



OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG



MASTERARBEIT

Tamara Szecsey

Intelligentes Schuhwerk für den humanoiden NAO Roboter basierend auf Magnetostiction zur Verbesserung der Bodenhaftung

Fakultät:	Elektro- und Informationstechnik
Studiengang:	Master Electrical- and Microsystem Engineering
Abgabefrist:	31. März 2021
Betreuung:	Prof. Dr. Gareth Monkman
Zweitbegutachtung:	Dr. Dirk Sindersberger

Erklärung

1. Mir ist bekannt, dass dieses Exemplar der Masterarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.
2. Ich erkläre hiermit, dass ich diese Masterarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum und Unterschrift

Vorgelegt durch:	Tamara Szecsey
Matrikelnummer:	3140789
Studiengang:	Master Electrical- and Microsystem Engineering
Bearbeitungszeitraum:	1. Juni 2020 – 31. März 2021
Betreuung:	Prof. Dr. Gareth Monkman
Zweitbegutachtung:	Dr. Dirk Sindensberger

Inhaltsverzeichnis

1	Einleitung	1
2	Theoretischer Hintergrund	2
2.1	Aufbau des NAO	2
2.2	Verwendete Software	6
2.3	Magneto-aktive Polymere	8
2.4	Magnetostiction	9
3	Versuchsaufbau	10
3.1	Schuhkonstruktion	10
3.2	Herstellung des MAP	11
3.3	Laufstegkonstruktion	11
4	Auswertung und Interpretation	13
5	Anhang	14

1 Einleitung

Kapitel 2.3 erklärt die Definition und Eigenschaften von Magneto-aktiven Polymeren (MAP), welche als Sohle für den Nao Roboter eingesetzt wurden. Diese Erklärungen basieren auf einem Buch von Pelteret und Steinmann [1], welches ich für tiefergehende Lektüren empfehle.

2 Theoretischer Hintergrund

2.1 Aufbau des NAO

NAO ist ein 574 mm großer, humanoider Roboter (siehe Abb. 1) ursprünglich entwickelt von dem französischen Unternehmen Aldebaran Robotics, welche 2015 von Softbank Group aufgekauft [3] und in Softbank Robotics umbenannt wurde. Während NAO's große Schwester Pepper mit ihren 1,20 m mit einem Tablet und Rollen statt Beinen ausgestattet ist [4], gibt es NAO in verschiedenen Ausführungen, unter anderem nur ab der Hüfte aufwärts oder mit Beinen. Es handelt sich hier um Roboter, die unter anderem Kinder und Jugendlichen die Robotik näher bringen sollen und der Vorführung von Mensch-Roboter Interaktionen dienen. NAO bietet außerdem die Gelegenheit zweibeinige Robotersysteme zu studieren und ist bereits in psychologischen Studien verwendet worden (cite).



Abb. 1. NAO V6 [2]

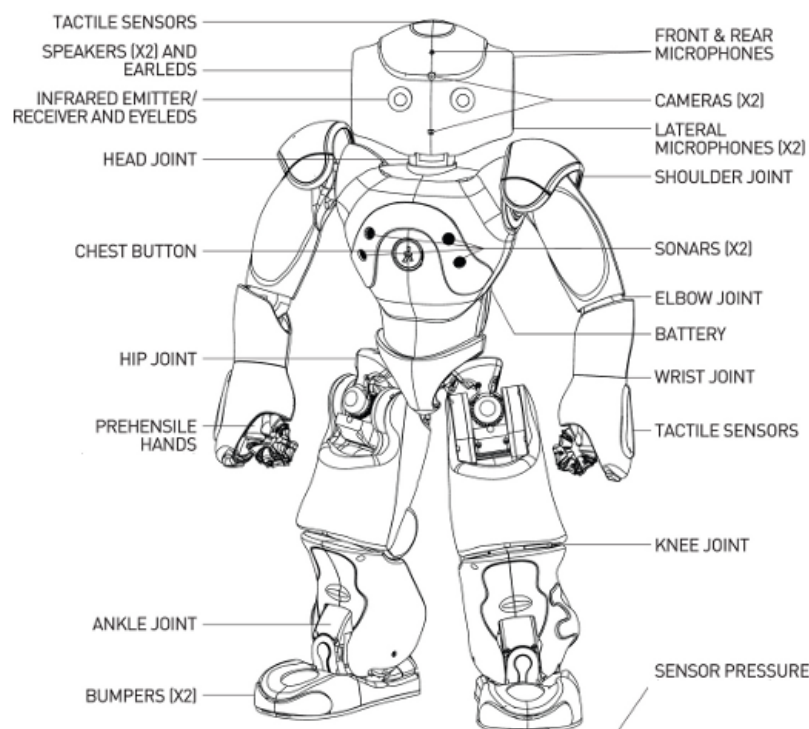


Abb. 2. Sensorenüberblick des NAO-H25 Version 6 [5, in /H25]

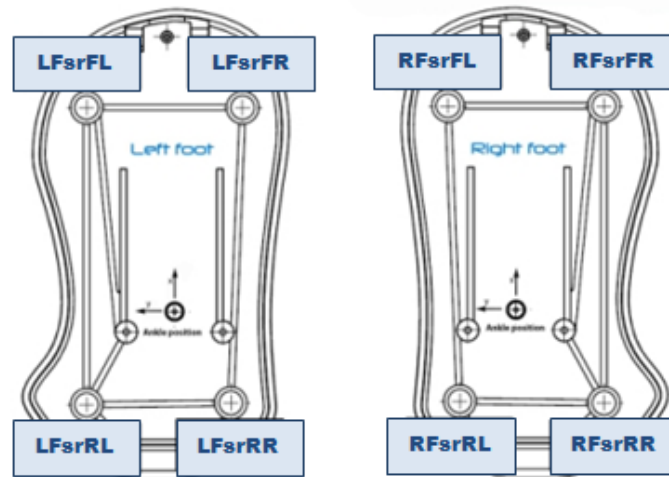


Abb. 3. Drucksensoren in den Füßen von NAO [2, in /Technical overview/FSRs]

Das hier verwendete Modell ist NAO-H25 Version 6, dessen Sensoren in Abb. 2 zu sehen sind. Im Unterschied zu anderen Ausführungen besitzt NAO-H25 Drucksensoren an Händen und Fußsohlen. Er gehört zu den kommerziellen Robotern deren Gelenke positionsbasierenden sind [6], hat 25 Freiheitsgrade und wiegt 5,4 kg. Über die an der Brust angebrachten Sonar Sensoren, die Kameras oberhalb und unterhalb der Augen LEDs, die Vorder- und Rückseitigen Mikrofone, den Stoßfängern an den Füßen sowie den Kontaktsensoren an Händen und Kopf kann der NAO mit seiner Umwelt vielseitig interagieren. Jedes Gelenk ist mit Sensoren für die Winkelmessung, den Stromverbrauch und Temperaturmessung ausgerüstet und in seiner Brust befindet sich außerdem ein Gyroskop. Auf die in dieser Arbeit verwendeten Messausgaben wird im Folgenden genauer eingegangen.

Druckempfindlicher Widerstand

An den Fußsohlen des NAO befinden sich pro Fuß vier sogenannte *Force Sensitive Resistors* (FSR) zu sehen in Abb. 3. Diese ändern ihren Widerstand sobald Druck ausgeübt wird und messen im Bereich von 0 bis 25 N.

Jeder dieser Sensoren kann einzeln ausgelesen werden, so zum Beispiel LFsrFL unter `Device/SubDeviceList/LFoot/FSR/FrontLeft/Sensor/Value`. Des weiteren können der berechnete zweidimensionale Massenschwerpunkt und das Gesamtgewicht ausgegeben werden. Diese Werte sind allerdings unzuverlässig, sobald wenig oder kein Gewicht auf den Sensoren lastet. In [7] haben Shayan et al. die eingebauten FSR mit barometrischen Drucksensoren verglichen und sind zu dem Schluss gekommen, dass die Sensoren, welche in NAO verbaut sind, unzuverlässig arbeiten. Deshalb werden hier die Ausgaben kritisch betrachtet, sowie die Balance des Roboters zusätzlich mit dem Gyroskop erfasst.

Eine weitere Einschränkung der Messungen ist die Kraftverteilung auf die FSRs. Denn in dem ursprünglichen unteren Teil des Schuhs liegt der obere Teil ausschließlich auf den

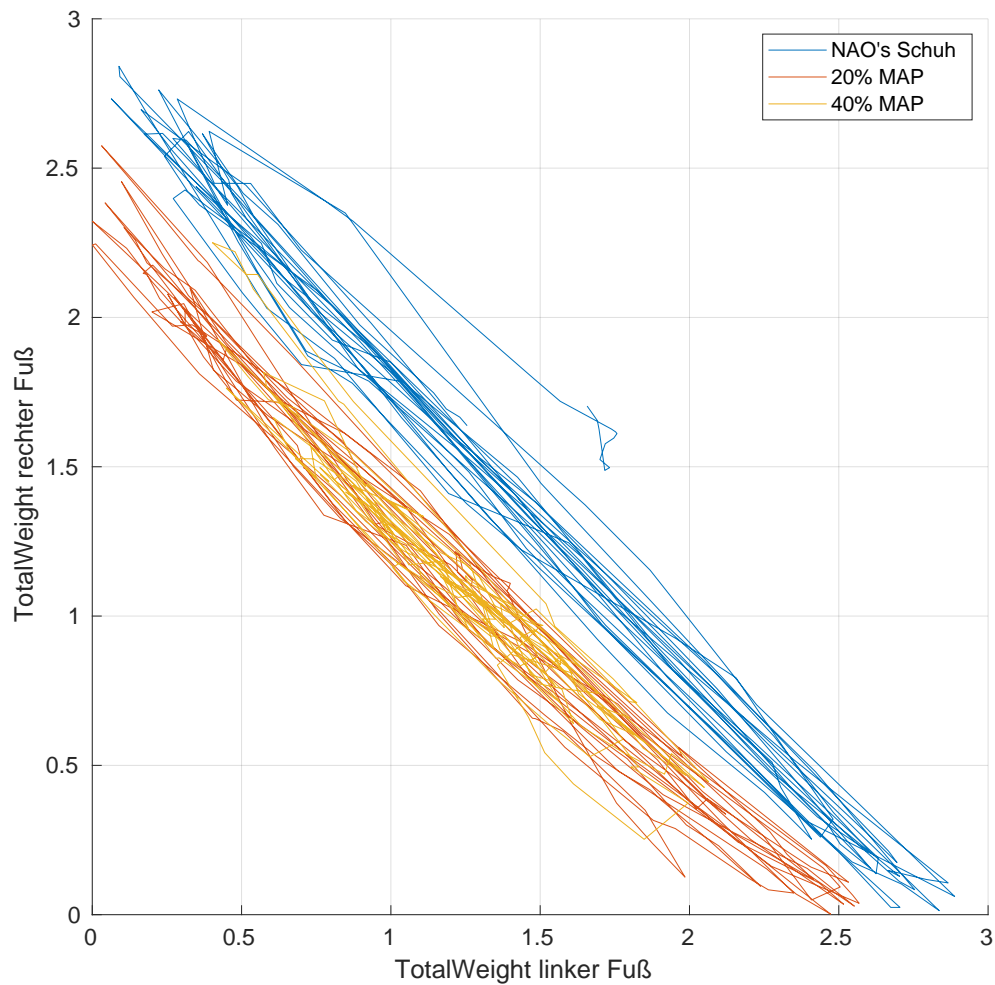


Abb. 4. Berechnetes Gesamtgewicht der Messungen für die normalen Sohlen von NAO in blau und dem Silikon versetzt mit Eisenpartikeln in 20% und 40% Anteilen. Das Gewicht wird in kg ausgegeben.

Sensoren und auf den Verbindungszylindern für die Schrauben auf. Dies ist für die Einlegesohlen aus Silikon bzw. MAP nicht der Fall, zu sehen in den Kapiteln 3.1 und folgende. Dies führt dazu, dass das Gewicht nicht mehr akkurat aufgenommen wird, wie in Abb. 4 zu sehen ist. Hier ist zu sehen, dass sich die Messung des Gewichtes etwa um 0,5 kg von den NAO Schuhen abweicht.

Aktoren und Sensoren der Beinen

Neben dem für diese Arbeit uninteressanten Bumper Sensoren und den bereits beschriebenen Drucksensoren hat NAO eine Ausgabe für jeden eingebauten Aktor mit den Werten:

- .../Position/Actuator/Value (Pos/Act)
- .../Position/Sensor/Value (Pos/Sens)

- .../ElectricCurrent/Sensor/Value (Current)
- .../Temperature/Sensor/Value
- .../Hardness/Actuator/Value
- .../Temperature/Sensor/Status

Hierbei unterscheiden sich die Pfade bei „...“ je nach Aktor. Die Bezeichnungen in Klammern dahinter dienen der abkürzenden Benennung für spätere Kapitel.

Die Temperatursensoren sowie die Starrheit der Motoren liefern keine verwertbaren Messausgaben, da sich beide während einer Messung nicht oder kaum ändern. Pos/Act und Pos/Sens geben ähnliche Werte in rad aus, da ersteres die Ausgabe des Programms vorgibt und zweiteres den tatsächlich gemessenen Wert an dem Gelenk ausgibt. Der Current Wert wird in Ampère gemessen und gibt an, wie viel Strom für das Erreichen der entsprechenden Aktorposition und Starrheit aufgewendet werden muss. Dies bedeutet, dass dieser Wert unter anderem eine Aussage über den Zustand des Aktors geben kann.

Bei den Probegängen des NAO stellte sich heraus, dass er in einer Kurve nach links läuft bei einem Befehl für geradeaus. Softbank Robotics betonte, dass es nicht möglich ist, dass NAO komplett gerade läuft. Grund dafür ist zum einen, dass die Motoren nicht immer komplett identisch hergestellt sind. Zum anderen ist der Gehbefehl, welcher hier ausgeführt wird, ohne Rückkopplungsschleife für Einwirkungen der Umgebung. Näheres zur Software wird in Kapitel 2.2 beschrieben.

Gyroskop

Der NAO Roboter verfügt über eine Inertialeinheit, welche sich zusammensetzt aus den 3 Achsen des Gyrometers, im einer Achsengeschwindigkeit von bis zu $500^\circ/\text{s}$, sowie den 3 Achsen des Beschleunigungssensors, mit einer Beschleunigung bis zu 2 g , zusammensetzt [2, /Technical overview/Inertial unit]. Die Z-Achse des Gyroskops ist allerdings noch nicht verfügbar. Außerdem wird durch diese beiden Sensoren der Winkel des Torsos bestimmt.

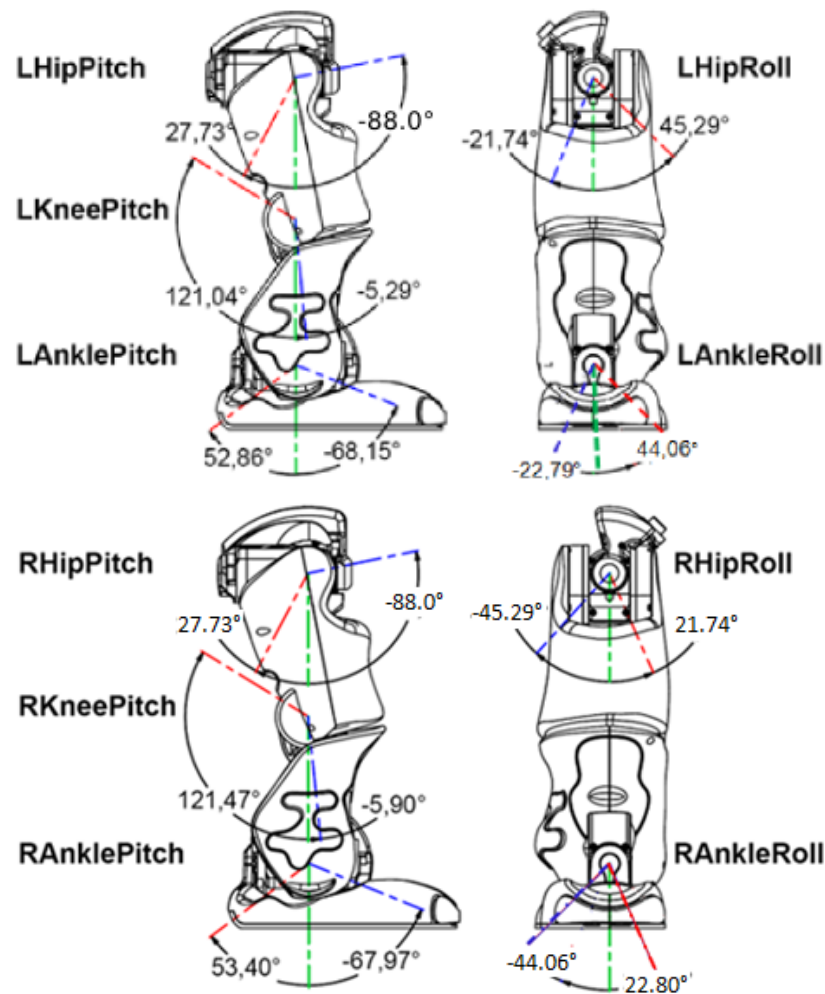


Abb. 5. Das obere Bild zeigt Vorder- und Seitenansicht der Positionen und möglichen Winkel des linken Beins. Die untere Abbildung veranschaulicht dieselben Parameter für das rechte Bein. [2, in /kinematics-data/joints]

2.2 Verwendete Software

Die Rahmenumgebung NAOqi

NAOqi wird die Hauptsoftware genannt, welche auf diesem Roboter sowie auf Pepper läuft, das Betriebssystem NAOqi OS basiert auf Linux. Der NAOqi Rahmen ist eine plattformübergreifende und sprachenübergreifende Umgebung, mithilfe derer Anwendungen für den NAO erstellt werden können. Sie kann über Windows, MacOS und Linux verwendet werden und es werden die Sprachen Python und C++ unterstützt, wobei ersteres direkt auf dem NAO kompiliert werden kann, während C++ komplizierter benutzt wird, aber mehr Eingriffe erlaubt. [8, /Former NAOqi Framework/Key concepts]

Außerdem gibt es eine Desktop Anwendung namens Choregraphe, in der Dialoge und Verhaltensmuster erstellt werden können, ohne Code schreiben zu müssen. Außerdem ist

hierüber der Akkustand und aktuelle Position in Form eines virtuellen Bildes des tatsächlichen NAO einsehbar, sowie der autonome Zustand ein- und ausschaltbar. [8, /Choregraphie Suite/What is Choregraphie]

Für die in dieser Masterarbeit benötigten Anwendungen war Python am besten geeignet. Erforderlich waren eine sich wiederholende Schrittabfolge sowie die Aufzeichnung diverser bereits in NAO verbauten Sensoren. Text verarbeitende Funktionen sind in dieser Sprache leicht zu erhalten und anzupassen. Es wurde sich für den `moveTo()` Befehl als Fortbewegung entschieden, denn dieser ist durch das Objekt `post` einem sogenannten *non-blocking call*. Dies ermöglicht das Aufrufen von weiteren Befehlen parallel zur Bewegung. Der Nachteil ist, dass der Gang dadurch unbeaufsichtigt vonstatten geht, d.h. NAO kann seine Schritte nicht seiner Umgebung anpassen. Die aufgezeichneten Daten werden in eine csv Datei übertragen und mit Matlab ausgewertet. Im Anhang ist der gesamte Programmcode 1 abgebildet, welcher auf dem NAO ausgeführt wird.

Konstruktionssoftware

Das CAD-Programm Inventor von Autodesk ist für 3D Konstruktionen ausgelegt und bietet einige nützliche Simulationserweiterungen, welche unter anderem einen Shape Generator enthält [9]. Dieser kann Flächen minimieren während die Stabilität erhalten bleibt, sodass ein minimaler Anteil an Material verwendet werden wird [10].

Für die Herstellung der Prototypen und Gussformen wurde das FDM (fused deposition modeling) 3D Druckverfahren mit PLA oder PETG Filamenten verwendet und für die Vorbereitung auf den Druck der Slicer Cura von Ultimaker [11].

Datenauswertung mit Matlab

Für die Auswertung wurden neben der gewöhnlichen `plot` Funktion von Matlab [12] auch die *Statistics and Machine Learning* Toolbox mit deskriptiver Statistik und Visualisierung verwendet [13]. Eine Messreihe ergibt sich aus 20-40 Messungswiederholungen, in denen NAO etwa einen Meter zurücklegt. Jede Auswertung enthält das arithmetische Mittel eines Sensors von allen Messungen der Messreihe. Diese Mittel wurden mit dem Befehl `mean()` berechnet und mit Werten aus weiteren Messreihen, mit anderem Schuh oder anderen Untergrundvoraussetzungen aber gleicher Schrittlänge und Sensorenmessung verglichen. Dieser Vergleich wurde mit den Funktionen `hist` und `scatterhist` der genannten Toolbox angestellt. Ersteres erstellt ein Histogramm und ist damit optimal für eine Häufigkeitsverteilung. Zweiteres erstellt zwei Histogramme von jeweils zwei Vektoren und einen aus diesen Vektoren resultierenden Scatterplot. Dies ermöglicht eine umfangreiche Verteilungsanalyse für Sensorausgaben, die sich aus zwei Achsen zusammensetzen, wie z.B. der 2 dimensional Gyroskop Ausgabe.

2.3 Magneto-aktive Polymere

Begriffserklärung und Eigenschaften

Der Begriff magneto-aktive Polymere (MAP) schließt eine Gruppe von intelligenten, auf Felder ansprechende Materialien ein, welche typischerweise Kombinationen aus einer weichen, polymetrischen Grundlage und magnetisch aktiven Partikeln sind. Diese Partikel werden während dem Vernetzungsprozess des Polymers in dieses eingebettet.

Die wesentlichen Verhaltensweisen, die MAP in der heutigen Zeit attraktiv für seine Verwendung gemacht hat, wurde bereits in den 80er Jahren von Rigbi und Jilken [14] sowie Rigbi und Mark [15] beschrieben. Ein Jahrzehnt später wurde eine genauere Analyse zum ersten Mal von Ginder und Jolly et al. [16] veröffentlicht. Diese kombinierten, aus mehreren Komponenten bestehenden Materialien stechen durch zwei Schlüsseigenschaften heraus. Zum einen ist es das magnetostriktive Verhalten, bei dem es sich um das Phänomen der Verformung eines Materials handelt, welches durch ein Magnetfeld hervorgerufen wird [17]. Zum anderen sind es die leicht veränderbaren Materialeigenschaften wie Elastizität und Dämpfungsfaktor, welche hauptsächlich mit der Mikrostruktur des Grundlagematerials zusammenhängt. [18] [19]

Außerdem ist entscheidend, wie die magnetischen Partikel in das Polymer eingebettet werden. Je nachdem ob während des Vernetzungsprozesses ein Magnetfeld wirkt, können sich die Partikel kettenförmig ausrichten und dadurch dem MAP eine anisotropisches Verhalten zuführen. Isotropisches MAP hingegen enthält keine gerichteten Partikel. Diese verschiedenen Ausrichtungsarten können sowohl die Steifigkeit verändern als auch bestimmen, ob das MAP in einem Magnetfeld ausgedehnt oder zusammengedrückt wird.

Anwendungsbereiche

Weiche, mit einem Feld manipulierbare, Polymere haben diverse Anwendungsbereiche in akademischen und industriellen Bereichen. Angefangen von anpassungsfähiger Vibrationsabsorption in der Luftfahrt und Automobilindustrie durch das Einsetzen durch Scherung [20] [21], Windung [22] und Kompression bzw. Elongation [23] und Vibrationsisolatoren [24] sowie Sensoren [24] [17], Ventile und Aktoren [25] [26] und anpassungsfähige Sandwichartige Strukturen [27] [28] [29] bis hin zur Anwendung in der Bionik wie zum Beispiel durch Mikro- und Nanoroboter und Schwimmroboter [30] [31] [32] [33], Schlauchradpumpen [34] und Erschütterungsdämpfer [35].

Des Weiteren wird die Verhärtung bei Anlegen eines Magnetfeldes für Greifer genutzt (cite suchen), ebenso wird der 3D Druck von magneto-aktiven Polymeren (cite von uns) wird bereits erforscht.

Herstellung der Proben

Der Herstellungsprozess allein dauert nicht mehr als eine halbe Stunde. Zunächst muss das Verhältnis für den Anteil des CIPs bestimmt werden. Die Masse ergibt sich aus

$$m_{\text{CIP}} = \frac{\text{Ratio}_{\text{CIP}}[\%]}{100} \cdot m_{\text{ges}}, \quad (2.1)$$

Die beiden additiven Komponenten A und B, welche auch als Basis und Katalysator bezeichnet werden, sind im Verhältnis 1:1 zu mischen. Das Volumen in ml ergibt sich aus:

$$V_A = \frac{m_A}{\rho_A}, \quad V_B = \frac{m_B}{\rho_B} \quad (2.2)$$

mit $\rho_A = 1.071$ sowie $\rho_B = 1.046$ sowie

$$m_A = \left(1 - \frac{\text{Ratio}_{\text{CIP}}[\%]}{100}\right) \times \frac{m_{\text{ges}}}{\alpha + \beta} \times \alpha, \quad m_B = \left(1 - \frac{\text{Ratio}_{\text{CIP}}[\%]}{100}\right) \times \frac{m_{\text{ges}}}{\alpha + \beta} \times \beta, \quad (2.3)$$

Die Platzhalter α und β stehen für das Mischverhältnis der jeweiligen Komponenten und sind hier Beide gleich 1, sodass die Formeln vereinfacht werden können in:

$$m_A = \left(1 - \frac{\text{Ratio}_{\text{CIP}}[\%]}{100}\right) \times \frac{m_{\text{ges}}}{2}, \quad m_B = \left(1 - \frac{\text{Ratio}_{\text{CIP}}[\%]}{100}\right) \times \frac{m_{\text{ges}}}{2}. \quad (2.4)$$

Mit der Laborwaage ABT 120-5DM von Kern wird m_{CIP} in einem Becher abgemessen. Mit den Spritzen lassen sich V_A und V_B auf die ersten Nachkommastelle genau beifügen. Es handelt sich um SF13 2k-Silikon vom Hersteller Silikon Fabrik. Der Becherinhalt wird dann mit einem kleinen Mixstab gemischt, um eine gleichmäßige Verteilung der beiden Komponenten zu erreichen und somit eine optimale Vernetzung zu gewährleisten. Anschließend wird die Probe in einen Exikator gestellt welcher mit einer Vakuumpumpe entleert wird, sodass das Silikon entgast ist. Schließlich kann das MAP in die Gussform gegossen werden, nach spätestens einem Tag ist die Sohle dann komplett vernetzt. Das Silikon selbst hat eine zu große Haftung, v.a. durch die Fläche des Schuhs. Deshalb wird es vor der Messung mit Speisestärke eingedeckt, was eine Bodenhaftung ähnlich der Plastiksuhsohle des Originalschuhs von NAO zur Folge hat.

2.4 Magnetostiction

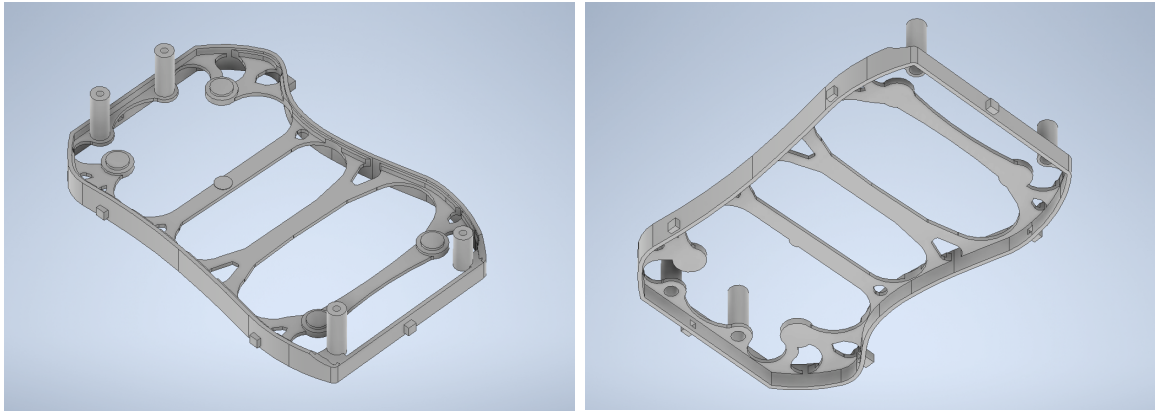


Abb. 6. Linker Schuh in Autodesk Inventors, Links von Oben, Rechts von Unten.

3 Versuchsaufbau

In diesem Kapitel wird auf alle selbstkonstruierten Komponenten eingegangen, welche für die Messungen benötigt wurden. Zum einen gibt es eine Schuhkonstruktion, die es erlaubt, die in einer Gussform angefertigten MAP Sohlen einzuhängen, zum anderen wurde ein Laufsteg mit Rampenfunktion entworfen, welche es erlaubt, Magnete unter die Lauffläche zu montieren.

3.1 Schuhkonstruktion

Die Hülle eines Fußes von NAO besteht aus einem zweiteiligen Oberteil, welches das Fußgelenk abschließt und nur mit dem unteren Teil zusammen mit 4 Schrauben fest sitzt. Die in Kapitel 2.1 beschriebenen vier Drucksensoren liegen dabei in dem unteren Teil auf Erhöhungen auf. Um die Auswirkungen anderer Sohlen für NAO messen zu können, muss der untere Teil des Fußes ausgetauscht werden. Dieser „Schuh“ welcher die ursprüngliche Fußsohle ersetzt, enthält wiederum einen Steckplatz für die MAP Sohlen in verschiedenen Stärkegraden.

In Abb. 6 sind links die vier flachen Zylinder zu sehen, auf denen die Drucksensoren aufliegen. Durch die vier Zylinder an den Seiten werden Schrauben gelegt, welche an das obere Teil des Fußes von NAO geschraubt werden. Die Außenform umschließt den oberen Teil während eine zweite Erhöhung, welche an den Innenseiten verläuft, auf dem Rand des oberen Teils aufsitzt, sodass die Passform fest ineinander greift. Da das gesamte Gewicht des NAO auf den 4 Drucksensoren lastet, kann Druckmaterial für die restliche Gesamtfläche bis auf stabilitätserhaltende Streben eingespart werden. Diese Form wurde mit dem Shape Generator von Autodesk Inventor (cite) generiert, sodass sie längs bis zu einem 45° Winkel ohne zu brechen gebogen werden kann. Die instabilsten Stellen sind die Zylinder der Schraubvorrichtung, welche mit Sekundenkleber verstärkt werden können. Die Instabilität ist auf den schichtweise Druckvorgang mit einem

Die Unterseite, zu sehen in Abb. 6 rechts, ist ein Hohlraum für die MAP Sohle zusammen mit den viereckigen Steckeinlässen für die Halterung. Die Seiten des Schuhs sind so hoch,

dass das MAP, welches in der Gussform gegossen wurde, etwa 1 mm herausragt. Andernfalls würde NAO auf der Schuhkante laufen und nicht auf dem MAP.

3.2 Herstellung des MAP

Das verwendete Polymer als Grundlage für das MAP ist ein additiv vernetzendes Silikon von (Firma und cite einbinden). Während dem Mischverfahren ist es flüssig und muss deshalb in eine Form gegeben werden. Silikon selbst lässt sich nur sehr schlecht durch etwaige Klebstoffe nach der Vernetzung verkleben. Deshalb wird hier wie in Abb. 7 zu sehen ist, eine 2 mm dicke Stangenkonstruktion eingehängt, welche bis auf die 6 Enden mit MAP umschlossen wird. Diese aus PLA gedruckte Konstruktion ist flexibel und kann deshalb durch Verbiegen in die Verankerungen gedrückt werden. Nach der vollständigen Vernetzung kann die Sohle aus der Form entnommen und in den Schuh aus dem vorherigen Kapitel eingesetzt werden.

Die vier Zylinder dienen als Platzhalter um die sechs Ecken in der Halterung des Schuhs für einen besseren Halt festzukleben und dann durch die Löcher des MAPs die Schrauben lockern zu können.

3.3 Laufstegkonstruktion

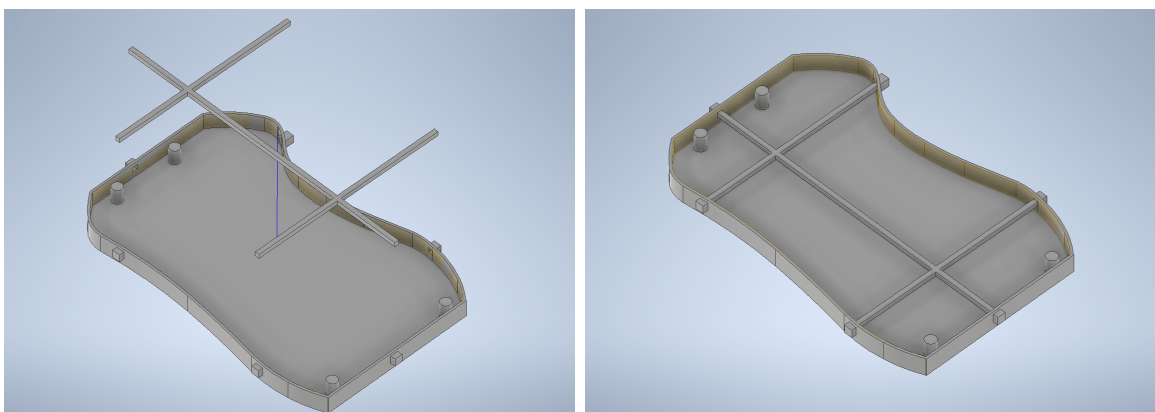


Abb. 7. Gussform der MAP Sohlen. Links ist die Innenhalterung herausgenommen, rechts ist sie eingespannt in den sechs Eckhalterungen.

Der NAO Roboter ist für den Einsatz auf geraden Bodenflächen im Innenbereich ausgelegt wobei er bei einem Bewegungsablauf ohne Anpassung an die Umwelt wie mit dem Befehl `moveTo()` durch Rutschen nicht immer die gleiche Strecke zurücklegt. Um wiederholbare Messreihen garantieren zu können ist eine Teststrecke von Nöten. Des Weiteren sind verschiedene, flache Untergründe für eine Sohlenentwicklung interessant. Außerdem kann auf das MAP nur Einfluss genommen werden, wenn ein magnetisches Feld angelegt wird. Deshalb wurde ein Laufsteg mit einem Hohlraum angefertigt, um unter der Fläche, auf der NAO läuft, Magneten angebracht werden.

Der Laufsteg besteht aus einer $120 \times 66,4$ cm großen Pressholzplatte, die auf der Oberseite mit einem Aluminiumkonstrukt erweitert ist, welches die Einschubplatten von beiden Längsseiten und nach oben hin abschließt, Abb. 8 links. Auf den kurzen Seiten verriegeln jeweils Zwei drehbare Keile den Einschub, sodass die Platten eingeschlossen werden, Abb. 9.



Abb. 9. Seitenansicht der Rampe mit einer Breite von $66,4$ cm.

Auf der Unterseite sind an den Längsseiten zwei mit T-Nut versehene Aluminiumstangen angebracht, sowie eine zweiteilige Stangenkonstruktion, die eine Winkelverstellung mit Raste erlaubt, zu sehen in Abb. 8 rechts. Die einstellbaren Winkel betragen ca. 5° bis 17° , oder es wird für 0° vollständig eingeklappt.

Man hat bereits NAO schräge Flächen gehen lassen wie in (cite). Dies erfordert einen komplett anderen Gang und hätte den Rahmen dieser Arbeit gesprengt. Die Neodymmagnete, die verwendet wurden, haben eine Haftkraft von ca. 16 kg, eine Maße von $40 \times 40 \times 4$ mm [36] und wurden an die Unterseite der Rampe geschraubt. Die ersten Versuche ergaben schließlich, dass das MAP nur bei einem Abstand unter den Einlageplatten reagierte. Deshalb wurde in den hiesigen Messungen nur ohne Platten gemessen.

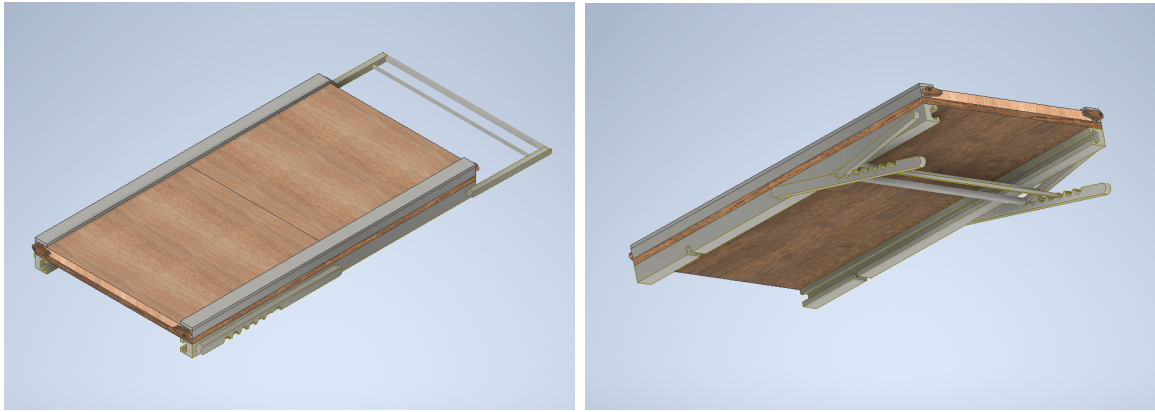


Abb. 8. Laufstegrampenkonstruktion mit zwei austauschbaren Platten und einer Winkelverstellung mit Raste. Links: Sicht von schräg oben mit eingeklappter Winkelverstellung. Rechts: Sicht von schräg unten mit niedrigster Winkeleinstellung.

4 Auswertung und Interpretation

5 Anhang

Code 1. Pythonprogramm für Messaufnahmen

```
1  # ! /usr/bin/env python
2  # -*- encoding: UTF-8 -*-
3
4  """Example: Use getData Method to Use FSR Sensors"""
5
6  import qi
7  import argparse
8  import sys
9  import time
10 import csv
11 import re
12 import shutil
13 from tempfile import mkstemp
14 import os
15
16 def sed(pattern, replace, source, dest=None, count=0):
17     """Reads a source file and writes the destination file.
18
19     In each line, replaces pattern with replace.
20
21     Args:
22     pattern (str): pattern to match (can be re.pattern)
23     replace (str): replacement str
24     source (str): input filename
25     count (int): number of occurrences to replace
26     dest (str): destination filename, if not given,
27         source will be over written.
28     """
29
30     fin = open(source, 'r')
31     num_replaced = count
32
33     if dest:
34         fout = open(dest, 'w')
35     else:
36         fd, name = mkstemp()
37         fout = open(name, 'w')
38
39     for line in fin:
40         out = re.sub(pattern, replace, line)
41         fout.write(out)
```

```

42         if out != line:
43             num_replaced += 1
44             if count and num_replaced > count:
45                 break
46     try:
47         fout.writelines(fin.readlines())
48     except Exception as E:
49         raise E
50
51     fin.close()
52     fout.close()
53
54     if not dest:
55         shutil.move(name, source)
56
57
58 def zeilen_aufteilen(file):
59     sed(',platzhalter,', '\n', file)
60     sed(',platzhalter', '', file)
61
62
63 def recordData(memory_service):
64     """ Get pressure sensor data from ALMemory
65     Returns a matrix of values
66
67     """
68     print "Recording data..."
69     data = list()
70     for range_counter in range(1, 230):
71         #Gyroscope
72         GyrX = memory_service.getData("Device/SubDeviceList
73             /InertialSensor/GyroscopeX/Sensor/Value")
74         GyrY = memory_service.getData("Device/SubDeviceList
75             /InertialSensor/GyroscopeY/Sensor/Value")
76         data.append(GyrX)
77         data.append(GyrY)
78
79         # Adding the summary of the FSR
80         LFsrTw = memory_service.getData("Device/
            SubDeviceList/LFoot/FSR/TotalWeight/Sensor/Value"
            )
81         RFsrTw = memory_service.getData("Device/
            SubDeviceList/RFoot/FSR/TotalWeight/Sensor/Value"
            )

```

```

81     LFcopX = memory_service.getData("Device/
        SubDeviceList/LFoot/FSR/CenterOfPressure/X/Sensor
        /Value")
82     LFcopY = memory_service.getData("Device/
        SubDeviceList/LFoot/FSR/CenterOfPressure/Y/Sensor
        /Value")
83     RFcopX = memory_service.getData("Device/
        SubDeviceList/RFoot/FSR/CenterOfPressure/X/Sensor
        /Value")
84     RFcopY = memory_service.getData("Device/
        SubDeviceList/RFoot/FSR/CenterOfPressure/Y/Sensor
        /Value")
85     data.append(LFsrTw)
86     data.append(RFsrTw)
87     data.append(LFcopX)
88     data.append(LFcopY)
89     data.append(RFcopX)
90     data.append(RFcopY)
91
92     # LeftAnkleRoll
93     PosAct = memory_service.getData("Device/
        SubDeviceList/LAnkleRoll/Position/Actuator/Value"
        )
94     PosSens = memory_service.getData("Device/
        SubDeviceList/LAnkleRoll/Position/Sensor/Value")
95     ElectrSens = memory_service.getData("Device/
        SubDeviceList/LAnkleRoll/ElectricCurrent/Sensor/
        Value")
96     data.append(PosAct)
97     data.append(PosSens)
98     data.append(ElectrSens)
99
100    # RightAnkleRoll
101    PosAct = memory_service.getData("Device/
        SubDeviceList/RAnkleRoll/Position/Actuator/Value"
        )
102    PosSens = memory_service.getData("Device/
        SubDeviceList/RAnkleRoll/Position/Sensor/Value")
103    ElectrSens = memory_service.getData("Device/
        SubDeviceList/RAnkleRoll/ElectricCurrent/Sensor/
        Value")
104    data.append(PosAct)
105    data.append(PosSens)
106    data.append(ElectrSens)
107
108    # LeftAnklePitch

```

```

109         PosAct = memory_service.getData("Device/
            SubDeviceList/LAnklePitch/Position/Actuator/Value
            ")
110         PosSens = memory_service.getData("Device/
            SubDeviceList/LAnklePitch/Position/Sensor/Value")
111         ElectrSens = memory_service.getData("Device/
            SubDeviceList/LAnklePitch/ElectricCurrent/Sensor/
            Value")
112         data.append(PosAct)
113         data.append(PosSens)
114         data.append(ElectrSens)
115
116         # RightAnklePitch
117         PosAct = memory_service.getData("Device/
            SubDeviceList/RAnklePitch/Position/Actuator/Value
            ")
118         PosSens = memory_service.getData("Device/
            SubDeviceList/RAnklePitch/Position/Sensor/Value")
119         ElectrSens = memory_service.getData("Device/
            SubDeviceList/RAnklePitch/ElectricCurrent/Sensor/
            Value")
120         data.append(PosAct)
121         data.append(PosSens)
122         data.append(ElectrSens)
123
124         data.append('platzhalter')
125         time.sleep(0.05)
126     return data
127
128
129 def count_files():
130     counter = 1
131     # str.zfill schreibt vor, wie lang die Zahl mit Nullen
    davor sein soll. also zfill(3) ist 3 Zahlen lang.
132     filename = 'measurement' + str(counter).zfill(3) + '.
        csv'
133
134     # Wenn das file nicht existiert, erstelle measurement001
    .csv
135     while os.path.exists(filename):
136         counter = counter + 1
137         filename = 'measurement' + str(counter).zfill(3) +
            '.csv'
138     create_file(filename)
139     return filename
140

```

```

141
142 def create_file(filename):
143     with open(filename, "w") as f:
144         pass
145
146
147 def main(session):
148     """
149     This example uses the getData method to use FSR sensors
150     """
151     # Get the ALProxy ALMemory and ALMotion
152     from naoqi import ALProxy
153     memory_service = session.service("ALMemory")
154     motion = ALProxy("ALMotion", "nao.local", 9559)
155
156     # wake up nao
157     motion.wakeUp()
158
159     motion.moveInit()
160     motion.post.moveTo(0.85, -0.10, -0.25, [{"
161         MaxStepFrequency", 0.0}])
162
163     data = recordData(memory_service)
164     filename = count_files()
165
166     output = os.path.abspath(filename)
167     with open(output, "wb") as file:
168         writer = csv.writer(file, delimiter=',')
169         writer.writerow(data)
170         zeilen_aufteilen(output)
171     print "Results written to", output
172     # go back to crouch position and sleep
173     motion.rest()
174
175 if __name__ == "__main__":
176     parser = argparse.ArgumentParser()
177     parser.add_argument("--ip", type=str, default="
178         127.0.0.1",
179         help="Robot IP address. On robot or Local Naoqi: use
180         '127.0.0.1'.")
181     parser.add_argument("--port", type=int, default=9559,
182         help="Naoqi port number")

```

```

183     session = qi.Session()
184     try:
185         session.connect("tcp://" + args.ip + ":" + str(args
            .port))
186     except RuntimeError:
187         print ("Can't connect to Naoqi at ip \"" + args.ip
            + "\" on port " + str(args.port) + ".\n"
188             "Please check your script arguments. Run with -h
            option for help.")
189     sys.exit(1)
190     main(session)

```

Der Programmcode 1 kann in mehrere Funktionen aufgeteilt betrachtet werden. Die Funktion `sed` ist aus (cite) entnommen und funktioniert wie die gleichnamige Funktion unter der Linux-Bash. Sie wird benötigt, um nach jedem Durchgang der Messschleife in `recordData` eine neue Zeile in die CSV Datei zu schreiben. Dies geschieht durch die Funktion `zeilen_aufteilen`. `recordData` wurde aus den Beispielen der NAO Dokumentation (cite) entnommen und angepasst, sodass am Ende jeder Zeile von `data` ein Platzhalter eingefügt wird und alle gewünschten Sensoren abgegriffen werden. Die Funktion `count_files` sorgt dafür, dass keine vorhandenen Messungen überschrieben werden und jede Messdatei eine fortlaufende Nummerierung erhält.

In der `main` Funktion werden `ALMemory` und `ALMotion` geladen, und der Gang einschließlich des Abgreifens der Sensorwerte ausgeführt. Die Ausgabe der Messwerte während dem Gang ist nur möglich durch den Präfix `post` vor `moveTo`.

Abschließend dient die letzte `if`-Abfrage zur Verbindung mit NAO, allerdings nur, wenn dieses Pythonprogramm selbst auf dem NAO liegt. Die Methode `post` sowie die Aufnahme der Sensoren während dem Lauf der Methode `moveTo` funktionieren nur lokal, deshalb ist es in diesem Fall nicht möglich, das Programm von dem eigenen Rechner aus zu starten. Mit anderen Methoden wäre eine Programmaufrufung über eine WLAN Verbindung durchaus möglich. Um Programme direkt auf dem NAO zu starten, wird eine `ssh`-Verbindung hergestellt und darüber dann `python` ausgeführt.

Literatur

- [1] Jean-Paul Pelteret und Paul Steinmann. *Magneto-Active Polymers - Fabrication, characterisation, modelling and simulation at the micro- and macro-scale*. De Gruyter, 2020. DOI: <https://doi.org/10.1515/9783110418576>.
- [2] SoftBank Robotics. *NAO - Documentation*. URL: <https://developer.softbankrobotics.com/nao6/nao-documentation/nao-developer-guide> (besucht am 21. 12. 2020).
- [3] URL: <https://spectrum.ieee.org/automaton/robotics/humanoids/aldebaran-robotics-founder-and-ceo-steps-down-softbank-appoints-new-leader> (besucht am 04. 01. 2021).
- [4] URL: <https://www.softbankrobotics.com/emea/en/pepper> (besucht am 04. 01. 2021).
- [5] *NAO (NAOqi-2.1) NAO - documentation*. URL: <https://developer.softbankrobotics.com/nao-naoqi-2-1/nao-documentation/nao-technical-guide/> (besucht am 04. 01. 2021).
- [6] P. Shahverdi, M. J. Ansari und M. T. Masouleh. "Balance Strategy for Human Imitation by a NAO Humanoid Robot". In: *2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM)*. 2017, S. 138–143. DOI: [10.1109/ICRoM.2017.8466225](https://doi.org/10.1109/ICRoM.2017.8466225).
- [7] A. M. Shayan u. a. "Design and Development of a Pressure-Sensitive Shoe Platform for Nao H25". In: *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*. 2019, S. 223–228. DOI: [10.1109/ICRoM48714.2019.9071802](https://doi.org/10.1109/ICRoM48714.2019.9071802).
- [8] SoftBank Robotics. *NAOqi - Developer guide*. Accessed: 2020-12-21. URL: <https://developer.softbankrobotics.com/nao6/naoqi-developer-guide> (besucht am 21. 12. 2020).
- [9] Version 2020.3. URL: <https://www.autodesk.de/products/inventor/overview?plc=INVPROSA&term=1-YEAR&support=ADVANCED&quantity=1#internal-link-what-is-inventor> (besucht am 31. 01. 2021).
- [10] URL: <https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Inventor-Help/files/GUID-D74F47F3-FE22-44EF-85BE-7C6B1F56DCF9-htm.html> (besucht am 31. 01. 2021).
- [11] Version 4.8.0. URL: <https://ultimaker.com/de/software/ultimaker-cura> (besucht am 31. 01. 2021).
- [12] Version R2020 Update 3. URL: <https://de.mathworks.com/discovery/what-is-matlab.html> (besucht am 31. 01. 2021).
- [13] URL: <https://de.mathworks.com/help/stats/exploratory-data-analysis.html> (besucht am 31. 01. 2021).
- [14] Z. Rigbi und L. Jilken. "The response of an elastomer filled with soft ferrite to mechanical and magnetic influences". In: *J. Magn. Magn. Mater* 37(3) (Juli 1983), 267–276. DOI: [10.1016/0304-8853\(83\)90055-0](https://doi.org/10.1016/0304-8853(83)90055-0).

- [15] Z. Rigbi und J. E. Mark. "Effects of a magnetic field applied during the curing of a polymer loaded with magnetic filler". In: *J. Polym. Sci., Polym. Phys. Ed.* 23(6) (Juni 1985), 1267–1269. DOI: [10.1002/pol.1985.180230618](https://doi.org/10.1002/pol.1985.180230618).
- [16] J. M. Ginder. "Encyclopedia of Applied Physics". In: Bd. 16. Wiley-VCH Verlag GmbH & Co KGaA, New York, New York, 1996. Kap. Rheology controlled by magnetic fields, 487–503. DOI: [10.1002/3527600434.eap402](https://doi.org/10.1002/3527600434.eap402).
- [17] James E Martin u. a. "Magnetostriction of field-structured magnetoelastomers". In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 74.5 Pt 1 (2006), S. 051507. ISSN: 1539-3755. DOI: [10.1103/physreve.74.051507](https://doi.org/10.1103/physreve.74.051507).
- [18] G. Filipcsei Z. Varga und M. Zrínyi. "Smart composites with controlled anisotropy". In: *Polymer* 46(18) (Aug. 2005), 7779–7787. DOI: [10.1016/j.polymer.2005.03.10](https://doi.org/10.1016/j.polymer.2005.03.10).
- [19] G. Filipcsei Z. Varga und M. Zrínyi. "Magnetic field sensitive functional elastomers with tuneable elastic modulus". In: *Polymer* 47(1) (Jan. 2006), 227–233. DOI: [10.1016/j.polymer.2005.10.139](https://doi.org/10.1016/j.polymer.2005.10.139).
- [20] John M. Ginder, William F. Schlotter und Mark E. Nichols. "Magnetorheological elastomers in tunable vibration absorbers". In: *Smart Structures and Materials 2001: Damping and Isolation*. Hrsg. von Daniel J. Inman. Bd. 4331. International Society for Optics und Photonics. SPIE, 2001, S. 103 –110. DOI: [10.1117/12.432694](https://doi.org/10.1117/12.432694).
- [21] H.X. Deng und X.L. Gong. "Adaptive Tuned Vibration Absorber based on Magnetorheological Elastomer". In: *Journal of Intelligent Material Systems and Structures* 18.12 (2007), S. 1205–1210. DOI: [10.1177/1045389X07083128](https://doi.org/10.1177/1045389X07083128).
- [22] N Hoang u. a. "Development of a torsional dynamic absorber using a magnetorheological elastomer for vibration reduction of a powertrain test rig". In: *Journal of Intelligent Material Systems and Structures* 24.16 (2013), S. 2036–2044. DOI: [10.1177/1045389X13489361](https://doi.org/10.1177/1045389X13489361).
- [23] M Kallio u. a. "Dynamic compression testing of a tunable spring element consisting of a magnetorheological elastomer". In: *Smart Materials and Structures* 16.2 (2007), S. 506–514. DOI: [10.1088/0964-1726/16/2/032](https://doi.org/10.1088/0964-1726/16/2/032).
- [24] John M. Ginder u. a. "Controllable-stiffness components based on magnetorheological elastomers". In: *Smart Structures and Materials 2000: Smart Structures and Integrated Systems*. Hrsg. von Norman M. Wereley. Bd. 3985. International Society for Optics und Photonics. SPIE, 2000, S. 418 –425. DOI: [10.1117/12.388844](https://doi.org/10.1117/12.388844).
- [25] Holger Böse, Raman Rabindranath und Johannes Ehrlich. "Soft magnetorheological elastomers as new actuators for valves". In: *Journal of Intelligent Material Systems and Structures* 23.9 (2012), S. 989–994. DOI: [10.1177/1045389X11433498](https://doi.org/10.1177/1045389X11433498).
- [26] Marc-André Keip, Paul Steinmann und Jörg Schröder. "Two-scale computational homogenization of electro-elasticity at finite strains". In: *Computer Methods in Applied Mechanics and Engineering* 278 (2014), S. 62–79. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2014.04.020>.

- [27] G Y Zhou und Q Wang. "Design of a smart piezoelectric actuator based on a magnetorheological elastomer". In: *Smart Materials and Structures* 14.4 (2005), S. 504–510. DOI: [10.1088/0964-1726/14/4/007](https://doi.org/10.1088/0964-1726/14/4/007).
- [28] G Y Zhou, K C Lin und Q Wang. "Finite element studies on field-dependent rigidities of sandwich beams with magnetorheological elastomer cores". In: *Smart Materials and Structures* 15.3 (2006), S. 787–791. DOI: [10.1088/0964-1726/15/3/014](https://doi.org/10.1088/0964-1726/15/3/014).
- [29] Meng G. Zhang Wm. et al Wei Kx. "Experimental investigation on vibration characteristics of sandwich beams with magnetorheological elastomers cores". In: *J. Cent. South Univ. Technol.* 15 (2008), 239–242. DOI: [10.1007/s11771-008-0354-7](https://doi.org/10.1007/s11771-008-0354-7).
- [30] Famin Qiu und Bradley J. Nelson. "Magnetic Helical Micro- and Nanorobots: Toward Their Biomedical Applications". In: *Engineering* 1.1 (2015), S. 021–026. ISSN: 2095-8099. DOI: <https://doi.org/10.15302/J-ENG-2015005>.
- [31] Tiantian Xu u. a. "Magnetic Actuation Based Motion Control for Microrobots: An Overview". In: *Micromachines* 6 (Sep. 2015), S. 1346–1364. DOI: [10.3390/mi6091346](https://doi.org/10.3390/mi6091346).
- [32] Guo Zhan Lum u. a. "Shape-programmable magnetic soft matter". In: *Proceedings of the National Academy of Sciences* (2016). ISSN: 0027-8424. DOI: [10.1073/pnas.1608193113](https://doi.org/10.1073/pnas.1608193113).
- [33] Wenqi Hu u. a. "Small-scale soft-bodied robot with multimodal locomotion". In: *Nature* 554.7690 (2018), S. 81–85. DOI: [10.1038/nature25443](https://doi.org/10.1038/nature25443).
- [34] Roland Fuhrer u. a. "Soft Iron/Silicon Composite Tubes for Magnetic Peristaltic Pumping: Frequency-Dependent Pressure and Volume Flow". In: *Advanced Functional Materials* 23.31 (2013), S. 3845–3849. DOI: <https://doi.org/10.1002/adfm.201203572>.
- [35] Yancheng Li u. a. "A highly adjustable magnetorheological elastomer base isolator for applications of real-time adaptive control". In: *Smart Materials and Structures* 22.9 (2013), S. 095020. DOI: [10.1088/0964-1726/22/9/095020](https://doi.org/10.1088/0964-1726/22/9/095020).
- [36] URL: https://www.supermagnete.de/magnete-zum-anschrauben/quadermagnet-40-x-40-x-4mm_CS-Q-40-40-04-N (besucht am 06.01.2021).