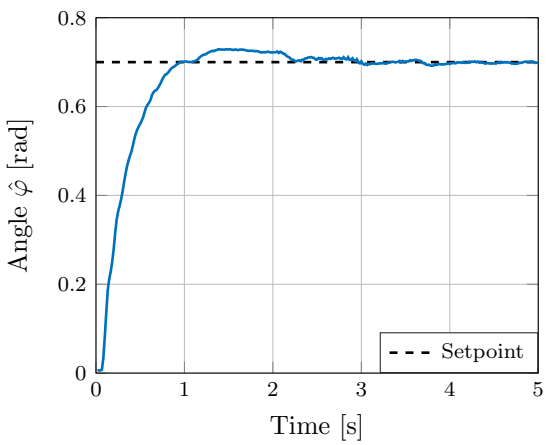


Figure 4.4: Orientation control with the PID controller, with control gains $K_p = 1000$, $K_i = 1000$, and $K_d = 10$.

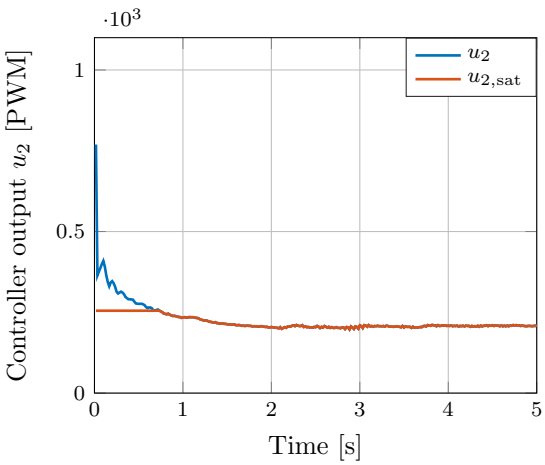


Figure 4.5: Controller output u_2 corresponding to Figure 4.4

The overshoot in Figure 4.4 is quite small, the highest observed angle is 0.7288 rad (41.76°). The settling time is approximately 2.3 seconds. By tuning the controller gains to decrease the overshoot, this time can probably be improved. The setpoint is first reached at approximately 0.95 seconds. This can not be improved notably by adjusting the controller due to the motor input limit. The unsaturated motor output u_2 can be seen in Figure 4.5. Up until $t = 0.73$ s the motor output is higher than the saturation limit, thus during this time the motor torque is already at its maximum level and cannot be increased by using a better controller. By tuning the gains the time it takes to reach the setpoint can thus be reduced a bit, but this would lead to a higher overshoot which is not beneficial for the overall settling time.

For a constant setpoint the PID controller works well. For saturated systems the integral term in the controller can cause bad performance due to integrator windup [38]. As an example, look at the same PID controller as in Figure 4.4, applied to a reference signal with a jump at $t = 5$ s, see

the blue line in Figure 4.7. After the jump, the error is negative and as such the control output is negative, see Figure 4.8. The motor input of the pumps cannot be negative, so the saturated control output is limited to $u_{2,\text{sat}} = 0$. As a result the error e_φ decreases slower than it would do ideally, i.e. if the pumps would also be able to decrease the pressure. This causes the integral term of the error to increase. So when the setpoint is reached, the controller output is still outside the feasible range due to the integral term. This causes a big overshoot, and long settling time.

Windup can be reduced by limiting the rate of change of the reference signal. But this might not be possible or desirable. Windup can be prevented altogether via a number of anti-windup strategies. A common method is integrator clamping [38]. To implement integrator clamping, the integral term of the controller has to be reset when the controller output is outside the feasible range. Every timestep k, the control output $\boldsymbol{u}_k$ is computed regularly. If $u_{i,k}$ is found to be outside the saturation limits, then the controller output at timestep k is calculated again but with $K_i = 0$. In other words, the error signal is fed through both a PID and a PD controller. A switching function then decides which controller output to use:

if $0 \leq u_{i,\text{PID}} \leq 255$ **then**
  $u_i = u_{i,\text{PID}}$
else
  $u_i = u_{i,\text{PD}}$
end if

The corresponding control scheme is shown in Figure 4.6. The controller with anti-windup technique, the orange line in Figure 4.7, has almost no overshoot after the jump. A downside of this anti-windup method is that it can make the system slower to respond. When the actuator is moving towards the setpoint before the jump, the controller with anti-windup is much slower.

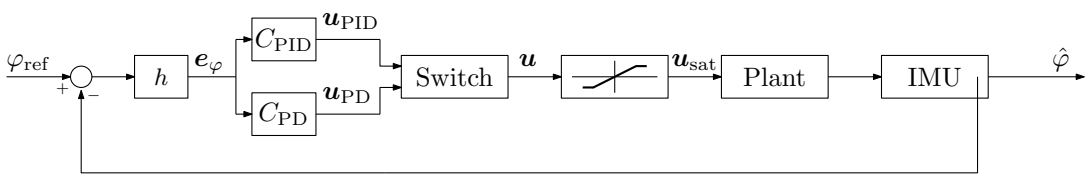


Figure 4.6: Control scheme with integrator clamping.

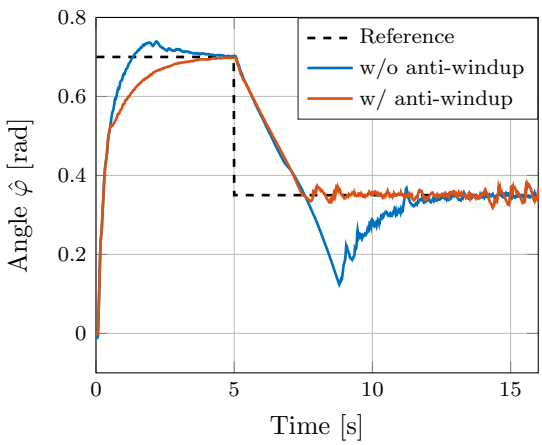


Figure 4.7: PID controller without and with anti-windup technique, tested on a reference signal with a jump. Control gains in both cases are $K_p = 1000$, $K_i = 1000$, and $K_d = 10$.

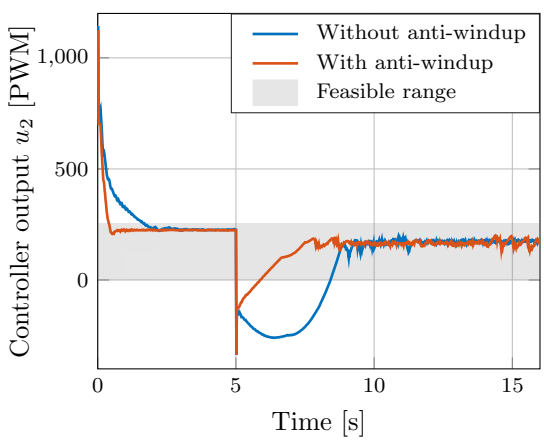


Figure 4.8: Controller output u_2 corresponding to Figure 4.7, with the grey area indicating the saturation range.

4.1.1 Orientation control in simulation

Using the model described in Chapter 3, we can perform simulations with controllers as well. Due to the fact that the pump model uses PWM signals as input, the discrete controllers presented here can be implemented in the simulations unaltered, with the same update rate of 60 Hz as on the Arduino. The same controllers as in Figures 4.2 and 4.4 are applied to compare the experiments with simulations. The results are shown in Figure 4.9. For the proportional controllers with gains $K_p = 500$, $K_p = 1000$ and $K_p = 2000$, the model and experiments show a reasonably good fit, at least for the steady state values. The settling time in the experiments is faster. For the controller with gain $K_p = 5000$ and the PID controller there is a much more significant discrepancy. It turns out that in the simulation with the PID controller the simulated motor input $u_{2,\text{sat}}$ is equal to the maximum value of 255 during the whole 5 second simulation, which indicates that the model cannot reach the setpoint. It is expected that this discrepancy can be attributed to a bad estimate of the rotational spring momentum (3.44).

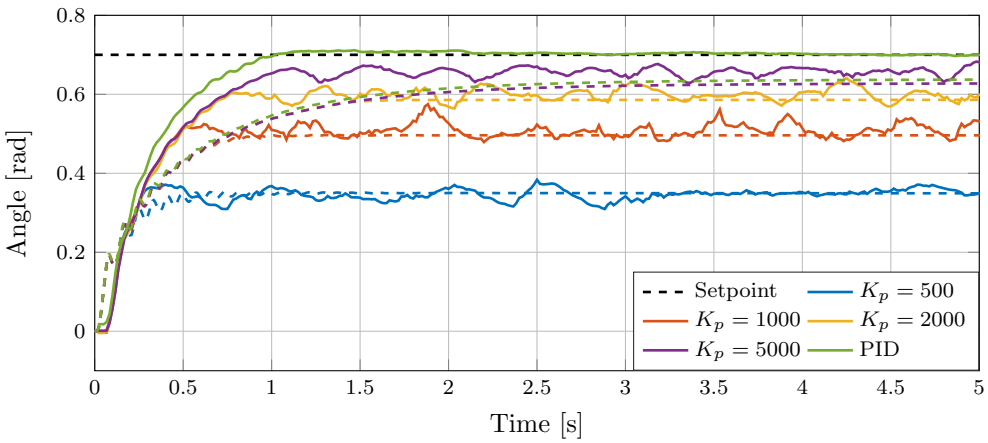


Figure 4.9: Comparison between the experimental data of Figures 4.2 and 4.4 (solid lines) with simulations (dashed lines).

4.2 Position and orientation control of the end effector

The orientation control results give a good insight in the behaviour of the system. We now move on to full position and orientation control of the end effector. This means that we want to control the task space coordinates $\begin{bmatrix} x & y & \varphi \end{bmatrix}^\top$, which requires the inverse kinematics. Similarly to the orientation control case, the control goal is to find the motor inputs such that the bellow length error converges to zero. The difference is that this time the actual length is of importance, and not just the length difference. Therefore, sensor data from both the IMU and the camera is required to estimate the bellow lengths $\boldsymbol{q}$.

As the first 5 seconds in Figure 4.7 show, the anti-windup controller makes the system slower. Both the anti-windup and no anti-windup cases are therefore non-desirable. Therefore, it might be beneficial to remove the integral term altogether. To overcome the static error when no integral term is used (see Figure 4.2), feedforward can be implemented instead of the integral term.

The control scheme for position and orientation control, with feedforward term, is shown in Figure 4.10. The control process is similar to the one in Figure 4.1. A reference signal for the bellow

lengths, $\boldsymbol{q}_{\text{ref}}$ can be chosen. The control error is defined as

$$\boldsymbol{e}_q = \boldsymbol{q}_{\text{ref}} - \hat{\boldsymbol{q}}. \tag{4.5}$$

Every timestep k, the discrete controller C computes the controller output $\boldsymbol{u}_k$, and the feedforward controller C_{ff} computes the feedforward output $\boldsymbol{u}_{\text{ff},k}$. The sum of $\boldsymbol{u}_k$ and $\boldsymbol{u}_{\text{ff},k}$ is fed through the saturation function to obtain the saturated controller output $\boldsymbol{u}_{\text{sat},k}$, which is ultimately sent to the motors as input. Both the Pixy2 and IMU are used to obtain the estimated task space coordinates $\hat{x}_k$, $\hat{y}_k$, and $\hat{\varphi}_k$ at timestep k. The task space coordinates are used to obtain an approximation of the bellow lengths, $\hat{\boldsymbol{q}}$, utilizing the Maclaurin series of the inverse kinematics equations (3.12).

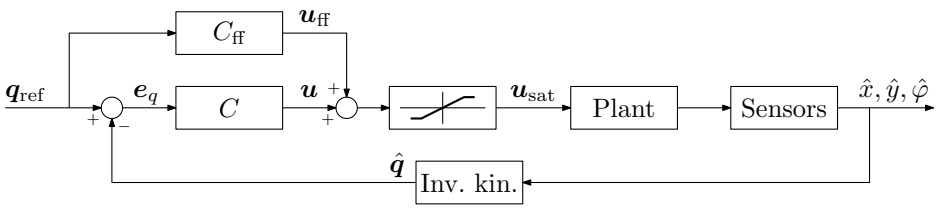


Figure 4.10: Block diagram of the full position and orientation control scheme, including feedforward.

A classical approach for the feedforward controller C_{ff} would be to use the derivative and double derivative of the reference trajectory, multiplied with feedforward gains, to approximate the inverse of the plant. Using this approach for our soft robot actuator has not led to a significant improvement of the tracking performance. Hence, a different approach is proposed depending on $\boldsymbol{q}$ itself, and not its derivatives. The proposed feedforward function is based on the experimental data regarding steady state elongation of Figure 2.11. A linear fit of this measurement data results in the feedforward function

$$u_{i,\text{ff}} = C_{\text{ff}}(q_{i,\text{ref}}) = \begin{cases} 0 & q_{i,\text{ref}} < 0.067 \\ \min(255,\ 5498.1(q_{i,\text{ref}} - 0.067) + 110) & q_{i,\text{ref}} \geq 0.067. \end{cases} \tag{4.6}$$

For a graphical rendition of the feedforward function, see Figure 4.11. For a reference bellow length $q_{i,\text{ref}} < 0.067$ the feedforward function is equal to 0, because this would mean that the actuator has to compress which is generally not possible. Because the the actuator has no steady state elongation for motor inputs below 100 PWM, for a reference bellow length $q_{i,\text{ref}} \geq 0.067$ the feedforward function has an offset of 110 PWM (slightly higher than 100 due to the linear fit). An upper limit is imposed at 255 PWM as that is the motor input limit. An explanation why this function is used now follows. First, let us consider a reference signal that comprises just elongation of the actuator, so $q_{1,\text{ref}} = q_{2,\text{ref}}$. From the steady state elongation measurements it is known that when the reference actuator length is in the range 65 mm to 91 mm, the pumps can deliver a high enough pressure level to reach that reference actuator length. The feedforward function converts the reference bellow length to the PWM value required to reach the reference bellow length if it were steady state. Therefore, it is expected that the static error will be zero if the reference length is chosen in the range 65 mm to 91 mm, when the feedforward function (4.6) is used in conjunction with a feedback controller where the integral term is omitted. The feedback is required to correct for disturbances. Also, because the feedforward is based on steady state behaviour, the feedback is required to correct for time-dependent behaviour.

To test the hypothesis that the static error is zero when the reference signal is composed of just elongation, the following time-dependent setpoint is proposed:

$$q_{1,\text{ref}}(t) = q_{2,\text{ref}}(t) = 0.0670 + 0.00276t^2 - 0.000368t^3. \tag{4.7}$$

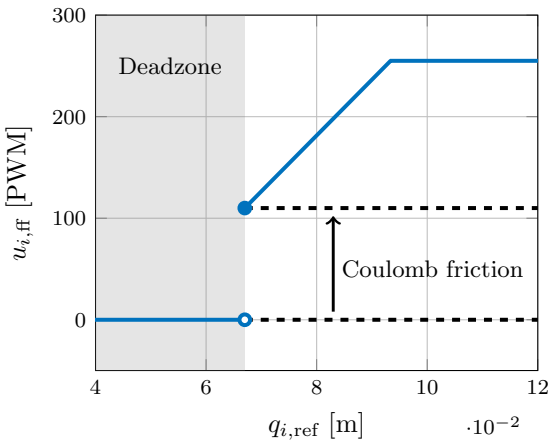


Figure 4.11: Feedforward prifle as a function of the reference bellow length.

The PID controller is reduced to a discrete PD controller

$$\begin{aligned} \boldsymbol{u}_k &= C_{\mathrm{PD}}(\boldsymbol{u}_{k-1}, \boldsymbol{e}_{q,k}, \boldsymbol{e}_{q,k-1}, \boldsymbol{e}_{q,k-2}) \\ &= \boldsymbol{u}_{k-1} + \left(K_p + \frac{K_d}{\Delta t}\right)\boldsymbol{e}_{q,k} + \left(-K_p - \frac{2K_d}{\Delta t}\right)\boldsymbol{e}_{q,k-1} + \frac{K_d}{\Delta t}\boldsymbol{e}_{q,k-2}. \end{aligned} \tag{4.8}$$

The blue line in Figure 4.12 shows the response of the actuator length to the reference (4.7) with both the feedback and feedforward controllers enabled. The setpoint tracking is decent, also illustrated by the error values in Figure 4.13, with a maximum error of 2.149×10^{-3} m. The static error, i.e. the error after approximately 6 seconds, is close to zero. The mean static error is -3.034×10^{-4} m for q_1 and -6.754×10^{-5} m for q_2. The error is thus not equal to zero, but close. This can be attributed to the fact that the data fit which is employed as feedforward function does not model the steady state elongation perfectly. Other factors to consider are slight differences between the pumps, flaws in the manufacturing process, and sensor inaccuracies. These factors can also explain the difference in length between q_1 and q_2.

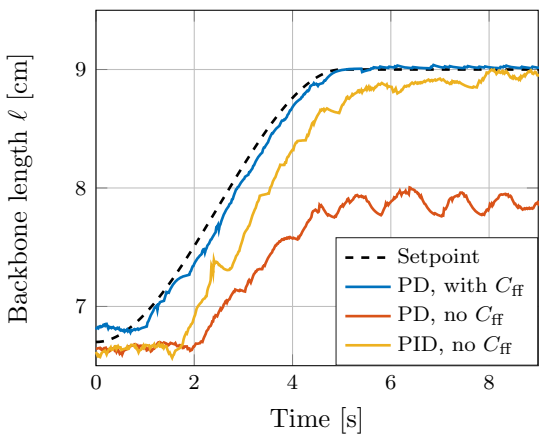


Figure 4.12: Comparison between a PD controller with feedforward, a PD controller without feedforward, and a PID controller without feedforward on a third order setpoint. The PD control gains are $K_p = 15000$ and $K_d = 150$. The PID gains are equal with the addition of $K_i = 8000$.

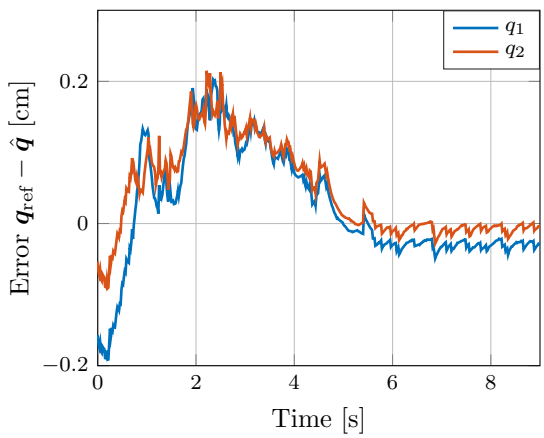


Figure 4.13: Bellow length error signals corresponding to the PD controller with feedforward in Figure 4.12.

The orange line in Figure 4.12 shows tracking results with the same PD controller but without the feedforward term. As expected, this controller does not reach the setpoint. The yellow line shows tracking results with a PID controller. Performance with this controller is worse then the PD controller with feedforward. This can be attributed to integral windup (at the start of the experiment, the PID controller only starts to do something after almost 2 seconds).

The feedforward function is based on the case with elongation only. When a setpoint with bending is considered, the feedforward function is not an accurate fit for the required motor input corresponding to the reference signal. It is expected that together with feedback control, we can still achieve reasonable tracking performance with the feedforward model.

To move on from the elongation-only setpoint, let us now consider a new setpoint that varies in both orientation and position. To this end, an elliptical setpoint in the task space is proposed, which is governed by the equations

$$\boldsymbol{x}_{\text{ref}}(t) = \begin{cases} x_{\text{ref}}(t) = \alpha \cos\left(2\pi \dfrac{t}{T_{\text{ell}}} + \dfrac{1}{2}\pi\right), & (4.9a) \\ y_{\text{ref}}(t) = y_{\text{offset}} + \beta \sin\left(2\pi \dfrac{t}{T_{\text{ell}}} + \dfrac{1}{2}\pi\right), & (4.9b) \\ \varphi_{\text{ref}}(t) = -\tan\left(\dfrac{x_{\text{ref}}(t)}{y_{\text{ref}}(t)}\right), & (4.9c) \end{cases}$$

with α and β the semi-major and semi-minor axis of the ellipse, respectively, T_{ell} the period of the ellipse, and y_{offset} a constant offset in the y-direction. The cosine and sine contain a phase shift of $\frac{1}{2}\pi$ that causes the elliptical reference signal to start at its top most position in the x, y-plane. The orientation $\varphi_{\text{ref}}(t)$ is chosen such that it is the angle between the base of the actuator $(0, 0)$ and the reference position $(x_{\text{ref}}(t), y_{\text{ref}}(t))$. By choosing the constants $\alpha = 0.015$, $\beta = 0.010$, $T_{\text{ell}} = 20$, and $y_{\text{offset}} = -0.078$, we obtain the counter-clockwise elliptical signal shown in Figure 4.14 in the x, y-plane. The ellipse dimensions are chosen such that the outer limits of the ellipse are in the actuator range. Due to the long ellipse period the required actuator motion is slow, which is required when the actuator has to deflate. The controller requires the reference signal in the actuator states, which is obtained by substituting (4.9) into (3.12). The resulting reference signal $\boldsymbol{q}_{\text{ref}}$ is shown in Figure 4.15.

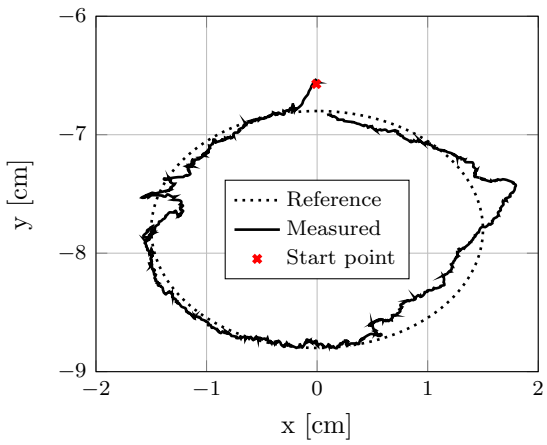


Figure 4.14: Tracking of the elliptical setpoint in the task space, with control gains $K_p = 10000$ and $K_d = 1000$.

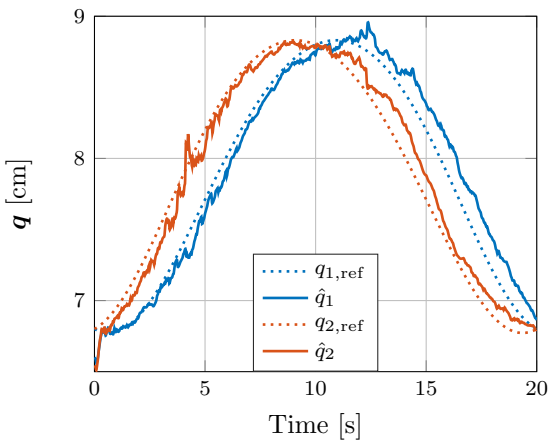


Figure 4.15: Tracking of the elliptical setpoint in the actuator space, with control gains $K_p = 10000$ and $K_d = 1000$.

Figure 4.14 shows the measured x- and y-positions of the actuator head. The red marker indicates the initial position of the actuator head at $t = 0$. The measured actuator states can be seen in Figure 4.15. Because the initial condition is not on the setpoint, the actuator first moves towards the setpoint and follows it reasonably well. Then, around $t = 4$ seconds, the soft robot starts to diverge from the setpoint, only to converge to the setpoint again around $t = 5$ seconds. The whole bottom left quadrant, tracking performance is quite good. Then for the right half of the ellipse, the bellow lengths are required to decrease to follow the reference, which leads to bad tracking performance because the pumps cannot decrease the pressure. The bellow lengths start to lag behind the reference lengths, as is evident from Figure 4.20.

The error $\boldsymbol{q}_{\text{ref}} - \hat{\boldsymbol{q}}$ corresponding to Figures 4.14 and 4.15 is shown in Figure 4.16. We can define four stages in this graph, denoted by the Roman numerals:

I. $\dot{\boldsymbol{q}}_{\text{ref}} > 0$ and $\ddot{\boldsymbol{q}}_{\text{ref}} > 0$,

II. $\dot{\boldsymbol{q}}_{\text{ref}} > 0$ and $\ddot{\boldsymbol{q}}_{\text{ref}} < 0$,

III. $\dot{\boldsymbol{q}}_{\text{ref}} < 0$ and $\ddot{\boldsymbol{q}}_{\text{ref}} < 0$,

IV. $\dot{\boldsymbol{q}}_{\text{ref}} < 0$ and $\ddot{\boldsymbol{q}}_{\text{ref}} > 0$.

In the transition regions between these four stages, the sign of $\dot{q}_{1,\text{ref}}$ and $\dot{q}_{2,\text{ref}}$ is not equal, and/or the sign of $\ddot{q}_{1,\text{ref}}$ and $\ddot{q}_{2,\text{ref}}$ is not equal. These four regions roughly correspond to the four quadrants of the ellipse. In phase I, the error is relatively small. Especially the error in q_1, which oscillates around zero. During this phase, the error increases over time, until it spikes in the transition between phase I and II. In this transition phase, in order to follow the setpoint, the acceleration of the bellow lengths has to adhere to $\ddot{q}_1 > 0$ and $\ddot{q}_2 < 0$. Since the two bellow lengths are influenced by each other, this difference in direction causes a big error spike. Then in phase II, the error decreases and tracking performance is good. Between phase II and III, the actuator is almost standing straight, which leads to good tracking as well. Then, in phase III, the bellow lengths have to decrease which cannot be done actively. As a result, the error increases as the actuator lengths start to lag behind the setpoint. In between phase III and IV, it is interesting to see that the error in q_1 increases, while the error in q_2 decreases. At last, in phase IV, the error decreases again even though the derivative $\dot{\boldsymbol{q}}_{\text{ref}} < 0$.

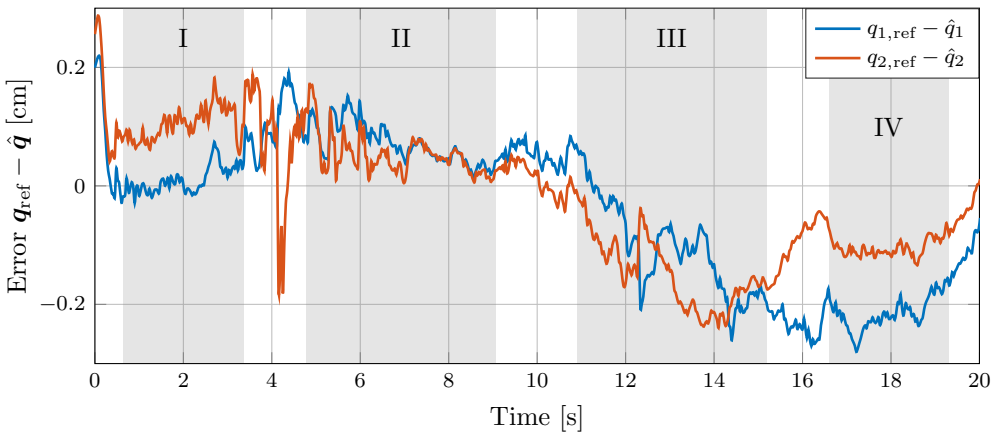


Figure 4.16: Error signal corresponding to Figure 4.15. The grey areas indicate the four different phases.

To explain the large errors in between phase I and II, and between phase III and IV, we look at the reference orientation and the measured orientation. as can be seen in Figures 4.17 and 4.18,

these large errors arise due to the change in direction of the reference angular rate. Thus, to avoid these large errors, the setpoint has to be chosen carefully.

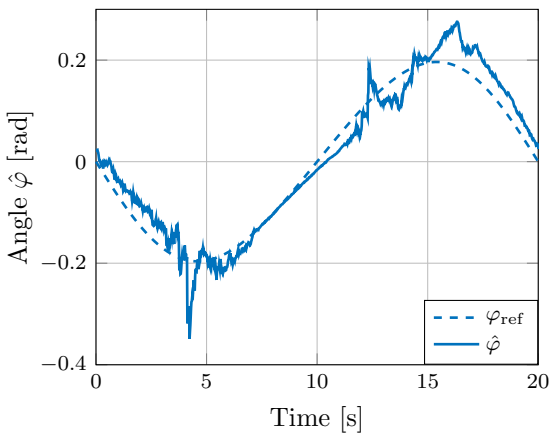


Figure 4.17: Orientation setpoint and experimental result corresponding to Figure 4.14.

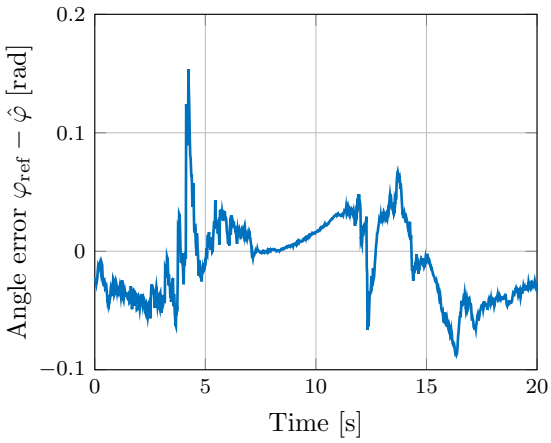


Figure 4.18: Error signal corresponding to Figure 4.17.

The elliptical reference signal used in Figures 4.14, 4.15, and 4.16 is presumably in the feasible range of the actuator, but it is difficult to tell for sure if the actuator states can actually reach this reference signal via pump actuation. This can be an extra factor in the measured error signal. A straightforward solution for designing a feasible reference signal, is to employ the measurement values of a previous control experiment as reference signal. Therefore, we use the measurement values observed in Figures 4.14, 4.15, and 4.16 as reference signal. Since it is not possible to store this data on the Arduino due to limited storage capacity, the setpoint data is sent to the Arduino with a frequency of 60 Hz via a serial connection with a PC. With the same control gains, the results can be seen in Figures 4.19, 4.20, and 4.21. The error profile is similar to the elliptical setpoint, albeit larger. Of course, this reference signal is not smooth like the elliptical profile, which is the likely reason for the larger error. Still, the similarity of the overall error profile indicates that the error spikes in Figure 4.16 are not necessarily caused by an infeasible reference signal, but rather by a bad performance controller or actuation limits.

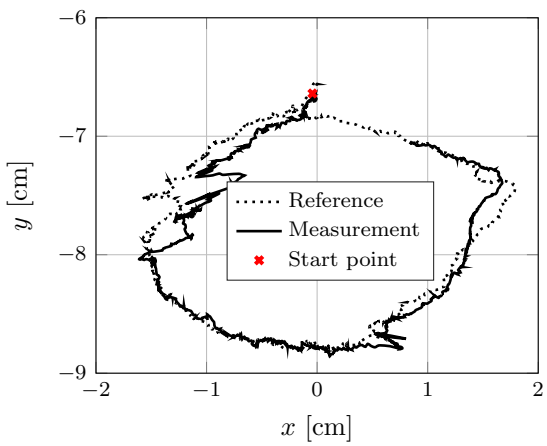


Figure 4.19: Tracking of the measured signal in Figure 4.14 in the task space, with control gains $K_p = 10000$ and $K_d = 1000$.

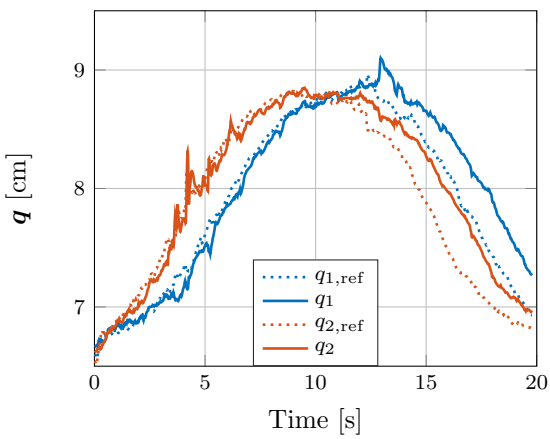


Figure 4.20: Tracking of the measured signal in Figure 4.14 in the actuator space, with control gains $K_p = 10000$ and $K_d = 1000$.

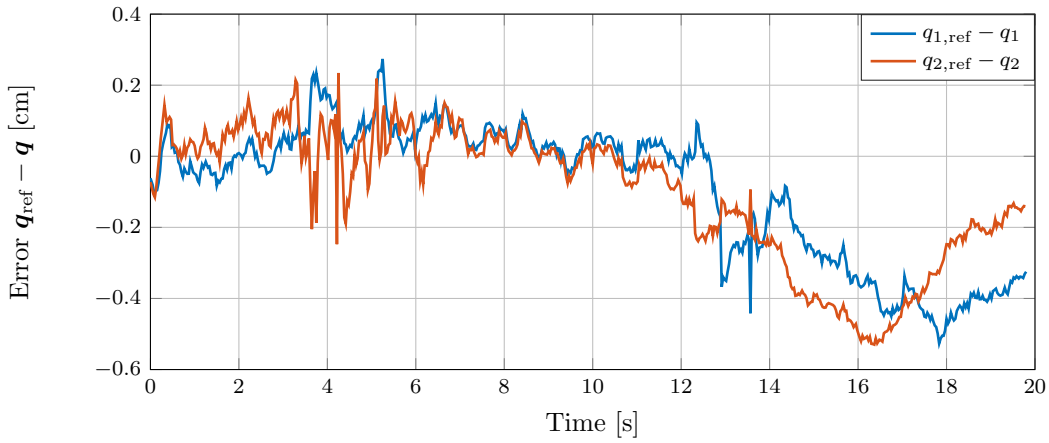


Figure 4.21: Error signal corresponding to Figure 4.20.

4.2.1 Position and orientation control in simulation

The dynamical model and pump model are also used to simulate the position and orientation controllers. First we look into the PD controller with feedforward as in Figure 4.12 for a setpoint with elongation only. Due to the way the model is set up, the feedforward function (4.6) can directly be implemented in the simulation model. The simulation result of the actuator length is shown in Figure 4.22, which uses the same setpoint, initial conditions and controller gains as in the experiment. Generally, the model and experiment show a good fit. In both cases the backbone length lags behind the setpoint a little. For the simulation, the steady state error for $t > 5$ is a bit larger. The biggest discrepancy occurs at the start, the simulation shows a large oscillation. This discrepancy is also illustrated in the difference graph in Figure 4.22. This oscillation is caused by the initial condition of the experiment. The initial actuator length is $\ell_0 = 0.0681$, which is quite a deviation from the nominal length. Due to the creep model and spring force, this causes a big oscillation in the simulation. The experimental data does not display this behaviour.

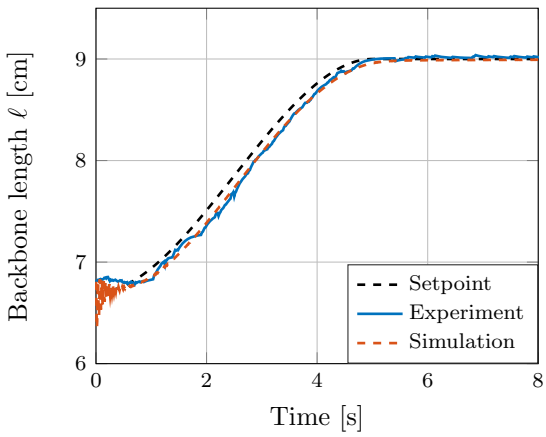


Figure 4.22: Comparison between experiment and simulation for the elongation only setpoint with a PD controller and feedforward, corresponding to Figure 4.12.

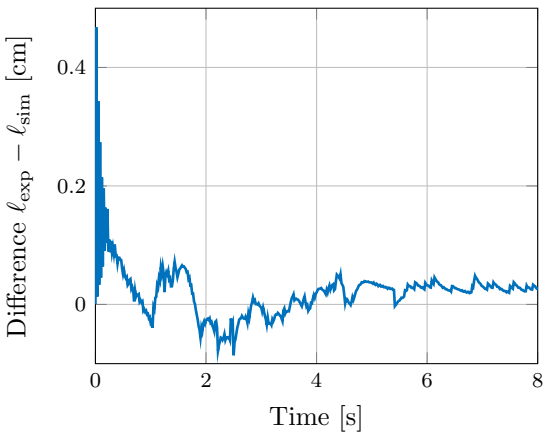


Figure 4.23: Difference between the simulation and experiment, corresponding to Figure 4.22.

Next, the elliptical reference signal is realized in the simulation model. The simulated task space

coordinates are shown in Figure 4.24, with the same setpoint, initial condtion and control gains as in Figure 4.14. At the start of the simulation, oscillations similar to the results in Figure 4.22 can be observed (see also the simulated bellow lengths in Figure 4.25). Then, remarkably, the actuator does not converge to the setpoint directly, which does happen in the experiment. However, the biggest difference can arguably be found in the behaviour around the 4-5 second mark. In the experiment, the actuator starts to wobble due to the change in the direction of the angle (see Figure 4.17). This behaviour is totally absent in the simulation. Continuing, in the bottom left quadrant of the ellipse the behaviour of the simulation and the experiment is similar. Next, after approximately 10 seconds the actuator has to deflate to follow the setpoint. Here again a big difference between the model and experiment arises. In the experiment, the actuator starts to diverge from the setpoint soon after the halfway point of the ellipse. When the direction of the angle changes, it makes a sharp turn and converges to the setpoint again. In the simulation, this

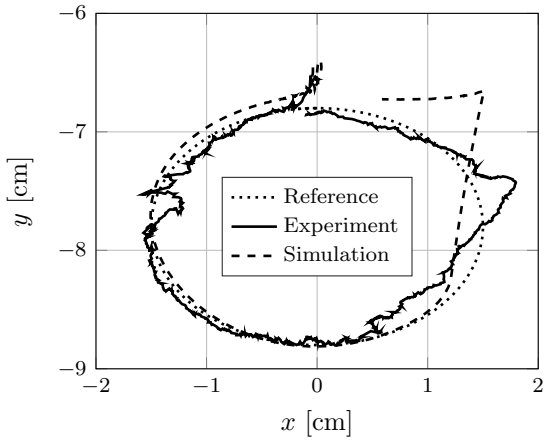


Figure 4.24: Comparison between the experiment and simulation for the elliptical setpoint with a PD controller and feedforward, corresponding to Figure 4.14.

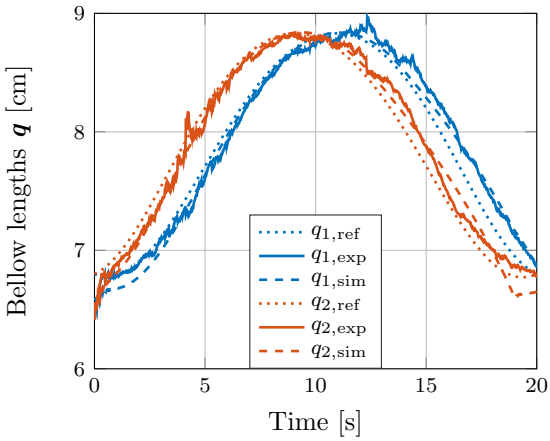


Figure 4.25: Bellow lengths corresponding to Figure 4.24.

divergence occurs later, and it makes a much steeper line. The sharp turn does happen as well, but at a different location. This indicates that the pump model does not capture the pump dynamics well when the soft robot has to deflate while simultaneously changing its orientation.

Summary

The experimental setup has been used to conduct experiments with feedback controllers, to control only the orientation and to control both the orientation and position. For orientation control, a PID controller is sufficient for a simple setpoint. When using a controller with an integral term, anti-windup can cause bad performance. Feedforward prevents this issue, and leads to better overall performance. For position and orientation control, an elongation only setpoint gives decent performance with a PD controller and feedforward. However, for an elliptical setpoint, performance is not as good, due to the change in orientation. Comparing the experiments to simulations, the orientation only and elongation only simulations match the experiments quite well, with a discrepancy in the bending stiffness. For the elliptical setpoint, the match between the simulation and experiment is much worse.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

In this thesis, a method to model and control planar soft robotic systems is explored. In order to validate the model and to test the control strategies in an experimental setting, a two-bellow soft robotic actuator has been developed. Air pumps are connected to these bellows that enable the soft robot to manoeuvre in a plane. This actuator, together with sensors, is placed in a test setup that allows us to measure the orientation and position of the soft robot. An inertial measurement unit is employed to measure the orientation of the actuator. The position in the actuation plane is tracked in real time with a camera system. With these two sensor systems it is possible to fully establish the position and orientation of the actuator.

In order to study the behaviour of the soft robot, first a kinematic model of the planar actuator is established using the constant curvature approach. A FEM simulation has shown that constant curvature is an adequate assumption. A dynamical model is derived with the Euler-Lagrange representation and the lumped mass approach, that enables us to simulate the actuator in real time. Damping and stiffness parameters are estimated with experimental data. With this initial model there is a substantial discrepancy between simulations and experiments, likely caused by the lumped mass approximation, out-of-plane motion in the experiments, and a material model that does not include visco-elastic effects. Creep and bending stiffness are added to the model, which reduces the model discrepancy by 26.2% to 55.0%.

The experimental setup is also used for closed-loop control of the orientation and position of the actuator. Our control goal is to find a motor input such that the error in bellow lengths, with respect to some setpoint, converge to zero. We limit ourselves to linear control techniques such as PD and PID controllers. As a first step we have looked at orientation control, and for a static setpoint a PID controller is able to reach the control goal. A comparison is made between the experiments and simulation for a static setpoint. For lower control gains, the model fit is reasonable. However, for higher gains, there is a noticable discrepancy which is caused by a bad estimation of the bending stiffness. Due to the input saturation of the pumps, the integral term in the controller can cause windup. By replacing the integral action with feedforward, windup is prevented and overall performance is improved. Furthermore we have looked at position control with an elongation only setpoint. A PD controller together with feedforward gives decent tracking

performance. An elliptical setpoint is introduced to study the performance of the actuator when both orientation and elongation have to change over time. The tracking result with a PD controller and feedforward is not as good a for the simpler setpoints. Big error spikes occur when the actuator has to the change the direction of the orientation. The need for deflation causes poor tracking performance as well because the bellows cannot be actively deflated. In simulations the elliptical setpoint shows very different behaviour, as the change in orientation does not cause any problems here. There also seems to be a mismatch when the actuator has to deflate, indicating that the pump model is inadequate.

Conclusively, the proposed controllers can achieve the control goal if an uncomplicated reference signal is chosen, such as a constant orientation setpoint or a setpoint that comprises just elongation of the actuator. With these simple setpoints, the model corresponds quite well with the experimental results. For a more complex reference signal, the controller performance is insufficient, and the simulations deviate noticeably from the experiments.

5.2 Recommendations

The current camera system has a low resolution and the update rate of 60 Hz is quite low, which leads to blocky data. A lowpass filter improves the camera data, but adds a small delay. A higher resolution, higher frequency camera system would be beneficial for the sensor accuracy. These specifications will lead to a more expensive camera system. Another possible way to increase the sensor performance is to introduce an observer such as a Kalman filter, so that the system states can be estimated with greater precision than by just measuring them individually.

The pumps in the setup have quite a small saturation range. The biggest issue with this saturation is that the pumps cannot be reversed, i.e. the pressure can not be actively decreased. If this were possible, it would greatly improve the tracking performance of the actuator. Instead of controlling the pumps directly, it could also be possible to control the pressure level directly using some kind of compressor and an orifice. This way the effects of pump dynamics can be reduced.

The soft robotic actuator presented in this thesis is only capable of planar actuation because it has just two bellows. A three-bellow soft robot is much more versatile and more promising for future applications due to its three dimensional motion capability. A disadvantage for three-bellow soft robots is that sensing is more difficult, and a single camera is not sufficient for position sensing. Multiple cameras would be required or another sensing method as described in Section 1.1.3. The model has to be converted to three dimensions as well. To fully embody the flexibility and adaptability of soft robotics, soft robotic actuators consisting of multiple, independently actuated, sections should be explored.

The lumped mass approximation, which is the basis for the model, can use some improvement. A possible way to improve the accuracy, while still holding on to the lumped mass approach, is to define multiple masses along the backbone curve that are connected to each other with springs and dampers, such that the mass distribution is more akin to the real distributed mass. Furthermore, a creep model has been added to model the visco-elastic material behaviour. This model shows mixed results. In the lateral swing experiments it improved the model fit significantly. In the pump model and closed-loop control simulations it decreased the accuracy of the model. Therefore, the visco-elastic material properties have to be studied in more detail by conducting experiments or with finite element analyses. Regarding the modelling part, judging from the simulation with the elliptical setpoint, the pump model is insufficient when the actuator has to deflate and change its orientation at the same time. Therefore the the rate limiter in the pump model should be replaced with a more sophisticated deflation model.

The controllers in this thesis are very simple linear controllers, and they do not take the coupling between the two bellow lengths into account. If the accuracy of the model can be increased, then it would be worthwhile to look into model-based controllers such as computed-torque control. Regarding the control process, choosing a reachable reference signal is of importance as well. Not all points in the task space are reachable due to actuation limits and the actuator geometry. Therefore it would be beneficial to determine a method to analyse the reachability of a reference signal.

Bibliography

[1] I. D. Walker. "Continuous Backbone "Continuum" Robot Manipulators". In: *ISRN Robotics* 2013 (2013), pp. 1–19. ISSN: 2090-8806. DOI: 10.5402/2013/726506. URL: http://www.hindawi.com/journals/isrn/2013/726506/.

[2] A. D. Marchese, R. Tedrake, and D. Rus. "Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator". In: *International Journal of Robotics Research* 35.8 (2016), pp. 1000–1019. ISSN: 17413176. DOI: 10.1177/0278364915587926.

[3] D. Rus and M. T. Tolley. "Design, fabrication and control of soft robots". In: *Nature* 521.7553 (2015), pp. 467–475. ISSN: 14764687. DOI: 10.1038/nature14543.

[4] A. Grzesiak, R. Becker, and A. Verl. "The Bionic Handling Assistant: a success story of additive manufacturing". In: *Assembly Automation* 31.4 (2011), pp. 329–333. ISSN: 0144-5154. DOI: 10.1108/01445151111172907. URL: http://www.emeraldinsight.com/doi/10.1108/01445151111172907.

[5] W. McMahan, V. Chitrakaran, M. Csencsits, et al. "Field trials and testing of the OcotArm continuum manipulator". In: *Proceedings of the 2006 IEEE international conference on robotics and automation (ICRA)* May (2006), pp. 2336–2341. DOI: 10.1109/ROBOT.2006.1642051.

[6] D. Trivedi, C. D. Rahn, W. M. Kier, et al. "Soft robotics: Biological inspiration, state of the art, and future research". In: *Applied Bionics and Biomechanics* 5.3 (2008), pp. 99–117. ISSN: 17542103. DOI: 10.1080/11762320802557865.

[7] R. J. Webster, III and B. A. Jones. "Design and kinematic modeling of constant curvature continuum robots: A review". In: *International Journal of Robotics Research* 29.13 (2010), pp. 1661–1683. ISSN: 02783649. DOI: 10.1177/0278364910368147.

[8] M. Rolf and J. J. Steil. "Constant curvature continuum kinematics as fast approximate model for the Bionic Handling Assistant". In: *IEEE International Conference on Intelligent Robots and Systems* (2012), pp. 3440–3446. ISSN: 21530858. DOI: 10.1109/IROS.2012.6385596.

[9] B. J. Caasenbrood, A. Y. Pogromsky, and H. Nijmeijer. "Soft Robot Manipulators: Design, Modeling, and Control". Master's thesis. Eindhoven University of Technology, 2017.

[10] V. Falkenhahn, T. Mahl, A. Hildebrandt, et al. "Dynamic Modeling of Bellows-Actuated Continuum Robots Using the Euler – Lagrange Formalism". In: *IEEE Transactions on Robotics* 31.6 (2015), pp. 1483–1496. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2496826.

[11] A. D. Marchese, K. Komorowski, C. D. Onal, et al. "Design and control of a soft and continuously deformable 2D robotic manipulation system". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2014), pp. 2189–2196. ISSN: 10504729. DOI: 10.1109/ICRA.2014.6907161.

[12] V. Falkenhahn, A. Hildebrandt, R. Neumann, et al. "Dynamic Control of the Bionic Handling Assistant". In: *IEEE/ASME Transactions on Mechatronics* 22.1 (2017), pp. 6–17. ISSN: 10834435. DOI: 10.1109/TMECH.2016.2605820.

[13] W. Felt. "Sensing Methods for Soft Robotics". Doctoral dissertation. University of Michigan, 2017.

[14] M. W. Hannan and I. D. Walker. "Real-time shape estimation for continuum robots using vision". In: *Robotica* 23.5 (2005), pp. 645–651. ISSN: 02635747. DOI: 10.1017/S0263574704001018.

[15] A. M. Franz, T. Haidegger, W. Birkfellner, et al. "Electromagnetic tracking in medicine -A review of technology, validation, and applications". In: *IEEE Transactions on Medical Imaging* 33.8 (2014), pp. 1702–1725. ISSN: 1558254X. DOI: 10.1109/TMI.2014.2321777.

[16] R. S. Penning, J. Jung, N. J. Ferrier, et al. "An Evaluation of Closed-Loop Control Options for Continuum Manipulators". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 5392–5397.

[17] B. Tondu, S. Ippolito, J. Guiochet, et al. "A Seven-degrees-of-freedom robot-arm driven by pneumatic artificial muscles for humanoid robots". In: *International Journal of Robotics Research* 24.4 (2005), pp. 257–274. ISSN: 02783649. DOI: 10.1177/0278364905052437.

[18] B. Verrelst, R. Van Ham, B. Vanderborght, et al. "The pneumatic biped "lucy" actuated with pleated pneumatic artificial muscles". In: *Autonomous Robots* 18.2 (2005), pp. 201–213. ISSN: 09295593. DOI: 10.1007/s10514-005-0726-x.

[19] C. Majidi, R. Kramer, and R. J. Wood. "A non-differential elastomer curvature sensor for softer-than-skin electronics". In: *Smart Materials and Structures* 20.10 (2011). ISSN: 09641726. DOI: 10.1088/0964-1726/20/10/105017.

[20] T. C. Searle, K. Althoefer, L. Seneviratne, et al. "An optical curvature sensor for flexible manipulators". In: *Proceedings - IEEE International Conference on Robotics and Automation* May (2013), pp. 4415–4420. ISSN: 10504729. DOI: 10.1109/ICRA.2013.6631203.

[21] C. To, T. L. Hellebrekers, and Y.-L. Park. "Highly stretchable optical sensors for pressure, strain, and curvature measurement". In: *IEEE International Conference on Intelligent Robots and Systems* 2015-Decem (2015), pp. 5898–5903. ISSN: 21530866. DOI: 10.1109/IROS.2015.7354215.

[22] H. Zhao, K. O'Brien, S. Li, et al. "Optoelectronically innervated soft prosthetic hand via stretchable optical waveguides". In: *Science Robotics* 1.1 (2016), eaai7529. ISSN: 2470-9476. DOI: 10.1126/scirobotics.aai7529. arXiv: arXiv:1204.6216v2. URL: http://robotics.sciencemag.org/lookup/doi/10.1126/scirobotics.aai7529.

[23] W. Felt, M. J. Telleria, T. F. Allen, et al. "An Inductance-Based Sensing System for Bellows-Driven Continuum Joints in Soft Robots". In: *Proceedings of Robotics: Science and Systems, Cambridge, Massachusetts* (2017). DOI: 10.15607/RSS.2017.XIII.027. URL: http://www.roboticsproceedings.org/rss13/p27.pdf.

[24] W. Felt, M. Suen, and C. D. Remy. "Sensing the motion of bellows through changes in mutual inductance". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 5252–5257.

[25] G. M. Whitesides. "Soft Robotics". In: *Angewandte Chemie International Edition* (2018), pp. 2–18. ISSN: 14337851. DOI: 10.1002/anie.201800907. URL: http://doi.wiley.com/10.1002/anie.201800907.

[26] C. Majidi. "Soft Robotics: A Perspective—Current Trends and Prospects for the Future". In: *Soft Robotics* 1.1 (2014), pp. 5–11. ISSN: 2169-5172. DOI: 10.1089/soro.2013.0001. URL: http://online.liebertpub.com/doi/abs/10.1089/soro.2013.0001.

[27] P. Polygerinos, B. Mosadegh, and A. Campo. *PneuNets Bending Actuators*. URL: https://softroboticstoolkit.com/book/pneunets-bending-actuator (visited on 09/24/2018).

[28] *Introducing Pixy2*. URL: https://pixycam.com/pixy2/ (visited on 04/09/2019).

[29] Bosch Sensortech. "BNO055, Intelligent 9-axis absolute orientation sensor". In: November (2014).

[30] OlliW. *IMU Data Fusing: Complementary, Kalman, and Mahony Filter*. 2013. URL: http://www.olliw.eu/2013/imu-data-fusing/ (visited on 11/27/2018).

[31] R. Mahony, T. Hamel, and J.-M. Pflimlin. "Nonlinear Complementary Filters on the Special Orthogonal Group". In: *IEEE Transactions on Automatic Control* 53.5 (2008), pp. 1203–1218.

[32] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22 (2000). ISSN: 01628828. DOI: 10.1109/34.888718. arXiv: arXiv:1011.1669v3. URL: http://research.microsoft.com/%CB%9Czhanghttp://research.microsoft.com/%CB%9Czhang.

[33] Z. Tang, R. Grompone von Gioi, P. Monasse, et al. *Self-consistency and universality of camera lens distortion models*. 2012.

[34] J. Weng, P. Cohen, and M. Herniou. "Camera Calibration with Distortion Models and Accuracy Evaluation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.10 (1992), pp. 965–980. ISSN: 0162-8828.

[35] V. Falkenhahn, T. Mahl, A. Hildebrandt, et al. "Dynamic modeling of constant curvature continuum robots using the Euler-Lagrange formalism". In: *IEEE International Conference on Intelligent Robots and Systems* Iros (2014), pp. 2428–2433. ISSN: 21530866. DOI: 10.1109/IROS.2014.6942892.

[36] D. J. Inman. *Vibration with Control*. John Wiley and Sons, 2006, pp. 19–22. ISBN: 0470010517. DOI: 10.1002/0470010533.

[37] P. Doležel, I. Taufer, and J. Mareš. "Discrete PID Controller Tuning Using Piecewise-Linear Neural Network". In: *Introduction to PID Controllers - Theory, Tuning and Application to Frontier Areas*. 2012, pp. 193–195. DOI: 10.5772/32478.

[38] A. Visioli. *Practical PID Control*. Springer, 2006, pp. 35–60. ISBN: 9781846285851. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

Appendix A

Offset for actuator head position

Let us denote $(\hat{x}, \hat{y})$ the measured marker coordinates in the world frame. We are interested in the coordinates of the actuator head, denoted (x, y) as in Figure A.1. Using orientation data from the IMU, and the distance d, the coordinates (x, y) can simply be obtained with

$$x = \hat{x} - d \sin \varphi, \tag{A.1a}$$

$$y = \hat{y} - d \cos \varphi, \tag{A.1b}$$

with $d = 0.015\,\mathrm{m}$. This equation is implemented on the microcontroller and is calculated each timestep.

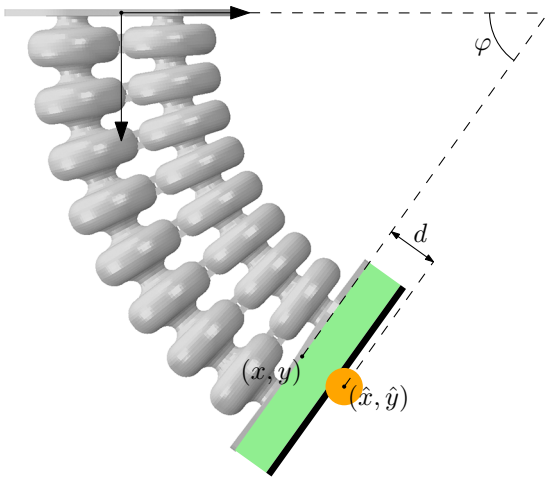


Figure A.1: Difference between the marker coordinates (denoted by the ˆ) and actuator head coordinates.

Appendix B

Lens distortion

Lens distortion can be divided into radial distortion, decentering distortion and thin prism distortion. Of these, radial distortion is the most significant, and a small component of non radial distortion is also present [33]. Lens distortion is mostly an issue with wide-angle lenses [34], which the Pixy2 does not have. Nevertheless, we still investigate the influence of lens distortion on the position accuracy.

Now define μ and ν as the ideal, non-observable distortion-free pixel coordinates, and $\breve{\mu}$ and $\breve{\nu}$ the corresponding measured, distorted, pixel coordinates. Radial distortion is often modelled with the symmetric model [32]

$$\breve{\mu} = \mu + (\mu - \mu_0)\left\{k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right\}, \tag{B.1a}$$

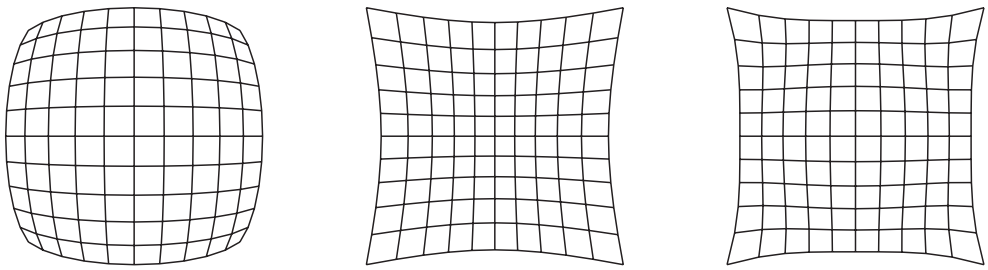
$$\breve{\nu} = \nu + (\nu - \nu_0)\left\{k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right\}, \tag{B.1b}$$

with the radial distortion coefficients k_1 and k_2. The distortion center (μ_0, ν_0) is equal to the principal point. If k_1 and k_2 are positive we have barrel distortion, if k_1 and k_2 are negative we have pincushion distortion, and if k_1 and k_2 have a different sign we have mustache distortion. Figure B.1 shows examples of these three kinds of radial distortion. The distortion parameters can be estimated by extending (2.10) to [32]

$$\begin{aligned} &\underset{\boldsymbol{K},\mathbf{R}_i,\boldsymbol{t}_i,k_1,k_2}{\arg\min} \sum_{i=1}^{b}\sum_{j=1}^{a}\left\|\boldsymbol{P}_{\mathrm{i},ij} - \breve{\boldsymbol{P}}_{\mathrm{i},ij}\left(\boldsymbol{K},\mathbf{R}_i,\boldsymbol{t}_i,\boldsymbol{P}_{\mathrm{w},j},k_1,k_2\right)\right\|^2, \\ &\text{s.t. } \boldsymbol{K}\in\mathbb{R}^{3\times3},\ \mathbf{R}_i\in\mathbb{R}^3,\ \boldsymbol{t}_i\in\mathbb{R}^3,\ k_1\in\mathbb{R},\ k_2\in\mathbb{R}, \end{aligned} \tag{B.2}$$

with $\breve{\boldsymbol{P}}_{\mathrm{i},ij}\left(\boldsymbol{K},\mathbf{R}_i,\boldsymbol{t}_i,\boldsymbol{P}_{\mathrm{w},j},k_1,k_2\right)$ the mapping of point $\boldsymbol{P}_{\mathrm{w},j}$ according to (2.9) and (B.1). Using MATLAB's Camera calibration application the distortion parameters for the Pixy2 are estimated to be $k_1 = -0.2201$ and $k_2 = 0.5494$, which indicates we have mustache distortion. Figure B.2 shows the difference between a raw Pixy2 image and an undistorted one (note the black borders around the edge). Figure B.2c shows a colour map of the Euclidean norm between the original distorted pixels of Figure B.2a and the undistorted pixels of Figure B.2b, i.e. the norm $\|(\mu-\breve{\mu},\ \nu-\breve{\nu})\|_2$. The red rectangles indicate the border around the checkerboard, which area is appropriately equal to the range of measurement that the marker will be in.

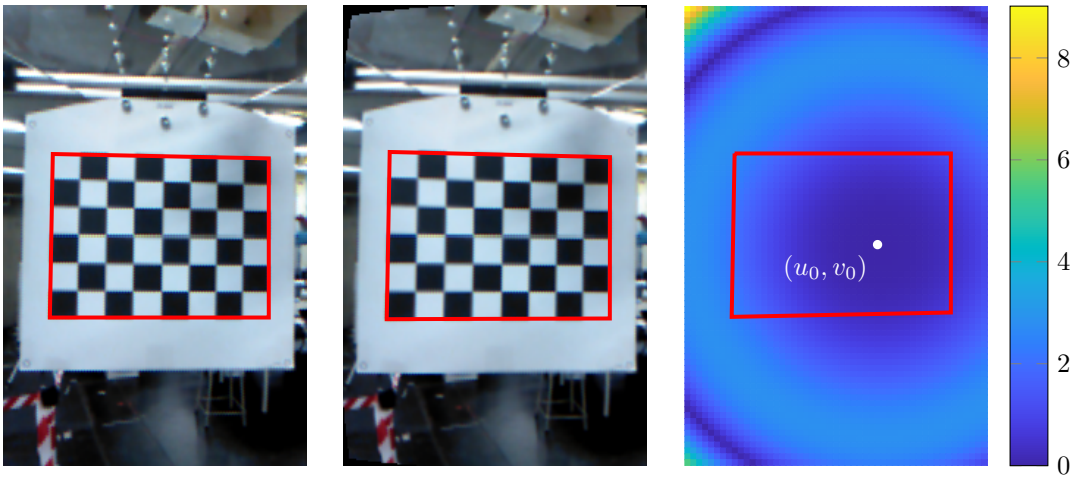
It is not possible to find a closed-form expression to undistort the pixel coordinates. This would mean that it is necessary to do a numeric nonlinear least-squares optimization procedure each time



(a) Barrel distortion **(b)** Pincushion distortion **(c)** Mustache distortion

Figure B.1: Types of radial distortion

step to obtain the undistorted pixel coordinates. Judging from Figure B.2, the Pixy2 distortion is barely noticeable in the area of interest. Therefore, to ensure the calculation time of the mapping remains fast the lens distortion is neglected.



(a) Original **(b)** Undistorted **(c)** Euclidean norm

Figure B.2: Image of the checkerboard made with the Pixy2, and a colour map of the Euclidean distance between the distorted and undistorted pixels.

Appendix C

Identity of homogeneous derivatives

A proof for the identity (3.23) is given here. Let us define the function

$$f(\mathbf{H}) = \mathbf{H}(q). \tag{C.1}$$

The time derivative of (C.1) is then

$$\frac{\partial f}{\partial t} = \frac{\partial \mathbf{H}}{\partial t} = \frac{\partial \mathbf{H}}{\partial q}\frac{\partial q}{\partial t} = \frac{\partial \mathbf{H}}{\partial q}\dot{q}. \tag{C.2}$$

the derivative of (C.2) with respect to $\dot{q}$ can now be written as

$$\frac{\partial}{\partial \dot{q}}\left(\frac{\partial f}{\partial t}\right) = \frac{\partial \mathbf{H}}{\partial q} = \frac{\partial \dot{\mathbf{H}}}{\partial \dot{q}}, \tag{C.3}$$

and we have obtained the identity (3.23).

Appendix D

Discrete PID controller

Consider the continuous PID controller

$$\boldsymbol{u}(t) = K_p \boldsymbol{e}(t) + K_i \int_0^t \boldsymbol{e}(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}. \tag{D.1}$$

The derivative can be approximated in discrete time with the backwards difference approach as

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} \approx u_{d,k} = \frac{e_k - e_{k-1}}{\Delta t} \tag{D.2}$$

The integral can be approximated with the trapezoidal rule as

$$\int_0^{k\Delta t} \boldsymbol{e}(t)\mathrm{d}t \approx u_{i,k} = u_{i,k-1} + \frac{e_k + e_{k-1}}{2}\Delta t. \tag{D.3}$$

Using the Z-transform, the discrete derivative and integral can be written as

$$U_d(z) = \frac{E(z) - z^{-1}E(z)}{\Delta t}, \tag{D.4}$$

$$U_i(z) = z^{-1}U_i(z) + \frac{E(z) + z^{-1}E(z)}{2}\Delta t. \tag{D.5}$$

respectively. With these Z-transforms we can define the discrete transfer function of the PID controller as

$$\frac{U(z)}{E(z)} = K_p + \frac{K_i \Delta t}{2}\frac{z+1}{z-1} + \frac{K_d}{\Delta t}\frac{z-1}{z}. \tag{D.6}$$

Rewriting gives rise to the equation

$$\frac{U(z)}{E(z)} = \frac{\left(K_p + \frac{1}{2}K_i\Delta t + \frac{K_d}{\Delta t}\right) + \left(-K_p + \frac{1}{2}K_i\Delta t - \frac{2K_d}{\Delta t}\right)z^{-1} + \frac{K_d}{\Delta t}z^{-2}}{1 - z^{-1}}. \tag{D.7}$$

Multiplying both sides with $(1 - z^{-1})E(z)$ gives

$$U(z) = z^{-1}U(z) + \left(K_p + \frac{1}{2}K_i\Delta t + \frac{K_d}{\Delta t}\right)E(z) + \left(-K_p + \frac{1}{2}K_i\Delta t - \frac{2K_d}{\Delta t}\right)z^{-1}E(z) + \frac{K_d}{\Delta t}z^{-2}E(z), \quad \text{(D.8)}$$

and we end up with the final expression for the discrete PID controller

$$u_k = u_{k-1} + \left(K_p + \frac{1}{2}K_i\Delta t + \frac{K_d}{\Delta t}\right)e_k + \left(-K_p + \frac{1}{2}K_i\Delta t - \frac{2K_d}{\Delta t}\right)e_{k-1} + \frac{K_d}{\Delta t}e_{k-2}. \quad \text{(D.9)}$$