Praktikum3_v03tt

January 11, 2024

1 Praktikum 3 Intelligente Sensorsysteme

Tim Tiedemann, Thomas Lehmann, Tobias De Gasperis

Version 11.05.2023

2 Einfache intelligente Sensoren und Datenvorverarbeitung

Im Praktikum 3 geht es um die Ermittlung von Eigenschaften einiger komplexerer Sensoren, sowie die Datenvorverarbeitung und die Feature-Erzeugung.

Lesen Sie sich die Aufgaben gut durch. Sollten Sie eine Aufgabe nicht lösen können, so beschreiben Sie zumindest, wie weit Sie gekommen sind und auf welche Weise Sie vorgegangen sind.

Beachten Sie auf der methodischen Seite, dass Sie die jeweiligen Ergebnisse Ihrer Analysen kommentieren. Alle Diagramme sind korrekt zu beschriften.

Die Aufgaben sind direkt hier als Protokoll zu bearbeiten. Das abgegebene Notebook soll ausführbar sein. Daneben ist der PDF-Export des Notebook mit abzugeben.

Autoren des Protokolls: Haron Nazari, Anton Tchekov

3 Hintergrund

Aus den gesammelten Daten/Signalen kann man weitere Eigenschaften (Features) ableiten. Dazu müssen diese zum Teil erst geeignete vorverarbeitet werden, z.B. mit Filtern geglättet werden.

Im industriellen Umfeld kommen komplexere Sensorsysteme mit einer höheren Genauigkeit zum Einsatz. Die Integration und Inbetriebnahme ist oftmals nicht ganz klar, deshalb soll das exemplarisch ausprobiert werden und ein Vergleich zwischen Sensorsystemen vorgenommen werden.

Aus dem letzten Praktikum wurden Bewegungsdaten erfasst. Die Datenmenge ist für eine einfache Analyse zu umfangreich. Mit Hilfe der Principle Component Analysis (PCA) soll die Datendimension reduziert und die Ergebnisse interpretiert werden.

4 Vorbereitungsaufgaben

4.1 Komplexe Sensoren

Analysieren Sie die zugehörige Datenblätter und Handbücher (siehe Teams) für die Sensoren SensoPart FT 80 und Welotec OWTC1. Um was für Sensoren handelt es sich jeweils und welches

Messprinzip wird verwendet? Über welche Schnittstellen bekommt man die Messwerte? Welche weiteren Daten könnten für die Praktikumsaufgabe relevant sein? Hinweis: Im Labor steht keine RS485-Schnittstelle zur Verfügung.

Für den Sensor Welotec OWTC1 benötigen Sie noch eine Folge von Kommandos, um laufende Messwerte auslesen zu können. Hierzu hilft Ihnen das Dokument zu dem Sensor "Welotec OWTC-1" in MS-Teams/EMIL weiter. Wie muss die Schnittstelle konfiguriert werden und welche Kommandos müssen Sie senden?

Als weiteres System wir im Praktikum der Joy-Pi-Koffer eingesetzt, welcher gleich sehr viele Sensoren beinhaltet. Verschaffen Sie sich einen ersten Überblick über diesen Aufbau. Welches Teilsystem ist für die Entfernungsmessung geeignet?

1. SensoPart FT 80 RLA-500-S1L8
2. Welotec OWTC1
3. Koffer Joy-Pi von joy-it

Sensorenanalyseergebnisse:

5 SensoPart FT 80 RLA-500-S1L8

Der FT 80 RLA ist ein optischer Abstandssensor, welcher nach dem Triangulationsprinzip funktioniert. D.h. es wird mit deem Winkel der Reflektion, statt mit TOF gemessen. Der Sensor hat einen Analogausgang und eine RS485 Schnitstelle Typ 1.

Die Distanz in welcher der Sensor funktioniert sind 250 mm bis 750 mm mit 0.1% Auflösung. Der Sensor ist dazu auch noch einstellbar.

Man kann den Sensor mit 18 - 32 V DC betreiben. Der analoge Ausgang geht von 4 - 20 mA um die Distanz zu bestimmen.

6 Welotec OWTC1

Der OWTC1 ist auch ein optischer Abstandssensor, welcher mit TOF (Phase shift) funktioniert. Er kann über RS-232, RS-422 und SSI (V2(H) only) Schnitstellen angesprochen werden. Er besitzt dazu auch noch einen digitalen Ausgang zur Fehleranzeige. 1 Programmierbaren Analogausgang von 4 - 20 mA. Die Baudrate ist 19200 und hat 7E1 pro Zeichen (ASCII) in Setting 7.

Die Distanz in welcher der Sensor funktioniert ist von 0.005 bis 500 m und wird mit 9 - 30 V DC betrieben.

Der Sensor hat einen Drehschalter, mit dem man die ID einstellen kann von einem Modul (von 0 bis 9). Es hat auch einen Reset-Schalter.

6.1 Benötigte Befehle:

- Modulidentifikation `N` : Man kann die Module mit dem ID Schalter addressieren.
- Parameter Seperator `+` : Man kann Parameter zu Befehlen hiermit trennen.
- STOP/CLEAR `sNc` : Stoppt und Cleared
- Einzel-Distanzmessung `sNg` : Bringt den Sensor dazu eine Distanzmessung auszuführen. Bei Erfolg wird `gng+xxxxxxxx` zurückgegeben wobei die `x`'e die Distanz in 1/10 mm sind.

6.2 Datensätze

Schliessen Sie die Aufnahme der Messdaten für die verschiedenen Bewegungen aus dem Praktikum 2 ab, damit Sie für die folgenden Aufgaben geeignete Daten mit 12 Dimensionen haben.

7 Im Labor

7.1 Datenvorverarbeitung und Feature-Generation

Erstellen Sie Python-Scripte für die Vorverarbeitung Ihrer Messdaten mittels Filter.

Verwenden Sie für die Anwendung eines Filters den Datensatz für das Bewegungsprofil 2 aus dem letzten Praktikum.

Filtern Sie die Daten von einem der Beschleunigungssenoren für alle drei Achsen mit jeweils einem IIR-Filter erster Ordnung (siehe Vorlesung); mindestens 1000 Samples. Implementieren Sie das Filter so, dass man es auch für einen Live-Betrieb (jeweils ein Sample pro Schleifendurchlauf) verwendet werden kann.

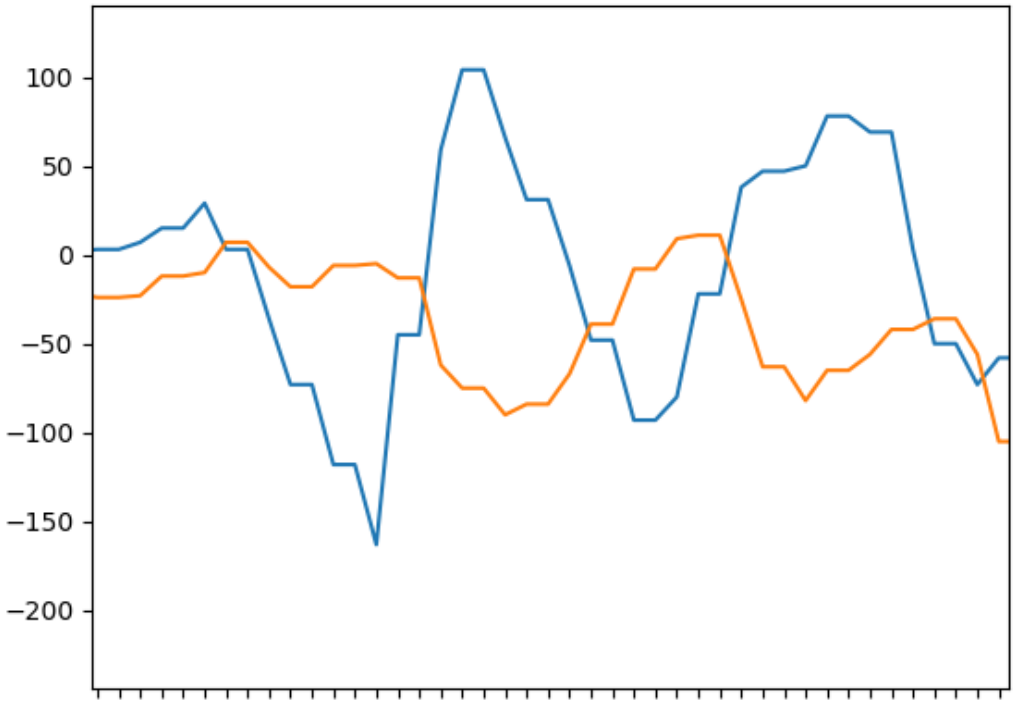
Code

[ ]:
```
import csv
import matplotlib.pyplot as plt

# Alpha as a
a = 0.5
one_minus_a = 1 - a

last_x = 0
last_y = 0
last_z = 0

sample = []
time = []
filtered_acc_x = []
filtered_acc_y = []
filtered_acc_z = []

current_row = 0
print('Sample, Acc_x, Acc_y, Acc_z')
with open('itsboard_drehungzumaufrichten_2.csv') as csv_file:
  csv_reader = csv.reader(csv_file, delimiter=',')
  for row in csv_reader:
    if current_row > 0:

      # Get current value with the help of the last
      x = int(row[2])
      y = int(row[3])
      z = int(row[4])
      x1 = a * x + one_minus_a * last_x
```

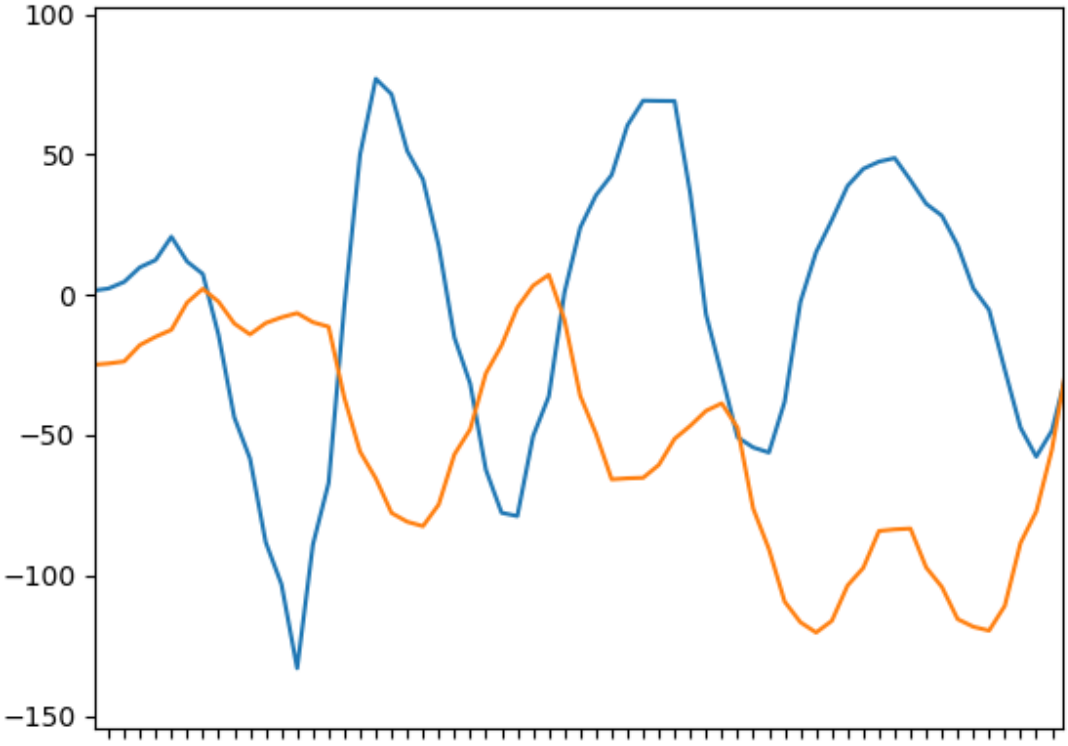
```
      y1 = a * y + one_minus_a * last_y
      z1 = a * z + one_minus_a * last_z

      # Print the values and update the last ones
      print(f'{row[0]}, {x1}, {y1}, {z1}')
      last_x = x1
      last_y = y1
      last_z = z1

      # Build Lists
      sample.append(row[0])
      time.append(row[1])
      filtered_acc_x.append(x1)
      filtered_acc_y.append(y1)
      filtered_acc_z.append(z1)
    current_row += 1


plt.plot(sample, filtered_acc_x, sample, filtered_acc_y, sample, filtered_acc_z)
plt.show()
# IIR Filter : aX(i)+(1-a)y(i-1) mit a = Ts / tLP + Ts
# Ts = Sample Period
# tLP filter time constant
```

Stellen Sie auch die sich ergebenden gefilterten Daten über die Zeit in Sekunden dar. Der Bereich darf auf einen relevanten Bereich beschränkt werden.

Vorher:



Nachher:



Welchen Faktor wählen Sie für α? Warum haben Sie diesen Wert von α gewählt?

Parameterwahl:

Es wurde mit dem Faktor rumgespielt bis die hohen Frequenzen zufriedenstellend herausgefiltert waren.

Welche algorithmische Komplexität/ungefähre Komplexität in Floating-Point-Operations hat Ihre Filterfunktion?

Berechnungsaufwand

[ ]:
```
x1 = a * x + one_minus_a * last_x
y1 = a * y + one_minus_a * last_y
z1 = a * z + one_minus_a * last_z
```

sind die 3 Hauptberechnungen im Code. Dadurch das Alpha eine Kommazahl ist, werden `a * x`, `a * y`, `a * z` als 3 Floating-Point Operationen gezählt, zudem ist die Variable `one_minus_a` auch eine Fließkommazahl. Das heißt, wir haben insgesamt 6 Floating-Point Multiplikationen und 3 Additionen.

7.2 Komplexe intelligente Sensoren

Es sind im Labor ggf. nicht alle Sensorsysteme verfügbar, da diese auch in Forschungsprojekten verwendet werden. Die Ausgabe eines Sensors erfolgt erst, wenn ausreichend Kenntnisstand über das Vorgehen bei der Inbetriebnahme vorhanden ist.

Bestimmen Sie exemplarisch mit den Sensoren Entfernungen und vergleichen Sie diese mit einer Referenzmessung.

7.2.1 Senso Part

Versuchen Sie (in Absprache mit dem Tutor), den Sensor in Betrieb zu nehmen. Welche Optionen der Messwertausgabe verwenden Sie?

Wir verwenden die analoge Schnitstelle (4 - 20 mA) und messen den Spannungsabfall über einen 100 Ohm Widerstand.

R = 100 Ohm

I = U / R

Die Ausgabe des Sensors war invertiert.

Unsere Werte im Vergleich mit den idealen Werten:

Abstand	Spannung	Strom	Ideal
250 mm	2 V	20 mA	20 mA
350 mm	1.7 V	17 mA	16.8 mA
500 mm	1.3 V	13 mA	12 mA
600 mm	840 mV	8.4 mA	8.8 mA
750 mm	400 mV	4 mA	4 mA

7.2.2 Welotec

Versuchen Sie (in Absprache mit dem Tutor), den Sensor in Betrieb zu nehmen.

Messergebnisse:

Abstand	Raw-Ausgabe in 1/10 mm
250 mm	2524
350 mm	3532
500 mm	5050
600 mm	6051
750 mm	7553

Unsere Vermutung ist, dass die Mess-Ungenauigkeit nicht durch den Sensor, sondern durch den Versuchsaufbau bedingt ist.

7.2.3 Joy-Pi-System

Versuchen Sie (in Absprache mit dem Tutor), den Aufbau in Betrieb zu nehmen. Öffnen Sie ein Terminal und schauen Sie sich die Dateien unter `/Desktop/Joy-Pi/` an. Testen Sie das Script `distance.py`. Schauen Sie sich das Script an und versuchen Sie, die einzelnen Schritte nachzuvollziehen. Kopieren Sie ggf. das Script in eine eigene Datei `distance_NAME_DATUM.py` und nehmen Sie darin Änderungen vor, um das Verhalten besser verstehen zu können. Erklären Sie die Funktionsweise der durchgeführten Messung:

Der Joy-Pi benutzt einen Ultraschallsensor um die Distanz zu berechnen.

Formel Zusammenhang: Geschwindigkeit `v` / Zeit `t` / Strecke `s`

```
v = s / t
```

gegeben ist:

Schallgeschwingikeit: `v = 343,2 m/s`

Gemessen wurde die Zeit des Echos `t` in Sekunden.

gesucht: Strecke `s` in cm

Umgestellt nach gesuchter Größe `s`:

```
s = v * t
```

Der Code verwendet als Umrechnungsfaktor die halbe Schallgeschwingikeit mal 100 (17150), da Zeitdauer für die doppelte Distanz (Hin- und Rückweg) gemessen wird, und die Einheit des Ergebnisses in cm Ausgegeben werden soll.

Abstand	Sensor Ausgabe
250 mm	245 mm
350 mm	347 mm
500 mm	496 mm
600 mm	590 mm

Abstand	Sensor Ausgabe
750 mm	756 mm

7.2.4 Vergleichsmessung von Sensorsystemen

Versuchen Sie, in einem Aufbau den zuletzt genutzten Sensor aus dem Joy-Pi-System und einen der anderen Sensoren parallel zu betreiben. Was ergeben die Messungen des einen und des anderen Sensors bei gleicher Objektentfernung?

Distanz	JoyPi	Welotec	Sensopart
250 mm	245 mm	252.4 mm	250 mm
350 mm	347 mm	353.2 mm	343 mm
500 mm	496 mm	505.0 mm	468 mm
600 mm	590 mm	605.1 mm	612 mm
750 mm	756 mm	755.3 mm	750 mm

7.2.5 Vergleich der Sensoren über Kenndaten

Vergleichen Sie die Leistungsdaten der Distanzsensoren, z.B. in einer Tabelle. Nehmen Sie hierzu auch den Entfernungsmesser mit auf, den Sie im ersten Praktikumsversuch untersucht haben. Bewerten Sie kurz die Sensoren.

Kategorie	JoyPi	Welotec	Sensopart	GYP2Y0A21
Messdistanz	2 - 400 cm	0.5 mm - 500 m	250 - 750 mm	10 - 80 cm
Stromverbrauch	15 mA @ 5 V	0.5 A @ 9 - 30 V	40 mA @ 24 V	30 mA @ 5 V
Größe	45 x 20 x 15 mm	150 x 80 x 54 mm	55 x 83 x 25 mm	29 x 13 x 13 mm
Spannung	5 V	9 - 30 V	24 V	4.5 - 5.5 V
Interface	TTL pulse	Serial	Analog	Analog
Genauigkeit	3 mm	1.5 mm	?	?

Die beste Leistung hat der Welotec, welcher eine große Messdistanz hat und auch eine hohe Genauigkeit, jedoch ist dieser Groß und verbraucht viel Strom.

Joy-Pis Ultraschallsensor ist einfach zu benutzten und hat einen sehr geringen Stromverbrauch, kann also direkt von einer MCU betrieben werden, dabei ist die Reichweite ist auch für größere Distanzen gut, aber es ist nicht sehr genau und muss auf gerade Oberflächen treffen damit es korrekte Ergebnisse liefert.

Zu dem Sensopart haben wir im Datenblatt keine Infos zur Genauigkeit gefunden, war aber in unserem Versuch auf +/- einige Zentimeter genau. Die anderen Daten, wie die hohe Betriebspannung und die Messdistanz sind im Vergleich schlechter als die anderen Sensoren.

Der GYP2Y0A21 sieht auf dem Papier gut aus, war aber in der Benutzung sehr ungenau, jedoch hat er einen geringen Stromverbrauch und ist sehr kompakt, Die analoge Schnittstelle ist auch sehr einfach zu benutzen.

7.3 Dimensionsreduktion

Im Praktikum 2 haben Sie Daten von einem Sensorsystem mit 12 Dimensionen aufgenommen. Für die Erkennung der Bewegungsprofile sind evtl. nicht alle Dimensionen relevant. Mit Hilfe der PCA sollen Sie die relvanten Dimensionen/Features identifizieren.

In dieser Analyse sollen zunächst nur die Datensätze verwendet werden, bei denen das System auf dem Tisch liegend bewegt wurde.

Führen Sie für die Datensätze (4-6) jeweils die Hauptkomponentenanalyse (PCA) mit Python durch. Hierzu können Sie Funktionen aus dem Paket `numpy` verwenden oder direkt die Klasse PCA aus dem Paket `scikit`.

```
# numpy
x  = np.array(data)
x_ = x - x.mean(axis=0)
r = np.cov(x_.T)
#print 'r =', r
evals, evecs = np.linalg.eig(r)
print(evals, evecs)
pca_score = np.matmul(x_, evecs)

# Scikit
# Dim festlegen: pca = PCA(n_components=3)
# sonst besser so:
pca = PCA()
pca.fit(x)
pca_ratio = pca.explained_variance_ratio_
eigVec = pca.components_
pca_score = pca.transform(x)
```

Achtung: Je nach verwendeter Bibliothek wird keine Eigenwertzerlegung sondern eine Singulärwertzerlegung durchgeführt. Wenn im folgenden von "Eigenwert" die Rede ist, können Sie stattdessen die "erklärte Varianz" (absolut oder relativ) verwenden.

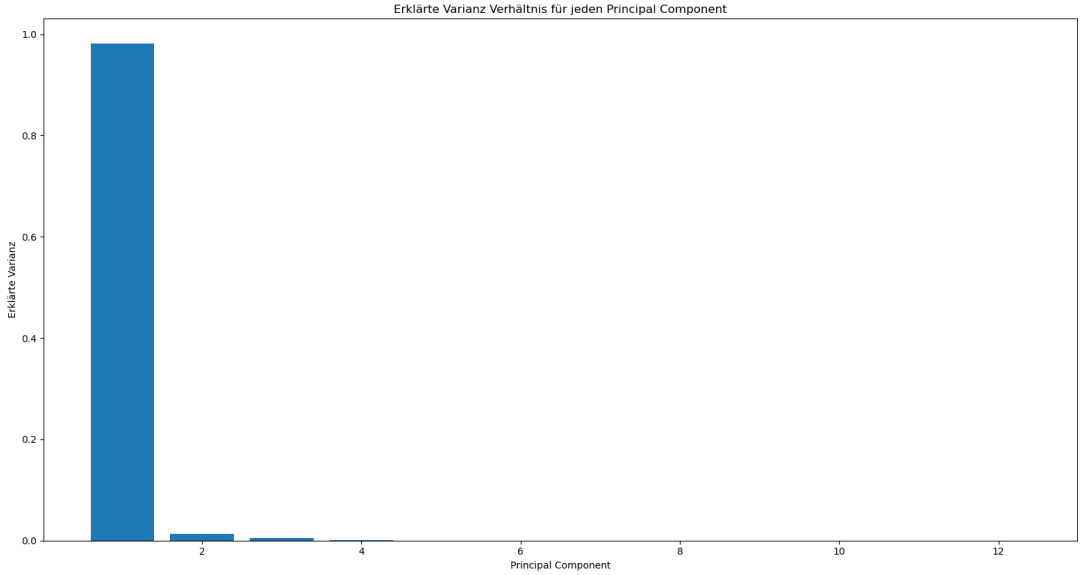
Wählen Sie je einen Datensatz mit einer einfachen (linearen) und einen mit einer komplexen Bewegung aus. Erstellen Sie für beide Datensätze jeweils den Scree-Plot der Eigenwerte (oder der erklärten Varianz, je nach verwendetem Verfahren) als Ergebnis einer PCA auf den Datensätzen. Kann man irgendeine Aussage auf Grund der Plots im Vergleich zu dem Bewegungsmuster machen?

[ ]:
```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pandas as pd

# itsboard_drehungzumaufrichten_2.csv

# Load your data
data = pd.read_csv('itsboard_drehungzumaufrichten_2.csv')
data = data.iloc[:, 2:]
```

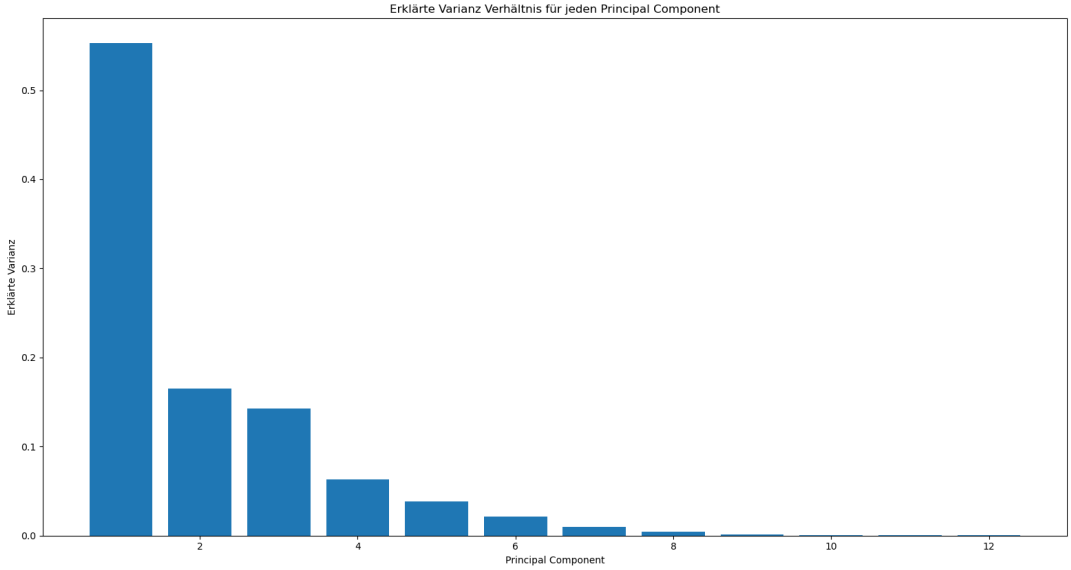
```
# Convert data to numpy array
x = data

scaler = StandardScaler()
x_standardized = scaler.fit_transform(x)

# Scikit
# Dim festlegen: pca = PCA(n_components=3)
# sonst besser so:
pca = PCA()
pca.fit(x_standardized)
pca_ratio = pca.explained_variance_ratio_
eigVec = pca.components_
pca_score = pca.transform(x)

print(eigVec)

# Plot
plt.bar(range(1, len(pca_ratio) + 1), pca_ratio)
plt.xlabel('Principal Component')
plt.ylabel('Erklärte Varianz')
plt.title('Erklärte Varianz Verhältnis für jeden Principal Component')
plt.show()
```

Wählen Sie nun einen Datensatz für die weitere Bearbeitung aus. Vergleichen Sie bei dem Datensatz den Scree-Plots der "erklärten Varianz" mit und ohne Normalisierung: Erscheint für Sie eine Normalisierung sinnvoll? Wenn ja: in welcher Form? Welche Normalisierung haben Sie aus welchem Grund gewählt?

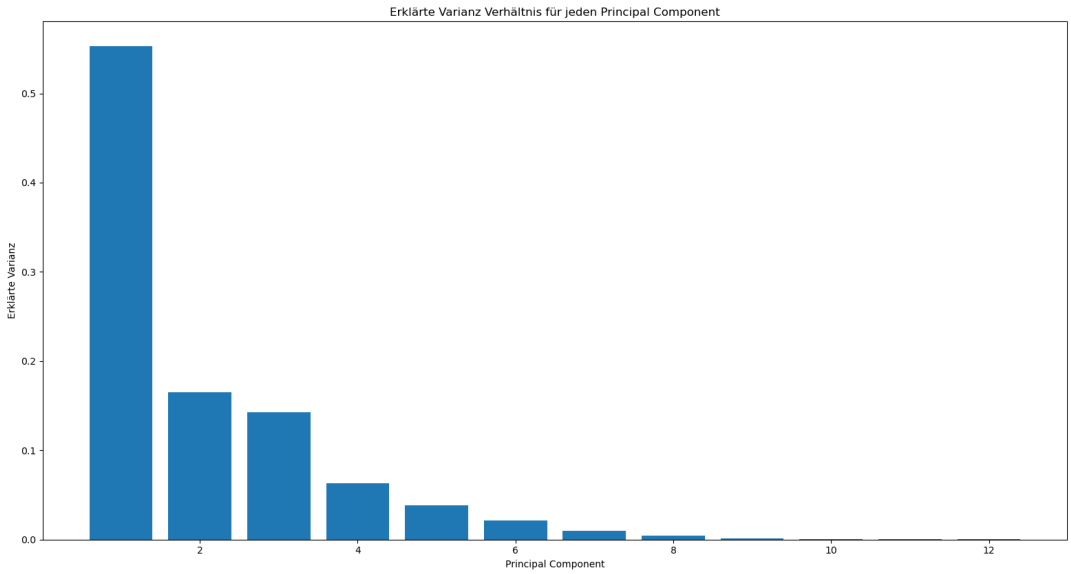
Ohne Normalisierung:



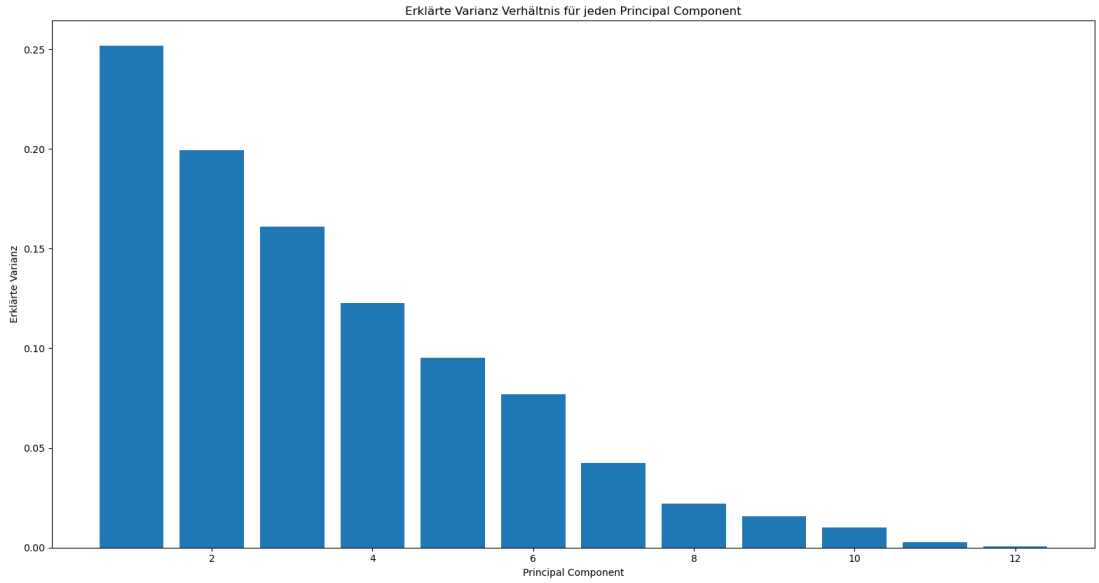
Mit Normalisierung:



Simple Bewegung:



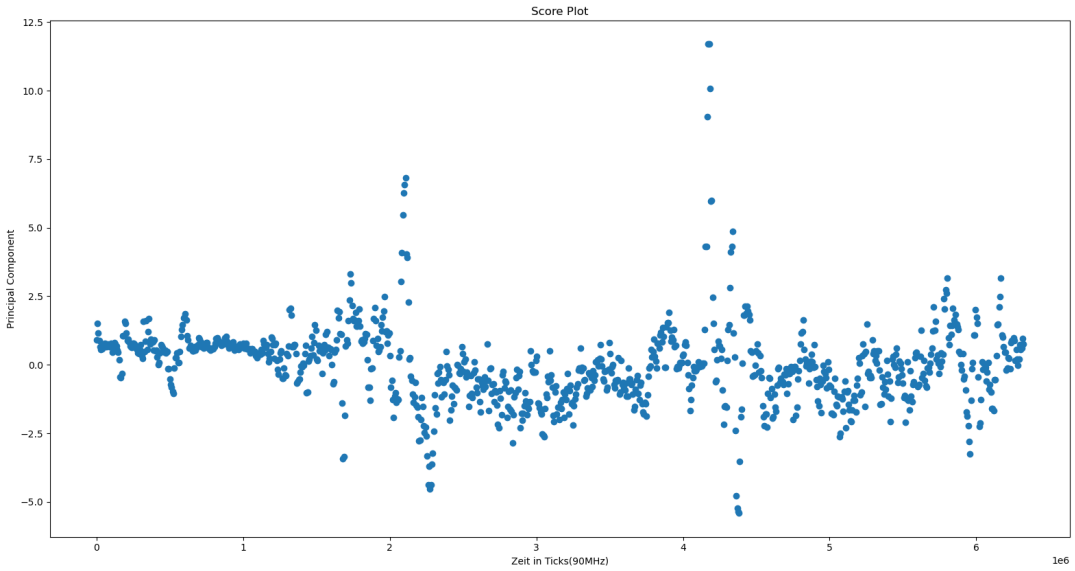
Komplexe Bewegung:



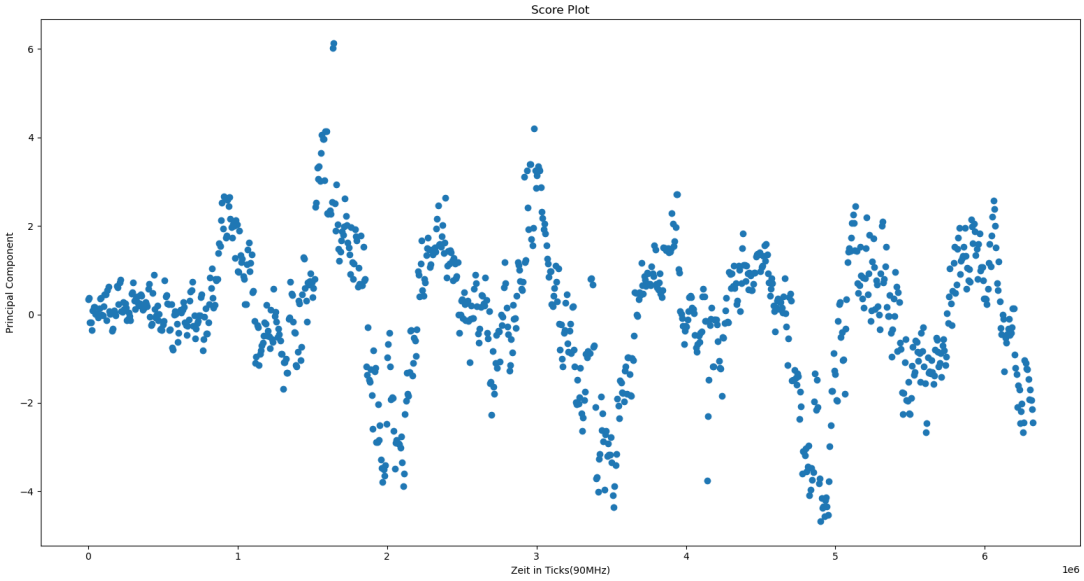
Wir haben den StandardScaler von SciKit-Learn gewählt (Standardisierung). Eine Normalisierung ist sinnvoll, da die Daten alle verschiedene Einheiten haben und da wir sie mithilfe der PCA vergleichen wollen, sollten die Daten die gleiche Größenordnung besitzen.

Verwenden Sie dieselbe Art von Normalisierung, oder eben keine, bei den ausgewählten Datensätzen und stellen Sie den Score-Plot dar. Stellen Sie den Score auch auf der Zeitachse dar. Können Sie die Bewegungsabschnitte zuordnen? Gibt es Ähnlichkeiten zu den rohen Sensordaten?

Simple Bewegung:



Komplexe Bewegung:



Man kann an beiden Score-Plots die Bewegungsabschnitte wiedererkennen. Bei der simplen Bewegung wurde Bewegung 2 also das Aufrichten genutzt, dort sieht man jeweils wann es aufgerichtet und wieder hingelegt wurde.

Bei der komplexen Bewegung wurde Bewegung 4 also das Geradliniege hin- und her auf dem Tisch genutzt, dort kann man sehr deutlich die Richtungswechsel sehen.

Wieso werden nicht alle 12 Dimensionen benötigt? Haben Sie eine Vermutung? Bedenken Sie, wie der Datensatz zu Stande gekommen ist: Wie hatten Sie das Sensor-Board bewegt? In welchem Bereich variierten die jeweils drei Dimensionen der Beschleunigung, der Winkelgeschwindigkeit und

die Ausrichtung des Magnetfeldes? Betrachten Sie ggf. auch die Loadings aus den Eigenvektoren bei der Begründung.

Weil das Board entlang einer Richtung bewegt wurde wird diese von der PCA als die wichtigste Achse ausgewählt in den anderen Achsen ist es meistens nur Rauschen, jedoch wurden die Magnetfeldsensoren zum Magnetfeld der Erde anders ausgerichtet zum Beispiel bei dem Aufrichten des ITS-Boards und dadurch wurden diese auch als wichtig von der PCA angesehen. Die andere Achsen schlagen jedoch wenig aus und meist wegen der leichten Erschütterungen beim Schieben über den Tisch (Beim Gyro).

Eigenvektor von Principal Component Nr. 1:

	X	Y	Z
Accelerometer 1	-5.06924739e-01	-3.03963910e-01	8.79527592e-02
Gyroskop	2.12988004e-01	-2.00861911e-01	6.79464981e-05
Accelerometer 2	-2.25665873e-01	5.11229839e-01	1.58932992e-02
Magnetometer	4.70718766e-01	-1.51826267e-01	5.21128443e-03

Die Eigenvektoren bei der Bewegung 4, bei der Principal Component 1 werden die Dimensionen Acc_x und Acc_2y und Mag_x am stärksten gewichtet. Bei den x-Achsen macht das ja Sinn, da die Bewegung entlang der x-Achse erfolgte, aber bei Acc_2y ist uns aufgefallen, dass es sich Invertiert zu Acc_x verhält, also das heißt, dass die Sensoren relativ zueinander 90 Grad gedreht sind. Man kann das an dem unteren Bild gut erkennen: