HAW HAMBURG

BACHELOR THESIS
Jonas Ningelgen

Ermittlung der Synchronität der Bewegungen bei Rudermannschaften mittels Sensorik und Pose Estimation

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Jonas Ningelgen

Ermittlung der Synchronität der Bewegungen bei Rudermannschaften mittels Sensorik und Pose Estimation

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas Lehmann
Zweitgutachter: Prof. Dr. Kai von Luck

Eingereicht am: 16. Juni 2022

Jonas Ningelgen

Thema der Arbeit

Ermittlung der Synchronität der Bewegungen bei Rudermannschaften mittels Sensorik und Pose Estimation

Stichworte

Human Pose Estimation, Rudern, Künstliche Intelligenzen, Bildverarbeitung, Sensorik, Beschleunigung, Sport, Bewegungsanalyse

Kurzzusammenfassung

Im Rudersport sind bisherige Bewegungsanalysewerkzeuge zur Ermittlung der individuellen und kollektiven Rudertechnik durch ihre Komplexität in ihrem Angebot stark begrenzt. Daher sind Bewegungsanalysen nur für den olympischen Spitzensport verfügbar.
Die vorliegende Forschungsarbeit beschäftigt sich mit der Analyse von kollektiver Rudertechnik, also der Synchronität der Bewegungen zwischen Rudernden.
Es wird ein prototypisches System für die Messung der kollektive Rudertechnik entwickelt. Hierfür wird geprüft, ob Beschleunigungssensoren, sowie Pose Estimation über Bildverarbeitung, für die Ermittlung von Synchronität bei Rudermannschaften geeignet sind.
Meine Ergebnisse zeigen, dass Synchronität für Rudermannschaften durch die Verarbeitung von Beschleunigungsdaten bewertbar und folglich vergleichbar ist. Im Gegensatz wird herausgearbeitet, aus welchen Gründen die Synchronität durch Pose Estimation nicht ermittelt werden konnte und unter welchen Umständen dies möglich wäre.
Durch Weiterentwicklung des Systems, kann ein Werkzeug geschaffen werden, welches Bewegungsanalysen für den Breiten- bis zum Spitzensport zugänglich macht.

Jonas Ningelgen

Title of Thesis

Determination of the Synchronicity of Movements in Rowing Teams With Sensor Technology and Pose Estimation

Keywords

Human Pose Estimation, rowing, artificial intelligences, image processing, sensor technology, acceleration, sports, motion analysis

Abstract

In rowing, previous motion analysis tools for determining individual and collective rowing technique are severely limited in what they can offer due to their complexity. Therefore, motion analysis tools are only available for elite Olympic sports.

The present research deals with the analysis of collective rowing technique, i.e. the synchrony of movements between rowers.

A prototypical system for the measurement of collective rowing technique is developed. For this purpose, it is tested whether accelerometers, as well as pose estimation via image processing, are suitable for the determination of synchrony in rowing teams.

The results show that synchrony for rowing teams can be assessed and consequently compared by processing acceleration data. In contrast, it is worked out why synchrony could not be determined by pose estimation and under which circumstances this would be possible.

By further development of the system, a tool can be created that makes motion analysis accessible for amateur to elite sports.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Der Einsatz moderner Technologie im Sport findet immer mehr Verbreitung. Pose Estimation Verfahren werden bereits zur Bewegungsanalyse genutzt, wie beispielsweise in einer aus Beijing stammenden Arbeit, in der ein Scoring für die Bewegungen im Tanztraining genutzt wird [9]. Ähnliche Technologien werden darüber hinaus zum Verbessern der Yoga Fertigkeiten angewandt. Im Rudersport werden keine Bildverarbeitungsverfahren, sondern hochtechnische sensorbasierte Werkzeuge eingesetzt, für dessen Auswertung man allerdings technisch versierte Sportwissenschaftler benötigt, sodass nicht jeder Bewegungsanalysen ausführen kann.

Das Wettkampfsziel im olympischen Rudersport ist es, sich mit einer hohen Bootsgeschwindigkeit vom Startblock bis über die Ziellinie zu bewegen. Dabei lässt sich die Geschwindigkeit eines Ruderbootes durch verschiedene Faktoren beeinflussen. Neben der Kraft und der Ausdauer der Sportler spielt die Rudertechnik eine wichtige Rolle. Die Technik einer Rudermannschaft lässt sich in die individuelle und in die kollektive Technik unterteilen. Die individuelle Rudertechnik beschreibt die Umsetzung eines Ruderleitbildes, also der Optimalvorstellung der Ruderbewegung einer einzelnen Person. Die kollektive Technik wiederum beschreibt die Gemeinsamkeit, folglich die Synchronität der Bewegung mehrerer Personen innerhalb eines Mannschaftsbootes.

Bisher sind die gängigen Arten der Messung von Synchronität eingeschränkt. Am häufigsten wird die Synchronität einer Rudermannschaft durch das geschulte Auge der Trainer eingeschätzt. Ein anderes, weit verbreitetes Mittel ist die Videoanalyse, also das Pausieren eines Videos und dem manuellen Einzeichnen von Linien in den einzelnen Sequenzen. Eine technisch anspruchsvolle Alternative ist das Messboot-Verfahren, für welches Sportwissenschaftler mit technischem Hintergrund benötigt werden, die die Ruderboote mit verschiedensten Sensoren ausrüsten. Weil das Ausrüsten der Boote komplex ist, werden

oft nicht die eigenen Boote der Mannschaften genutzt, sondern das mit den Sensoren ausgestattete Boot weitergegeben, welches nicht optimal auf die Mannschaft eingestellt ist und somit die Ergebnisse der Mannschaft verfälscht.

1.2 Ziel der Arbeit

Ziel der vorliegenden Arbeit ist es Anhand des Rudersports die Bewegungsanalyse mithilfe von Bildverarbeitung und einem sensorbasierten Ansatz zu erweitern und damit die Synchronität der Bewegung zwischen Personen objektiv ermitteln und bewerten zu können. So soll die Entwicklung der kollektiven Technik durch die Anpassung der individuellen Bewegungen der Rudernden sichtbar gemacht werden.

1.3 Aufbau der Arbeit

Untergliedert wird diese Thesis in fünf Kapitel. Zunächst wird in der Einleitung das bestehende Problem der Thematik, sowie die Zielsetzung der Arbeit erläutert. Im nächsten Kapitel, der Analyse, soll der Bestand von, im Kontext dieser Arbeit relevanter, wissenschaftlicher Literatur mitsamt relevanter Lösungsansätze aufgearbeitet werden. Hierfür wird herausgearbeitet was Synchronität im Rudersport ist und wie sie bewertet werden kann. Daraus wird abgeleitet, ob die vorgestellte Lösung umsetzbar ist. Darauf aufbauend werden im Kapitel Design, zwei verschiedene Systeme entwickelt, die dazu dienen die Synchronität der Bewegung einer Rudermannschaft zu messen und zu bewerten. Ein Ansatz wird hierfür Bildverarbeitung nutzen um die Synchronität aus der jeweiligen Pose der Rudernden zu ermitteln. Dieser Ansatz wird mit Sensortechnik ergänzt, um einen Vergleich zu ermöglichen. Im zweiten System werden Sensoren am Unterarm der Rudernden befestigt, um dort die jeweilige Beschleunigung zu messen, damit diese miteinander verglichen werden können. Diese Lösung wird im vierten Kapitel, der Evaluation auf ihre Schwächen und Stärken untersucht und im Gesamtkontext bewertet. In Kapitel fünf, dem Fazit, wird die Arbeit zusammengefasst und ein Ausblick für mögliche auf dieser Thesis aufbauende, Arbeiten gegeben.

2 Analyse

2.1 Bewegungsablauf im Rudern

Rudern zeichnet sich durch eine zyklische Bewegung ab, die sich in den Freilauf, die vordere Umkehr, den Durchzug und die hintere Umkehr unterteilen lässt. Die Ruderbewegung beginnt, wie es in Abbildung 2.1 links zu sehen ist, mit der Rücklage. Die Ruderblätter befinden sich waagerecht über dem Wasser.

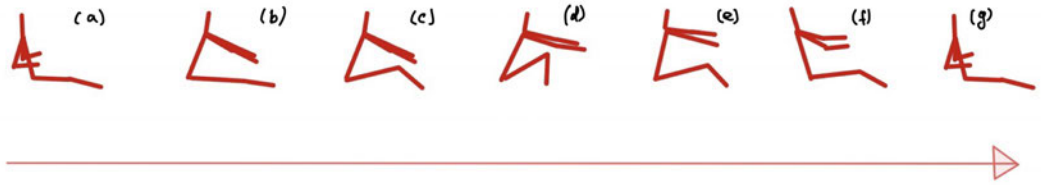

Abbildung 2.1: Bewegungsablauf Rudern

Die Griffe der Ruder sind am Körper, der Oberkörper ist leicht zurück gelehnt und die Beine sind gestreckt (a). Von hier aus beginnt der sogenannte Freilauf bis zum Wiedereintauchen der Blätter. Die Arme werden gestreckt, der Oberkörper nach vorne gelehnt (b). Als Nächstes werden die Beine gebeugt (c), diese Bewegung wird auch das Vorrollen genannt. Während des Vorrollens werden die Blätter aufgedreht, sodass diese senkrecht stehen, wenn die Unterschenkel senkrecht ausgerichtet sind (d). In dieser Position werden die Blätter ins Wasser eingetaucht, was auch als Vordere Umkehr gilt. Der sogenannte Durchzug beginnt und dauert an, bis die Blätter wieder aus dem Wasser gehoben werden. Der Durchzug beginnt mit dem Strecken der Beine (e). Leicht versetzt folgt das Zurücklehnen des Oberkörper, also der sogenannte Rückschwung. Die Arme bleiben gestreckt bis die Hände über den Knien sind (f). Dann beginnen sich die Arme zu beugen, der Oberkörper bleibt stehen, die Beine bewegen sich bis zu Streckung weiter. Die Arme und Beine beenden den Durchzug gemeinsam (g). In der Rücklage werden die Blätter senkrecht aus dem Wasser gehoben und abgedreht, sodass diese wieder waagerecht stehen.

Diese Bewegung, die auch als hintere Umkehr gilt, beendet den einzelnen Ruderschlag [17].

2.2 Synchronität im Mannschaftsrudern

Untersucht wird die Synchronität der Bewegung zwischen Rudernden innerhalb eines Mannschaftsbootes. Eine synchrone Bewegung der Mannschaft optimiert die Geschwindigkeit eines Mannschaftsbootes. Wie Boucher, Labré und Clanet in ihrer Arbeit Row bots bestätigten, erzeugt ein synchroner Ruderschlag mehr Vortrieb für ein Ruderboot, als ein asynchroner [1]. Im Rudersport wird viel Zeit in die Perfektion der Synchronität investiert, um die Geschwindigkeit der Mannschaftsboote zu steigen. Allerdings gibt es bisher kein allgemeingültiges Mittel zur Messung von Synchronität. Die Entwicklung der Synchronität wird in der Breite des Rudersports subjektiv wahrgenommen und darauf aufbauend bewertet. Um Objektivität gewährleisten zu können, sollte ein sachlicher Maßstab für Synchronität eingeführt werden. In seiner Masterthesis nennt Doeksen zum Einen die Möglichkeit Ankerpunkte in einem Ruderschlag zu wählen und die zeitliche Differenz zwischen den Rudernden zu dem Ankerpunkt zu stoppen. Zum Anderen benennt er die Möglichkeit die jeweilige Dauer der Freilauf- und der Zugphasen zu messen und zu vergleichen [6].

In dieser Arbeit wird der erste Ansatz, Ankerpunkte innerhalb eines Ruderschlages für die Bildverarbeitung festzulegen, als auch die Sensorik genutzt. Dieser Ansatz bietet den Vorteil, dass besonders prägnante Momente des Ruderschlages verglichen werden können. Darüber hinaus können beliebig viele Ankerpunkte innerhalb eines Schlages festgelegt werden.

Als Grundlage der Synchronität wird der Rhythmus des Schlagmannes herangezogen, da dieser im Heck, für den Takt verantwortlich ist, wie in Abbildung 2.2 zu sehen ist. Die Position des Schlagmannes wird hier bildlich in grün jeweils über die Positionen der anderen drei Rudernden gelegt, während die anderen Rudernden in rot eingezeichnet sind. Die Positionen sind von einer realen Rudermannschaft abgezeichnet, sodass hier reale Abweichungen zu sehen sind. Die anderen Mitglieder der Mannschaft haben die Aufgabe, dem Rhythmus zu folgen. Folglich wird die Differenz in der Synchronität der Rudernden zum Schlagmann als Abweichung in Millisekunden gemessen. Grundsätzlich sollte ein Grenzwert für eine als synchron geltende Mannschaft wissenschaftlich ermittelt

werden. In der Literatur konnte kein konkreter Messwert für eine übereinstimmende Synchronität gefunden werden. Für diese Arbeit gilt eine Mannschaft mit einer Abweichung zum Schlagmann von bis zu 100 Millisekunden als Synchron.

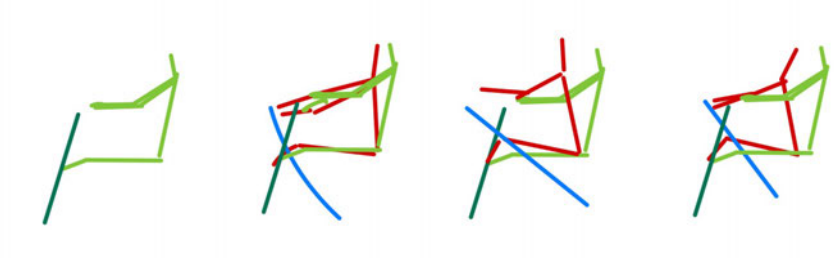

Abbildung 2.2: Asynchrone Bewegung: Die grüne Figur stellt die Bewegung des Schlagmannes dar. Die rote Figur stellt die abweichende Bewegung eines Mannschaftsmitglieds dar

2.3 Bildverarbeitung - Pose Estimation

Human Pose Estimation (HPE) beschäftigt sich mit dem Erkennen, Zuordnen und Verfolgen, also dem Lokalisieren von menschlichen Körpern und deren Schlüsselpunkten (SP), in Bildern und Videos [13, 7].

HPE lässt sich als Teilgebiet von Computer Vision (CV) und dem der Künstlichen Intelligenzen (KI) zuordnen, da die besten HPE Frameworks mit Deep Learning (DL) arbeiten [13]. Unter KI versteht man die Intelligenz, die von Maschinen erbracht werden soll und welche meist die menschlichen kognitiven Fähigkeiten, wie das Lernen und das Lösen von Problemen, nachahmt [14]. Um diese Aufgaben zu erreichen, nutzt DL, als Teildisziplin von KI, tiefe neuronale Netze zu der Modellerstellung, die eine hohe Genauigkeit, z.B. auch für das erkennen von Emotionen in einem Gesicht, bieten [12, 24]. CV ist ein eigenes Forschungsfeld mit dem Ziel, Computern und Maschinen das Sehen, also das Erkennen und Zuordnen von Objekten und deren Position zu ermöglichen [13]. Traditionelle CV Ansätze nutzen Mustererkennung basierend auf mathematischen Filteroperationen. Diese sind darauf ausgelegt, Merkmale von Objekten abzuspeichern und diese in einem Bild wiederzufinden. Dieser Ansatz ist jedoch mit einem großen Rechenaufwand für das Wiederfinden, der in der Datenbank hinterlegten Muster, mit dem zu analysierenden Bild verbunden [16]. DL getriebene CV Ansätze sind ebenfalls rechenintensiv, mit der Weiterentwicklung der Hardware ist die Rechenleistung von Maschinen jedoch ausreichend für

den Einsatz. Die traditionellen CV Werkzeuge sind besonders im Bezug auf Genauigkeit von mit DL kombinierten Verfahren überholt worden [15, 21] .

HPE lässt sich in Einzel-Personen (eng. single-person) und Multi-Personen (eng. multi-person) HPE unterscheiden. Einzel-Personen HPE verarbeiten eine Person pro Bild, während Multi-Personen HPE mehrere Personen pro Bild verarbeiten können. Des weiteren gibt es zwei Ansätze, Top-Down und Bottom-Up, die sich in der Reihenfolge der Verarbeitung von Mensch und SP unterscheiden. Top-Down-Ansätze ermitteln zuerst die einzelnen Personen in einem Bild und erst danach die einzelnen SP innerhalb eines Menschen. Dadurch können Einzelpersonen HPE durch einen Personen-Ermittler, der zuerst die einzelnen Personen aus einem Bild ausschneidet, leicht in Multi-Personen HPE überführt werden. Diese Ansätze bedingen jedoch eine hohe Fehleranfälligkeit, wenn sich Personen in Bildern oft überschneiden. Weiterhin ist die Laufzeit von Top-Down-Ansätzen proportional zur Anzahl von Personen innerhalb eines Bildes. Dem Gegenüber stehen deutlich robustere und performantere Bottom-Up-Ansätze. Für jeden Bildausschnitt wird berechnet, wie hoch die Wahrscheinlichkeit ist, dass sich an dieser Stelle ein bestimmter SP befindet. Anhand dieser Wahrscheinlichkeiten werden im Nachhinein Personen zugeordnet. [3].

2.3.1 HPE für Synchronität im Rudersport

HPE eignet sich für die Ermittlung von Synchronität in den Bewegungsabläufen von Sportlern, da sich anhand der SP viele Eigenschaften und Metriken ableiten lassen. Eine gängige Möglichkeit besteht daraus, die Abstände zwischen den gleichen SP der verschiedenen Personen zu ermitteln und dessen Abweichungen in ihrer Distanz als Asynchronität zu werten. Für eine Rudermannschaft hieße dies, den Abstand der Handgelenke, Schultern oder anderen SP miteinander zu vergleichen. Sofern die Abstände konstant sind, ist die Mannschaft synchron, bei abweichenden Abständen zeigt sich dementsprechende Asynchronität. Dadurch dass die Kamera im Motorboot platziert ist und dieses nicht immer den gleichen Winkel und die gleiche Distanz zum Ruderboot halten kann, ist dieser Ansatz problematisch. In Abbildung 2.3 ist eine synchron bleibende Bewegung zweier Rudernden zu sehen, da die Abstände zwischen den SP gleich bleiben. In Abbildung 2.4 bewegen sich die Rudernden auseinander, da sich die Abstände aus der Rücklage (j) im Vergleich zu Freilaufphase (k) verändern, obwohl die Entfernung zwischen den Füßen der Rudernden gleich bleibt.

Abbildung 2.3: Synchrone Bewegung bei Distanzmessung zwischen SP

Abbildung 2.4: Asynchrone Bewegung bei Distanzmessung zwischen SP

Weiterhin besteht die Möglichkeit mithilfe der SP die Winkel zwischen Körperteilen und somit die Gesamtposition, wie z.B. die Aus- oder Rücklage der Rudernden, zu bestimmen. Daraus kann ein Abgleich der Pose der Rudernden zu einem bestimmten Zeitpunkt erfolgen. Neben dem Bestimmen der kompletten Pose, kann zur Vereinfachung die Entfernung zwischen dem Handgelenk und den Füßen ermittelt werden. Daraus kann abgeleitet werden, wann die Rudernden sich in bestimmten Posen wie der Vor- und Rücklage befinden. In Abbildung 2.5 sind die Wendepunkte zu sehen, an denen die maximale positive und negative Entfernung zwischen Händen zu den Füßen eingezeichnet ist. Diese Zeitpunkte können gemessen und zwischen den Personen verglichen werden, sodass die Mannschaft sich bei gleichen Zeitpunkten synchron bewegt.

2.3.2 Mögliche Fehlerquellen der HPE

In ersten Versuchen hat sich gezeigt, dass das Reflektieren des Wassers dafür gesorgt hat, dass die HPE in der Reflexion Personen erkannt hat und die Messdaten somit verfälscht werden. Um diesem Problem entgegenzuwirken, sollten die finalen Versuche an Tagen stattfinden an denen leichter Wind und damit leichte Wellen auf dem Wasser sind, sodass

Abbildung 2.5: Maximale Bewegungsamplitude im Ruderschlag, hintere und vordere Umkehr

Personen nicht in den Spiegelungen zu erkennen sind. Als eine weitere Fehlerquelle stellte sich die Beschaffenheit des Ruderbootes heraus, welches einen Teil der Beine und Füße der Rudernden verdeckte.

2.4 Beschleunigungssensoren

Sensorik gehört zum Forschungsgebiet der Messtechnik und setzt sich mit der Entwicklung und Anwendung von Sensoren zur Messung von Veränderungen in technischen Systemen auseinander. Die Schlüsselfunktion von Sensoren besteht daraus, Informationen in elektrische Signale, sprich eine technische Messgröße, umzuwandeln [8]. Beschleunigungssensoren (BS) sind spezielle Sensoren für die Ermittlung von Beschleunigung in g oder in m/s^2. Beschleunigung ist eine mechanische, kinematische Größe, die mit Hilfe eines Feder-Masse-Systems, also der Streckung von Federn, die eine Masse halten, abgleitet werden kann [10]. Das Signal der BS kann positiv, negativ oder neutral sein und erfolgt in drei Raumachsen. BS lassen sich in ihrer Abtastrate einstellen, wodurch die Frequenz der Messung an die vorausgesetzten Anforderungen angepasst werden kann. BS eignen sich für die Messung von Synchronität im Sport, da die Beschleunigung der einzelnen Person gemessen und miteinander verglichen werden kann. BS sind ein bewährtes Mittel für Analysen im Sport und werden im Rudersport bereits für die Messung von Synchronität genutzt [5]. Wichtig ist, dass die Sensoren an den gleichen Stellen angebracht sind. Mithilfe des aufgezeichneten Beschleunigungsverlaufes eines gesamten Bootes können Störfaktoren innerhalb eines Ruderschlages identifiziert werden.

3 Design

3.1 OpenPose

OpenPose bietet eine 2D und 3D Posen Erkennung des gesamten menschlichen Körpers. Es ist das gängigste HPE Framework und kann bis zu 135 verschiedene SP bei Single-Person HPE erkennen [3]. Die hohe Anzahl der Single-Person HPE ergibt sich aus der Hand- und Gesichts-Pose Estimation, die in der Multi-Personen HPE jedoch nicht zur Verfügung steht. Für Multi-Personen HPE besteht die Auswahl zwischen 15, 18 und 25 SP pro Person, wobei Fotos und Videos als Eingabe genutzt werden können. Die Ausgabeoptionen sind Bilder und Videos mit eingezeichneten SP oder die textuelle Interpretation im Koordinatensystem in JSON, XML oder anderen Formaten.

OpenPose ist auf den gängigen Betriebssystemen (Windows, Mac OS, Linux) nutzbar und stellt neben GPU auch CPU lauffähige Versionen zur Verfügung. Die Laufzeit des Framework bleibt konstant, während sich die Laufzeit von anderen HPE-Frameworks wie AlphaPose und Mask R-CNN mit der Anzahl der Personen linear erhöht [2, 18, 4, 22]. Für das Training von OpenPose wurde der CMU Panoptic Studio Datensatz [19] genutzt, der von Panoptic Studio erstellt wurde, um soziale Interaktionen zu analysieren [11]. Für die Ermittlung von Synchronität zwischen mehreren Personen eignen sich besonders Multi-Personen HPE, da diese die verschiedenen Personen innerhalb eines Bildes ohne weiteren Aufwand verarbeiten kann. Bei der Nutzung einer Single-Personen HPE, müssten zunächst die einzelnen Personen in den Bildern herausgearbeitet werden, um dann die einzelnen Keypoints herauszulesen. Genutzt wird das Multi-Personen und 2D Model BODY_25 von OpenPose, welches 25 SP pro Person innerhalb eines Bildes ermittelt. Als Eingabeformat werden MP4-Videos genutzt, als Ausgabe JSON-Dateien und Videos mit eingezeichneten SP zu späteren Evaluation.

3.2 MetaMotionS

Für den Einsatz von BS ist es wichtig, dass die Sensoren die Bewegungsfreiheit der Rudernden nicht stören. Außerdem sollte es möglich sein, die Sensoren an den gewünschten Stellen zu befestigen und vom Motorboot oder aus dem Ruderboot zu bedienen. Von der HAW Hamburg wurden für diese Arbeit MetaMotionS Sensoren zur Verfügung gestellt, die die o.g. Ansprüche erfüllen. Der Sensor misst die Beschleunigung in drei Raumachsen in g, und stellt CSV-Dateien als Ausgabedateien der gemessenen Beschleunigung zur Verfügung.

3.3 Versuchsablauf

Für das Experiment wurde ein Doppelzweier, welcher von zwei Rudernden mit vier Rudern gefahren wird, für fünf bzw. zehn und dreißig Ruderschläge aufgezeichnet. Die Kamera und Sensoren befanden sich, wie vorausgehen beschrieben ausgerichtet und fixiert. Die Messung der Sensoren und die Videoaufzeichnung starteten vor dem eigentlichen Experiment. Als Start- und Endsignal wurde, um die Sensordaten untereinander, sowie die Bildaufzeichnungen mit den Sensordaten vergleichen zu können, ein Handschlag praktiziert. Dieser ist in Abbildung 3.1 zu sehen. Die Rudernden begannen mit dem Startsignal und begaben sich in die Rücklage. Auf Ansage des Bugmannes begannen sie gemeinsam zu Rudern. Nach einer vorher ausgemachten Anzahl von Schlägen beendeten beide Rudernden ihre Bewegung mit dem Endsignal, worauf hin die Aufzeichnungen gestoppt wurden. Wiederholt wurde der Versuch mit fünf, zehn und dreißig Schlägen.

3.3.1 Kamerapositionierung

Die Rudernden werden mit Hilfe einer Kamera aus dem begleitenden Motorboot gefilmt. Die Kamera befindet sich in etwa 1,5 Metern über den Köpfen der Rudernden und zwischen 2,5-10 Meter von ihnen entfernt. Dies soll die Sichtbarkeit der Körper und damit die Zuverlässigkeit der HPE erhöhen. Die Aufnahmen werden aus verschiedenen Winkeln gemacht, dafür fährt das Motorboot auf Höhe des Hecks, neben den Rudernden her. In ersten Versuchen hat sich herausgestellt, dass die seitlich leicht frontale Kamerapositionierung zu den besten Resultaten mit OpenPose führt. Ein Beispiel der gewählten Kamerapositionierung ist in Abbildung 3.2 zu sehen.

Abbildung 3.1: Start- und Endsignal Handschlag

Abbildung 3.2: Aufnahme der Rudernden leicht frontal

3.3.2 Sensorpositionierung

Um die Synchronität der rudernden Personen mittels Beschleunigungssensoren zu messen, gibt es mehrere Punkte an denen die Sensoren befestigt werden können. Der Rollsitz läuft auf einer Schiene und hat daher den Vorteil, dass die Sensoren wenigen Störfaktoren ausgesetzt sind. Um die Sensoren untereinander zu synchronisieren, ist es jedoch nötig ein Start- und Endsignal der Messung festzulegen. Des Weiteren lassen sich durch die Beschleunigung der Rollsitze nur Rückschlüsse auf die Bewegung der Beine schließen, sodass nicht die komplette Bewegung ausgewertet werden kann. Das Ruder beschreitet den vollständigen Bewegungsablauf. Es rotiert jedoch für die Auf- und Abdrehbewegung des Blattes, sodass diese Rotation die Analyse erschwert. Für diese Arbeit werden die Sensoren mithilfe eines Armbandes am Unterarm der rudernden Personen befestigt. Diese Befestigungsmöglichkeit sorgt dafür, dass die Synchronität des ganzheitlichen Bewegungsradius ähnlich wie bei den Rudern, erfasst werden kann und enthält keine großen Störfaktoren. Deshalb wird in dieser Arbeit die Synchronität der Rudernden mit Sensoren gemessen, die sich etwa 5cm oberhalb des Handgelenkes mit Hilfe eines eigenst für diese Arbeit entworfenen Armbands fixiert werden. Der Bewegungsradius, den der Sensor mit der gewählten Positionierung beschreitet ist in Abbildung 3.3 dargestellt.

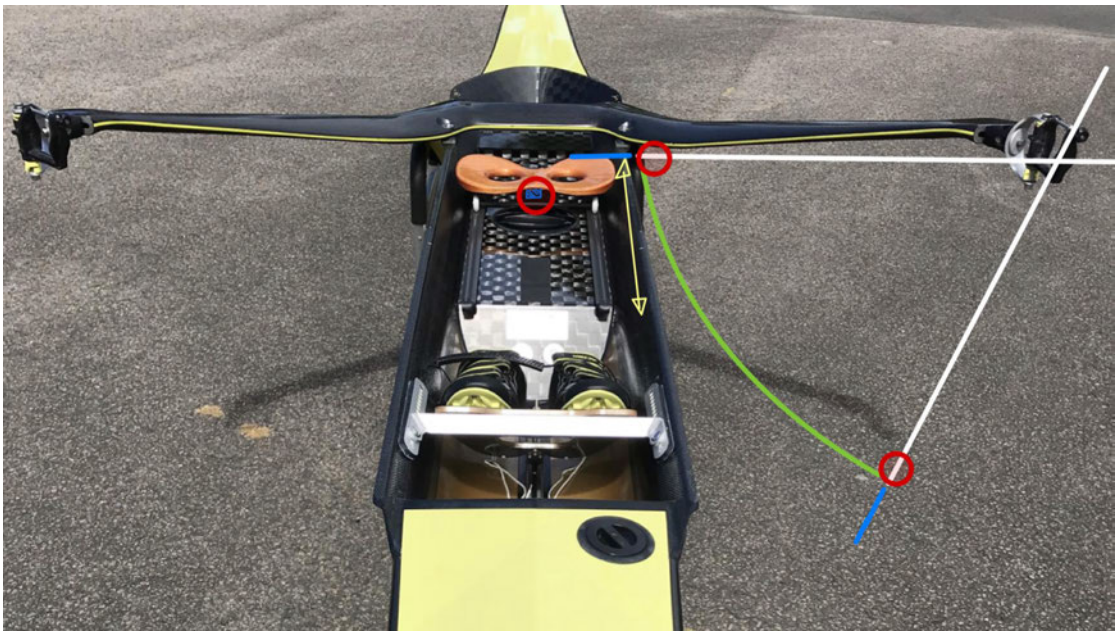

Abbildung 3.3: Winkeländerung des Ruders im Schlag

Abbildung 3.4: Platzierung des Sensors am Handgelenk

3.4 Datenverarbeitung der Pose Estimation

Die HPE wird mit Hilfe eines Skripts, welches in Google Colab läuft, auf das Video der Rudernden angewandt. In Abbildung 3.2 ist ein Beispielbild der von OpenPose errechneten und in das Video eingezeichneten SP zu sehen. Die errechneten Keypoints pro Bild werden von OpenPose jeweils in eine JSON-Datei geschrieben, wie sie in Listing 3.1 zu sehen ist. Dabei beschreibt die erste Gleitkommazahl die X-Koordinate, die zweite die Y-Koordinate und die dritte die Wahrscheinlichkeit, mit der der jeweilige SP auf diesen Koordinaten liegt. Der obere Eintrag spiegelt das Handgelenk, der untere Eintrag den Fuß der ersten Person im Boot wieder.

Listing 3.1: Beispiel JSON:Ausgabe OpenPose

```
{
    ...
    "pose_keypoints_2d":[
        ...
        977.674, 609.744, 0.789403,
        ...
        930.627, 671.477, 0.749273,
        ...
    ], ...
}
```

Abbildung 3.5: Beispiel: Anwendung OpenPose auf Video

Da die Daten für jedes einzelne Bild in einer eigenen JSON-Datei gespeichert wird, werden die Daten zunächst in einer großen CSV-Datei zusammengefasst. Für die eigentliche Verarbeitung der Daten, werden diese mit Hilfe von Python und Pandas geladen. Die Pose Estimation produziert viele SP die nicht gebraucht werden, deshalb werden nur für diesen Anwendungsfall relevante SP, also die X- und Y- Koordinaten der Handgelenke und der Füße als Pandas Dataframe extrahiert. Wenn die Daten für einen SP nicht korrekt ermittelt werden konnten, werden diese durch den Wert der vorherigen Koordinate ersetzt, sodass die Verzerrung der Daten reduziert wird. Auch wenn die Daten dadurch beeinflusst werden, ist die Abweichung vom realen Wert geringer als vorher. Für den ersten Lösungsansatz werden jeweils die X-Koordinaten der Füße von den X-Koordinaten der Handgelenke subtrahiert, um die entsprechende Relation bezüglich der Vergleichbarkeit zwischen den Rudernden zu erlangen. Für den zweiten Lösungsansatz wird die euklidische Distanz zwischen Füßen und Handgelenken errechnet, sodass nicht die Position auf der X-Achse, sondern die Distanz zwischen Handgelenk und Fuß berechnet wird. Die euklidische Distanz d wird mit der folgenden Formel

$$d = \sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2}$$

berechnet.

3.5 Ergebnisse der Pose Estimation

Die HPE erkennt die Rudernden und liefert Auswertungen zu den verschiedenen SP. Die Ergebnisse der HPE lassen sich besonders gut im Video mit den von OpenPose eingezeichneten errechneten SP darstellen. Eine korrekte Feststellung der SP ist in Abbildung fig. 3.6 zu sehen. Die SP werden optimal erkannt, auch die Positionierung der verdeckten Füße wird von der HPE korrekt berechnet.

Abbildung 3.6: Korrekt erfasste SP in einem Bild

Die HPE liefert nicht nur gute Ergebnisse, exemplarisch werden einige fehlerhafte Erkennungen beschrieben, die einen Großteil der Auswertungen ausmacht. Ein Fehler, der vermehrt in den Auswertungen auftritt, ist, dass die Füße nicht optimal berechnet werden, obwohl OpenPose grundsätzlich in der Lage ist, diese zu ermitteln. In Abbildung 3.7 ist zu sehen, dass die HPE die Füße der Rudernden im Wasser berechnet.

In Abbildung 3.8 ist zu sehen, dass die HPE oftmals Bootsmaterial als Körperteile der Rudernden erkennt und dadurch falsche SP errechnet.

In Abbildung 3.9 ist ein Weiterer, aber selten auftretender Fehler zu sehen, nämlich, dass Personen im Bild erkannt werden, wo sich überhaupt keine Person befindet.

Nicht festgestellte SP werden an der Position null, mit den X- und Y-Koordinaten null eingetragen, wie es in Abbildung 3.10 der Spalte mit der Beschriftung min zu sehen ist.

Abbildung 3.7: HPE erfasste SP mit Fehlerkennung der Füße im Wasser

Abbildung 3.8: HPE erfasste SP mit Fehlerkennung von Bootsmaterial als Körperteil

Abbildung 3.9: HPE erfasste SP mit Erkennung einer nicht existierenden Person

	wrist_x-coordinate	wrist_y-coordinate	wrist_percent	foot_x-coordinate	foot_y-coordinate	foot_percent	wrist_count
count	647.000000	647.000000	647.000000	647.000000	647.000000	647.000000	647.000000
mean	837.117689	557.962473	0.859524	870.319153	583.153607	0.744921	323.000000
std	132.856029	54.184224	0.088214	163.809412	99.052974	0.186237	186.917094
min	562.690000	439.128000	0.497506	0.000000	0.000000	0.000000	0.000000
25%	771.599500	524.402000	0.805814	789.239500	533.289000	0.713257	161.500000
50%	818.752000	544.990000	0.864662	904.194000	568.508000	0.804781	323.000000
75%	920.269000	597.921500	0.922762	954.117500	647.896500	0.848140	484.500000
max	1127.780000	683.364000	1.031450	1139.450000	739.220000	0.998927	646.000000

Abbildung 3.10: Übersicht der erfassten SP

3.6 Verarbeitung der BS-Daten

Im Folgenden soll besprochen werden, wie die Datenverarbeitung stattfand, welche Probleme dabei angetroffen wurden und wie diese gelöst wurden.

Die MetaMotionS Sensoren stellen die Beschleunigungsdaten in drei Achsen jeweils in einer CSV-Datei zur Verfügung. Diese Daten werden mit Hilfe von Pandas [20], einem Datenanalyse Werkzeug für Python, aus den CSV-Dateien ausgelesen und in einen Pandas Dataframe umgewandelt. Weiter extrahiert werden die Daten der x-Achse, die die Beschleunigung in der horizontalen liefern. Die Beschleunigungsdaten werden von den Sensoren in g zur Verfügung gestellt. Nach dem Extrahieren werden diese jedoch mit dem Faktor 9,81 multipliziert, sodass die Beschleunigungsdaten in m/s^2 weiterverarbeitet werden.

Wenn die BS-Daten nun in einem Liniendiagramm dargestellt werden, fällt auf, dass die Sensoren nicht zum gleichen Zeitpunkt sondern versetzt voneinander mit dem Aufzeichnen der Daten beginnen, dies ist in Abbildung 3.11 zu sehen. Das oben beschriebene Verhalten tritt auf, obwohl die Sensoren über das gleiche externe Gerät, in diesem Fall ein Smartphone mit der zu den Sensoren gehörenden App MetaBase gestartet wurden.

Um die zeitlich verschobenen Datensätze korrigieren zu können, wird mithilfe der Kreuzkorrelation die lineare Abhängigkeit zwischen den Messreihen festgestellt [23]. Durch die Kreuzkorrelation wird derjenige Wert ermittelt, um den die Datenreihen verschoben werden müssen, damit die Daten optimal übereinander gelegt und weiterverarbeitet werden können.

Da die Startsignale eine sehr hohe Übereinstimmung produzieren, liegen diese nun übereinander. Um die Ergebnisse des Ruderschlages mit angestrebter Synchronität besser Evaluieren zu können, wurde ein Datensatz erstellt, für den absichtlich asynchron gerudert wurde. Dieser wurde mit Hilfe der von den Sensoren erstellten Zeitstempel verschoben, da die Kreuzkorrelation für die Bestimmung der linearen Abhängigkeit in diesem Fall nicht funktionierte.

Es wird der erste Zeitstempel der asynchronen Datenreihe genommen, die als letztes mit der Aufzeichnung beginnt. Die Daten der anderen Datenreihe, die vor diesem Zeitstempel liegen, werden gelöscht, sodass die Startzeitpunkte ebenfalls für den asynchronen Datensatz zeitlich angeglichen sind. Das gleiche Verfahren könnte für den synchronen Datensatz genutzt werden. Der asynchrone Datensatz wurde jedoch im Nachhinein hinzugezogen,

sodass die Verarbeitung der synchronen Daten schon abgeschlossen war. Es besteht weiterhin das Problem des Clock drifts, welches bedeutet, dass die realen Abtastraten nicht übereinstimmen, obwohl die Sensoren mit der gleichen Abtastrate aufzeichnen. Dies liegt daran, dass die Sensoren jeweils über eine eigene Notation der Zeit verfügen und die Uhren daher nicht synchron arbeiten.

Um die Abtastraten im Nachhinein wieder zu korrigieren und die Datensätze aneinander anzugleichen erfolgt auf einem der beiden Datensätze eine Streckung oder ein Stauchen. Für das Anpassen der Abtastraten im Nachhinein wird geprüft, in welchem Intervall Werte aus der längeren Datenreihe gelöscht werden müssen, damit die Abtastrate korrigiert wird.

In Folge der Verschiebung mithilfe der Kreuzkorrelation, oder der manuellen Anpassung und der anschließenden Abtastratenanpassung liegen die Datenreihen optimal übereinander, wie es in Abbildung 3.12 für das synchrone und in Abbildung 3.13 für das asynchrone Rudern zu sehen ist. Die vorverarbeiteten Daten lassen sich analysieren. Eine Wertigkeit für die Synchronität wird nun durch die Korrelationskoeffizienten berechnet, diese ermittelt einen Wert zwischen 1 und -1 für die Übereinstimmung der Signale. Die Korrelationskoeffizienten werden zunächst über fünf synchrone und asynchrone aufeinander folgende Ruderschläge berechnet, dann für die jeweils einzelnen Schläge errechnet. So kann man die Synchronität zunächst für mehrere Schläge bestimmen und im Nachhinein auf die Besonderheiten einzelner Schläge untersuchen.

Für die weitere Untersuchung der einzelnen Schläge, werden die Kurven mit Hilfe einer Extrempunkt- und Nullstellenanalyse untersucht, sodass Abweichungen zwischen den Bewegungen der Rudernden ermittelt werden. Dies hat zur Folge, dass die Trainierenden auf Grundlage dieser Ergebnisse, Änderungen zur Weiterentwicklung der kollektiven Technik vornehmen können.

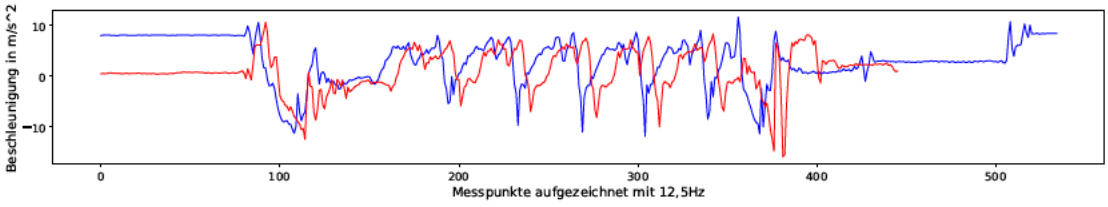

Abbildung 3.11: Rohdatensätze synchrones Rudern

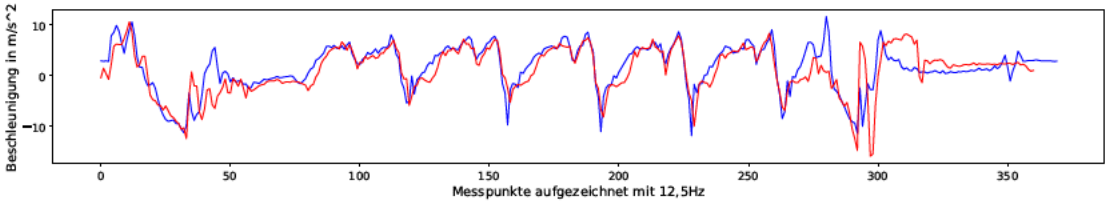

Abbildung 3.12: Korrigierte Datensätze synchrones Rudern

Abbildung 3.13: Korrigierte Datensätze asynchrones Rudern

3.7 Ergebnisse der Beschleunigungssensoren

Die Korrelationskoeffizienten sind in der Tabelle 3.1 und der Tabelle 3.2 dargestellt. Für die synchron angestrebten Aufzeichnungen wird über fünf Schläge eine Übereinstimmung von 0,882 ermittelt, während für die asynchron angestrebten Aufzeichnungen eine Übereinstimmung von 0,297 ermittelt wird, womit sich eine Differenz von 0,585 ergibt. Dies zeigt, dass mit Hilfe von BS die Synchronität von Rudermannschaften gemessen und verglichen werden kann.

Die Korrelationskoeffizienten der einzelnen Schläge liegen für synchron angestrebten Aufzeichnung zwischen 0,804 und 0,933, für die asynchron angestrebten Aufzeichnung zwischen -0,015 und 0,656. Der jeweilige Median liegt bei dem zweiten Schlag für die synchronen Schläge bei 0,864 und im fünften Schlag für die asynchronen Schläge bei 0,319. Dies ergibt zwischen den Medianen eine Differenz von 0,545.

Ruderschlag 1-5	0,882
Ruderschlag 1	0,804
Ruderschlag 2	0,864
Ruderschlag 3	0,933
Ruderschlag 4	0,886
Ruderschlag 5	0,889

Tabelle 3.1: Korrelationskoeffizienten synchron angestrebtes Rudern

Ruderschlag 1-5	0,297
Ruderschlag 1	0,657
Ruderschlag 2	0,015
Ruderschlag 3	-0,015
Ruderschlag 4	0,389
Ruderschlag 5	0,320

Tabelle 3.2: Korrelationskoeffizienten asynchron angestrebtes Rudern

Die fünf Ruderschläge sind für das synchron angestrebte Rudern sind in Abbildung 3.14 und in Abbildung 3.15 zu sehen.

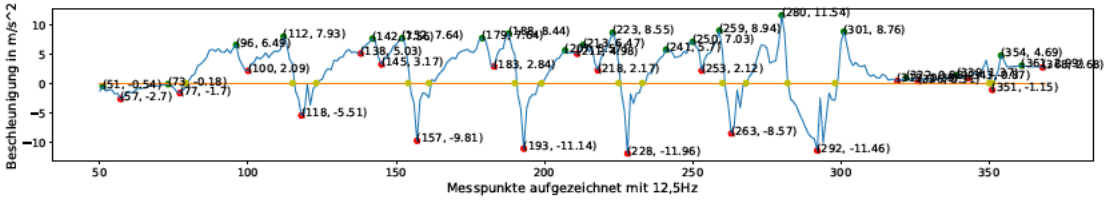

Abbildung 3.14: Errechnete Peaks der Person 1 synchron angestrebtes Rudern

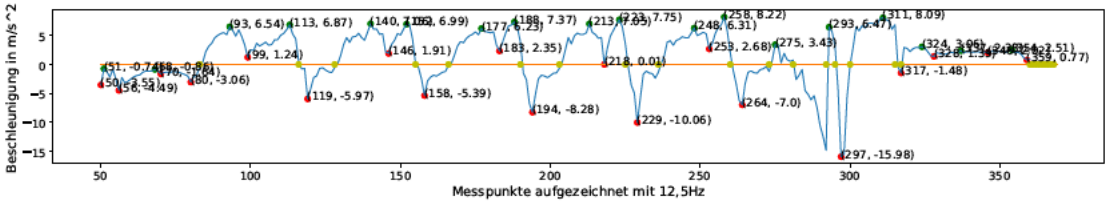

Abbildung 3.15: Errechnete Peaks der Person 2 synchron angestrebtes Rudern

Die Kurven zeigen ein sich wiederholendes Muster, aus welchem sich die Position der Rudernden zu einem Zeitpunkt ableiten lässt. Der zweite Schlag der ersten Person in Abbildung 3.14 startet beim Messpunkt (MP) 157 mit der hinteren Umkehr und leitet den Freilauf ein. Die vordere Umkehr wird bei MP 179 erreicht und leitet den Durchzug ein. Der nächste Hochpunkt bei MP 188 zeigt den Mittelzug. Der Durchzug endet in der hinteren Umkehr bei MP 193, in welcher ein neuer Ruderschlag beginnt. Die Tabelle 3.3 listet die ermittelten Hoch- und Tiefpunkte der Graphen aus den Abbildungen 3.14 und 3.15 zwischen MP 150 und MP 350 auf.

Die Maxima und Minima zwischen den beiden Personen liegen meist auf nahezu den gleichen Werten auf der X-Achse, jedoch werden unterschiedlich viele Hoch- und Tiefpunkte erfasst. Für eine bessere Übersicht werden die Werte in der Tabelle 3.4 mit Platzhaltern

Min P1	157	183	193	211	218	228	253	263	292	319	326	343
Min P2	158	183	194	218	229	253	264	297	317	328	346	x
Max P1	152	179	188	207	213	223	241	250	259	280	301	322
Max P2	152	177	188	213	223	248	258	275	293	311	324	337
Nst P1	154	161	190	199	225	233	260	268	282	298	x	x
Nst P2	155	166	190	203	225	236	260	273	281	292	295	300

Tabelle 3.3: Beispiel Peaks synchron angestrebtes Rudern

Min P1	157	183	193	211	218	228	253	263	292	319	326	343
Min P2	158	183	194	x	218	229	253	264	297	317	328	346
Max P1	152	179	188	207	213	223	241	250	259	280	301	x
Max P2	152	177	188	x	213	223	x	248	258	275	293	311
Nst P1	154	161	190	199	225	233	260	268	282	x	x	298
Nst P2	155	166	190	203	225	236	260	273	281	292	295	300

Tabelle 3.4: Beispiel Peaks synchron angestrebtes Rudern: Korrigiert

dargestellt. Der zu Beginn festgelegte Versatz von 100 Millisekunden, der für eine synchron rudernde Mannschaft eingehalten werden muss, wird hier durch etwa einen MP repräsentiert.

Es wird sichtbar, dass die Mannschaft bei neun von elf Messungen innerhalb der tolerierten Abweichungen als synchron gilt. In den ersten beiden Minima gibt es jeweils eine Abweichung von 200 Millisekunden. In den Maxima halten die Messungen sechs von elf mal den definierten Schwellwert ein und weisen somit fünf Messpunkte ohne Synchronität auf.

Die Extrempunkte der Ruderschläge lassen sich nun vergleichen. Der zweite Ruderschlag verläuft für den Schlagmann von MP 157 bis MP 193. Die Kurvenverläufe der beiden Personen sind innerhalb der Abbildung 3.16 dargestellt.

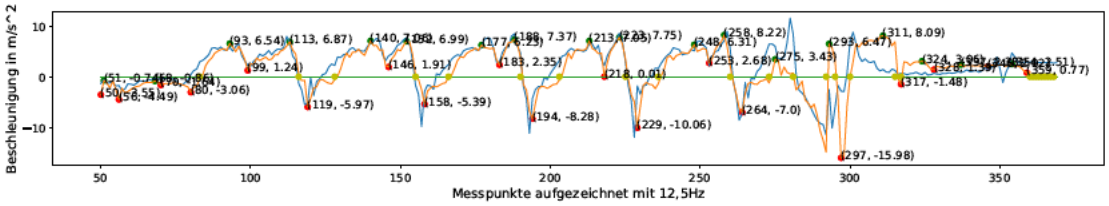

Abbildung 3.16: Kurven und errechnete Extrempunkte für synchron angestrebtes Rudern

Die hintere Umkehr der Person 1 (Schlagmann) liegt bei MP 157, die der Person 2 (Bugmann) bei MP 158, dort sind die Rudernden also um einen Messpunkt versetzt, ist. Das zweite Minimum zu Beginn des Durchzuges, liegt jeweils bei MP 183 und die nächste hintere Umkehr, also das Beenden des Ruderschlages liegt bei MP 193 für den Schlagmann und für den Bugmann bei 194. Die vordere Umkehr des Schlagmannes liegt bei MP 179, für den Bugmann bei 177, diese gilt also nicht als Synchron. Der Mittelzug ist wiederum synchron und liegt beim Bug- und Schlagmann bei MP 188. Die erste Nullstelle liegt beim Schlagmann bei 166, während die des Bugmannes bei 161 liegt. Die zweite Nullstelle liegt jeweils bei 190. Die Abweichung von fünf MP für die erste Nullstelle, zeigt dass sich die Rudernden sich zu Beginn der Freilaufphase nicht synchron bewegen. In Abbildung 3.17 ist zu sehen, dass die Beine des Bugmannes noch in der Streckung befinden, während die Beine des Schlagmannes bereits gebeugt sind, der Bugmann bewegt sich folglich verzögert. Durch die Abweichung der Extrem- und Nullstellen der beiden Ruderschläge

Abbildung 3.17: Versatz beim Beine beugen

wird deutlich, dass die Mannschaft in den meisten Punkten synchron rudert. Lediglich in der vorderen Umkehr, also dem Eintauchen der Ruder und im Beginn der Freilaufphase überschreiten sie den Grenzwert für Synchronität. Der Versatz beim Eintauchen der Blätter ist ebenfalls in Abbildung 3.18 zu erkennen. Die Blätter des Bugmannes sind

Abbildung 3.18: Versatz beim Eintauchen der Blätter

bereits leicht eingetaucht, während die Blätter des Schlagmannes noch vollständig zu sehen sind. Es zeigt sich dass der synchrone Schlag durchaus in den Sensordaten zu erkennen ist. Abweichungen der Mannschaft, da diese noch nicht optimal auf synchrones Rudern trainiert ist, lassen sich ebenfalls aus den gewonnenen Ergebnissen ableiten. So wird deutlich dass der Bugmann sich in der ersten Phase des Freilaufs schneller, danach jedoch langsamer bewegen sollte, um sich an den Schlagmann anzugleichen.

Ein Ausschnitt der Extrempunkte und Nullstellen der fünf Schläge des asynchron angestrebten Ruderns sind in sind in der Tabelle 3.5 zu sehen. Das asynchron angestrebte Rudern ist in Abbildung 3.19 und in Abbildung 3.20 grafisch dargestellt.

Min P1	x	79	96	105	116	144	156	170	176	183	x
Min P2	74	83	x	x	120	144	158	173	x	184	x
Max P1	73	x	89	99	111	128	142	150	168	181	x
Max P2	71	78	86	101	114	132	141	153	171	177	x
Nst P1	66	67	74	82	111	x	121	153	160	168	170
Nst P2	xx	80	91	116	127	155	164	195	x	x	x

Tabelle 3.5: Beispiel Peaks asynchron angestrebtes Rudern: Korrigiert

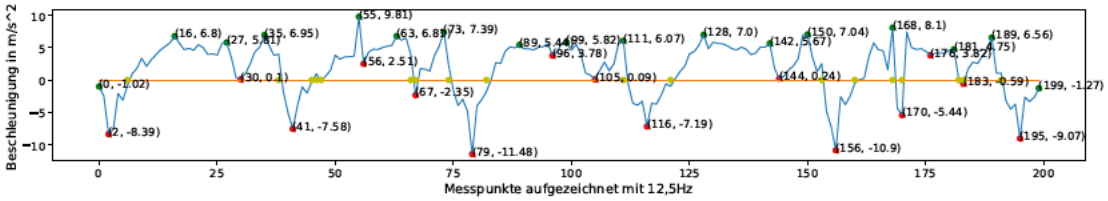

Abbildung 3.19: Errechnete Peaks der Person 1 asynchron angestrebtes Rudern

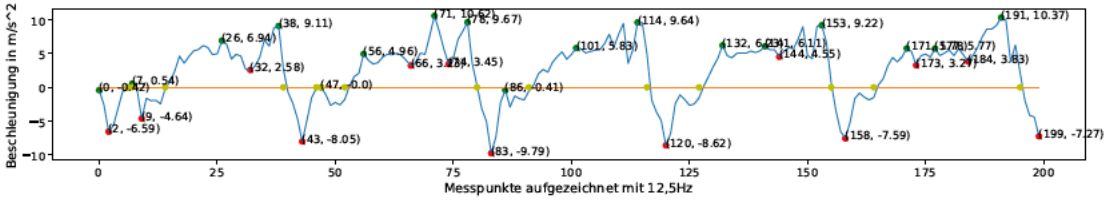

Abbildung 3.20: Errechnete Peaks der Person 2 asynchron angestrebtes Rudern

Für eine genauere Betrachtung der asynchron angestrebten Ruderschläge wird der vierte Schlag analysiert. Dieser beginnt für den Schlagmann bei MP 120 und endet bei MP 158.

Die Kurvenverläufe der beiden Personen übereinander gelegt sind innerhalb der Abbildung 3.21 dargestellt. Die hintere Umkehr des Schlagmannes liegt bei MP 116, die des

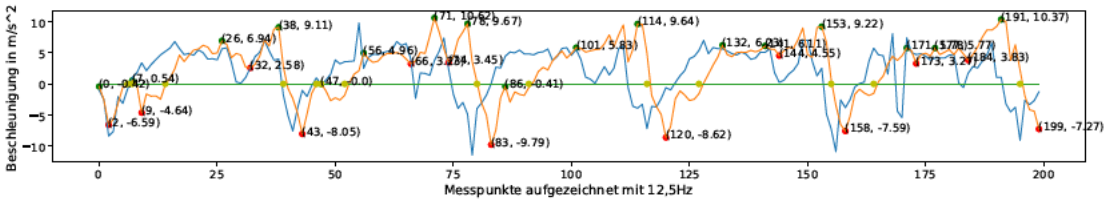

Abbildung 3.21: Kurven und Extrempunkte für asynchron angestrebtes Rudern

Bugmannes bei MP 120, sodass die Rudernden sich um vier Messpunkte versetzt bewegen. Das zweite Minimum zu Beginn des Durchzuges liegt jeweils bei MP 144. Das Beenden des Ruderschlages liegt bei MP 156 für den Schlagmann und für den Bugmann bei 158. Dies verdeutlicht eine verzögerte hintere Umkehr des Bugmannes. Zu sehen ist dieser Versatz ebenfalls im dazugehörigen Bild in Abbildung 3.22.

Abbildung 3.22: Verzögerte hintere Umkehr des Bugmannes

Die vordere Umkehr des Schlagmannes liegt bei MP 128, die des Bugmannes bei 132. Der Versatz der vorderen Umkehr ist im zugehörige Bild in Abbildung 3.23 dargestellt. Der Mittelzug liegt beim MP 150 für den Schlagmann und bei MP 153 für den Bugmann und ist daher ebenfalls versetzt.

Die erste Nullstelle liegt bei MP 121 für den Schlagmann, während die des Bugmannes bei MP 127 liegt. Aus der ersten Nullstelle lässt sich ableiten, dass der Bugmann sich im ersten Teil der Freilaufphase verzögert bewegt. Die Abweichung lässt sich ebenfalls im zugehörigen Bild erkennen und ist in Abbildung 3.24 zu sehen. Die zweite Nullstelle liegt

bei MP 153 für den Schlagmann, während die des Bugmannes bei 155 liegt und resultiert einen versetzten Durchzug.

Abbildung 3.23: Verzögerte vordere Umkehr des Bugmannes

Abbildung 3.24: Verzögerte Vorbereitung des Bugmannes

Bei Betrachtung der Kurven in Abbildung 3.21 sieht man, dass diese weit auseinander gehen, die Extrempunkte jedoch im Verhältnis dazu nah beieinander liegen. Grund dafür ist, dass die Rudernden sich auf einem sehr hohen ruderischen Niveau befinden und darauf konditioniert sind, sich synchron zu bewegen, und sich daher unbewusst an den prägnanten Punkten im Schlag wieder angleichen. Dennoch lässt sich auch hier aus den Extrempunkten und Nullstellen klar erkennen, ob eine Bewegungen synchron oder asynchron ist.

4 Evaluation

4.1 Evaluation der Pose Estimation

Die Ergebnisse der Pose Estimation haben gezeigt, dass es grundsätzlich möglich ist, die Posen und die Synchronität der Rudernden zu messen, zu untersuchen und weiterzuentwickeln. Die HPE liefert jedoch für eine aussagekräftige Untersuchung zu viele Fehlinterpretationen der SP der Rudernden. Eine Auswertung einfach aufgenommener Videos ist also nicht möglich. Jedoch gibt es Möglichkeiten die Fehlerquellen für die HPE zu reduzieren. Eine Optimierung ist es, die Spiegelung des Wassers zu entfernen, indem die Bilder manuell oder automatisiert an der Wasserlinie des Bootes angeschnitten werden, sodass die Spiegelung auf dem Wasser nicht im Bild vorhanden ist. Weitere Fehlerquellen sind, dass Teile der Rudernden von der Bordwand des Bootes verdeckt werden und dass der Kamerawinkel aus dem Motorboot aufgenommen dazu neigt sich zu ändern. Diese Fehlerquellen können reduziert werden, indem die Kamera mit Hilfe eines Stativs an das Boot angebracht wird, oder die Aufnahmen mit einer Drohne gemacht werden. Die Kamera kann so positioniert werden, dass die gesamten Menschen im Ruderboot zu sehen sind und die Spiegelung des Wassers von Anfang an nicht mit aufgezeichnet wird.

Des Weiteren sollte der Versuchsablauf auf dem Ruderergometer ausgeführt werden. So wären die Störfaktoren deutlich reduziert, da es weder Wasser, noch Bordwände gäbe, welche die Hauptfehlerquellen des Versuchs auf dem Wasser sind. Für einen Versuch auf dem Ruderergometer kann auch ohne Stativ leicht aus einem gleichbleibenden Winkel und gleichbleibender Entfernung gefilmt werden. Es sollte darauf geachtet werden einen schlichten Hintergrund, z.B. eine weiße Wand zu wählen. Die Zuverlässigkeit der HPE für Ermittlung der Synchronität zwischen Rudernden sollte auf dem Ruderergometer jedoch vorher untersucht werden und ist nicht Teil dieser Arbeit.

4.2 Evaluation der Beschleunigungsdaten

Die Ergebnisse der Beschleunigungsdaten zeigen eine objektive Bewertung der Synchronität in der Bewegung von Rudernden mithilfe der Korrelationskoeffizienten. Diese liefern einen Wert zwischen 1 und -1 für die Synchronität. Des Weiteren liefern die Extrempunkte, sowie die Nullstellen Aufschluss über die Abweichungen in prägnanten Positionen in der Bewegungsabfolge des Ruderschlages.

Die Verarbeitung kann vereinfacht und weiter automatisiert werden, indem ein präziseres Start- und Endsignal gewählt wird, welches eindeutiger als ein Klatschen in den Aufzeichnung zu erkennen ist. Hierfür sollte eine eindeutigere Bewegung in der horizontalen gewählt werden. So könnten beispielsweise die Rudernden gemeinsam eine Stange oder eine Schnur in der horizontalen Bewegen.

Weiterhin zeigt sich, dass die Rudernden im Versuch, auch wenn sie auf einem sehr hohen Niveau rudern, eine hohe Synchronität erreichen, obwohl zuvor kein großer Trainingsaufwand für in die kollektive Technik stattfand. Dies spricht dafür, dass für die Aufzeichnung mit eingefahrenen Mannschaften auf hohem Niveau, sprich Mannschaften in Vorbereitungen auf bedeutsame Wettkämpfe wie beispielsweise die Deutschen Meisterschaften, Weltmeisterschaften oder Olympia, eine höhere Abtastrate für die Sensoren gewählt werden sollte.

Die Vorverarbeitung für die Daten der Beschleunigungssensoren bringt dadurch, dass die Sensoren eigenständig funktionieren, einen großen Aufwand mit sich. Zur Vereinfachung kann ein einzelnes Data Acquisation System genutzt werden, an welches Beschleunigungssensoren angeschlossen werden. Dies hätte zur Folge, dass die Sensoren eine gemeinsame Uhr, also eine gemeinsame Notation der Zeit hätten. Hierfür müsste jedoch darauf geachtet werden, dass durch die eventuelle Verkabelung des Bootsmaterials die Mannschaft in ihrer Bewegung nicht eingeschränkt werden sollte um die Messergebnisse nicht zu beeinflussen. Durch das Einsparen der Vorverarbeitung der Daten, wird das sogenannte in time processing möglich. Die Ergebnisse würden bereits während des Ruderns übermittelt und das System könnte dementsprechend als direktes Feedback fungieren.

5 Fazit

Die HPE funktioniert nicht wie gewünscht, da die Zuverlässigkeit noch zu gering ist. HPE Frameworks werden sich wahrscheinlich in ihrer Zuverlässigkeit verbessern, sodass es in Zukunft möglich sein wird diese im Rudersport als Werkzeug zur Unterstützung des Traineralltages zu nutzen.

Das Messboot-Verfahren bietet eine komplexe Analyse der Individualtechnik der Rudernden, mit der sich diese Arbeit, wie in der Einleitung beschrieben, nicht beschäftigt hat. Das aufgrund der Komplexität begrenzte Angebot der Auswertungen mit dem Messboot-Verfahren liegt weit unter der Nachfrage nach Auswertungen von Ruderbewegungen. Das System, dass in dieser Arbeit entwickelt wurde, kann besonders im semiprofessionellen Bereich und ebenfalls im professionellen Bereich als schlanke Alternative zum Messboot-Verfahren für den Traineralltag genutzt werden.

Die Daten der Beschleunigungssensoren liefern eine objektive Bewertungsmöglichkeit für die Synchronität von Rudermannschaften. Die asynchronen Bewegungen der Rudernden werden mit Hilfe des Systems aufgezeigt. Nach der Auswertung sind die aufgezeigten asynchronen Bewegungen in einzelnen Bildern der zugehörigen Videos erkennbar. Erfahrene Trainer haben ein taktisches Vorgehen, um diese Punkte in Videoanalysen manuell durch pausieren der Videos herauszufinden. Das System kann beim Verarbeiten der Sensordaten nach den Trainingseinheiten besonders unerfahreneren Trainern großen Nutzen liefern. Außerdem wird für die Analyse ein technisches Grundwissen benötigt, welches die meisten Trainer jedoch besitzen. Einen großen Mehrwert bietet das in dieser Arbeit genutzte Verfahren dann, wenn es so angepasst wird, dass mit geringem zeitlichen Verzugs, also von Sekunden oder weniger, als Feedback während des Ruderns genutzt werden kann.

Eine Anpassung des Ruderschlages mit Hilfe der gewonnen Daten kann entscheidend für den Erfolg einer Mannschaft sein. Besonders im Leistungssport werden Ruderregatten oftmals mit hundertstel Sekunden und wenigen Zentimetern Vorsprung gewonnen. Auch

wenn die Anpassung des Ruderschlages zu einer synchronen Mannschaft nur Gewinne von wenigen Zentimetern pro Schlag erwirtschaftet, kann dies dazu führen, dass Mannschaften mit Einsatz des Systems über ein 2.000 Meter langes Rennen, welches aus über 200 Ruderschlägen besteht, einige Sekunden vor den anderen Mannschaften über die Ziellinie bewegt, das System somit von großem Vorteil sein kann.

Das Verarbeitung der BS-Daten kann modifiziert für andere rhythmische Mannschaftssportarten wie Kanu oder Kajak fahren, aber auch für das Tanzen oder das Synchronschwimmen genutzt werden. Für nicht Wassersportarten bietet die HPE bereits in ihrer aktuellen Form große Möglichkeiten der Analyse, sodass in dieser Arbeit entwickelte Verfahren für das Feststellen von Synchronität genutzt werden können.

5.1 Zusammenfassung

Das Ziel der Arbeit war es, am Beispiel des Rudersports, die Synchronität der Bewegung zwischen Rudernden zu ermitteln und somit objektive Bewegungsanalysen erstellen zu können.

Um dies zu erreichen wurde die Ruderbewegung zunächst in die hintere Umkehr, die Freilaufphase, die vordere Umkehr und den Durchzug aufgefächert und erläutert. Die Wichtigkeit von Synchronität für den Rudersport wurde durch Literatur belegt. Zudem wurde erklärt, wie diese gemessen werden kann. Als Werkzeuge zur Ermittlung der Synchronität eignen sich die Auswertungen der HPE und der Beschleunigungssensoren. Um Daten zu generieren wurden die Rudernden in mehreren Messfahrten mit Beschleunigungssensoren ausgestattet und gefilmt.

In der Auswertung der HPE gelang, aufgrund der geringen Genauigkeit, nur vereinzelt die korrekte Pose der Rudernden abzuleiten. Somit konnten keine quantifizierbaren Ergebnisse erbracht werden. Dennoch konnte aufgezeigt werden, unter welchen Modifikationen des Versuchablaufs, Fehlerquellen reduziert werden können, um die Genauigkeit der HPE zu steigern.

Die in den Messfahrten erhobenen Beschleunigungsdaten wurden durch Kreuzkorrelation und weitere Verfahren aufbereitet. Durch die Errechnung der Korrelationskoeffizienten für die, in den Messfahrten aufgezeichneten Ruderschläge, ergaben sich konkrete für die Übereinstimmung der Ruderbewegungen. Daraus ließen sich synchrone von asynchronen

Ruderschlägen klar unterscheiden. Die ermittelten Differenzen in den Korrelationskoeffizienten konnten durch Stichprobenartige Untersuchungen der Aufnahmen verifiziert werden.

Das Ziel der objektiven Synchronitätsermittlung von Rudernden, wurde durch die Verarbeitung von Beschleunigungsdaten erreicht. Somit ist der einleitend geforderte Maßstab für die Ermittlung von Synchronität im Rudersport konzipiert.

5.2 Ausblick

Zukünftige Forschungsarbeiten können diese Arbeit als Startpunkt für die Entwicklung von Systemen zur Ermittlung der Synchronität in Mannschaftssportarten nutzen. Besonders die Entwicklung eines HPE Frameworks für das Rudern, oder allgemein für Wassersportarten kann den Einsatz der HPE in diesem Bereichen vorantreiben und die Analysemöglichkeiten von Bewegungen im Sport verbessern. Zusätzlich kann in zukünftigen Forschungsarbeiten ein leichtgewichtiges Verfahren entwickelt werden, um die hier entwickelte Analyse der kollektiven auf die individuelle Rudertechnik auszuweiten.

Literaturverzeichnis

[1] Boucher, Jean-Philippe ; Labbé, Romain ; Clanet, Christophe: Row bots. In: *Physics today* 70 (2017), S. 82

[2] Cao, Z. ; Hidalgo Martinez, G. ; Simon, T. ; Wei, S. ; Sheikh, Y. A.: OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019)

[3] Cao, Zhe ; Simon, Tomas ; Wei, Shih-En ; Sheikh, Yaser: Realtime multi-person 2d pose estimation using part affinity fields. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 7291–7299

[4] Cao, Zhe ; Simon, Tomas ; Wei, Shih-En ; Sheikh, Yaser: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In: *CVPR*, 2017

[5] Cesarini, Daniel ; Lelli, Giovanni ; Avvenuti, Marco: Are we synchronized? Measure synchrony in team sports using a network of wireless accelerometers. In: *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing*, 2014, S. 1–6

[6] Doeksen, Jelte: Synchronization in Rowing: Thesis on the effect of crew synchronization on rowing performance. (2018)

[7] Groos, Daniel ; Ramampiaro, Heri ; Ihlen, Espen A.: EfficientPose: Scalable single-person pose estimation. In: *Applied intelligence* 51 (2021), S. 2518–2533

[8] Heinrich, Berthold ; Linke, Petra ; Glöckler, Michael: Grundlagen zur Automatisierung. In: *Grundlagen Automatisierung*. Springer, 2020, S. 1–29

[9] Hou, Yuxin ; Yao, Hongxun ; Li, Haoran ; Sun, Xiaoshuai: Dancing like a superstar: Action guidance based on pose estimation and conditional pose alignment. In: *2017 IEEE International Conference on Image Processing (ICIP)* IEEE (Veranst.), 2017, S. 1312–1316

[10] Isermann, Rolf: *Mechatronische systeme: grundlagen*. Springer-Verlag, 2007

[11] Joo, Hanbyul ; Simon, Tomas ; Li, Xulong ; Liu, Hao ; Tan, Lei ; Gui, Lin ; Banerjee, Sean ; Godisart, Timothy S. ; Nabbe, Bart ; Matthews, Iain ; Kanade, Takeo ; Nobuhara, Shohei ; Sheikh, Yaser: Panoptic Studio: A Massively Multiview System for Social Interaction Capture. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017)

[12] Minaee, Shervin ; Minaei, Mehdi ; Abdolrashidi, Amirali: Deep-emotion: Facial expression recognition using attentional convolutional network. In: *Sensors* 21 (2021), Nr. 9, S. 3046

[13] Odemakinde, Elisha: *Human Pose Estimation with Deep Learning Ultimate Overview in 2022*. https://viso.ai/deep-learning/pose-estimation-ultimate-overview/. – Accessed: 2022-04-30

[14] Ongsulee, Pariwat: Artificial intelligence, machine learning and deep learning. In: *2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE)* IEEE (Veranst.), 2017, S. 1–6

[15] O'Mahony, Niall ; Campbell, Sean ; Carvalho, Anderson ; Harapanahalli, Suman ; Hernandez, Gustavo V. ; Krpalkova, Lenka ; Riordan, Daniel ; Walsh, Joseph: Deep learning vs. traditional computer vision. In: *Science and information conference* Springer (Veranst.), 2019, S. 128–144

[16] Pouly, Marc ; Koller, Thomas ; Gottfrois, Philippe ; Lionetti, Simone: Künstliche Intelligenz in der Bildanalyse–Grundlagen und neue Entwicklungen. In: *Der Hautarzt* 71 (2020), Nr. 9, S. 660–668

[17] Schwarzrock, M ; Mattes, K ; Grahn, M u. a.: Trainingsmethodische Grundkonzeption 2017–2020. In: *Hannover, Germany: Deutscher Ruderverband* (2017)

[18] Simon, Tomas ; Joo, Hanbyul ; Matthews, Iain ; Sheikh, Yaser: Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In: *CVPR*, 2017

[19] Studio, CMU P.: *CMU Panoptic Dataset SENSORS & DATA ACQUISITION*. http://domedb.perception.cs.cmu.edu/index.html. – Accessed: 2022-05-11

[20] team, The pandas development: *pandas-dev/pandas: Pandas*. Februar 2020. – URL https://doi.org/10.5281/zenodo.3509134

[21] Voulodimos, Athanasios ; Doulamis, Nikolaos ; Doulamis, Anastasios ; Protopapadakis, Eftychios: Deep learning for computer vision: A brief review. In: *Computational intelligence and neuroscience* 2018 (2018)

[22] Wei, Shih-En ; Ramakrishna, Varun ; Kanade, Takeo ; Sheikh, Yaser: Convolutional pose machines. In: *CVPR*, 2016

[23] Werner, Martin: *Digitale Signalverarbeitung mit MATLAB*. Springer, 2012

[24] Zhang, Zhiqin: Deep Face Emotion Recognition. In: *Journal of Physics: Conference Series* Bd. 1087 IOP Publishing (Veranst.), 2018, S. 062036

A Anhang

A.1 Erstellter und genutzer Code der Arbeit

Listing A.1: Notebook: Pipeline Sensorverarbeitung

```
{
from pandas import *
import matplotlib.pyplot as plt
import numpy as np
from scipy import *
from sympy import *
import pandas as pd
import sys
!{sys.executable} -m pip install distfit
from scipy.signal import argrelextrema
from scipy.integrate import quad
import distfit
import scipy.integrate as integrate

def dropNElem(mylist, n):
    newList = [item for index, item in enumerate(mylist) if index % n != 0]
    return newList

 Read data from csv-files
data1 = read_csv("Messungen/sync/stroke.csv", delimiter=",")
data2 = read_csv("Messungen/sync/bow.csv", delimiter=",")
data_boot = read_csv("Messungen/sync/boot.csv", delimiter=",")

 extract axis to work with
```

```
xAxis1 = round(data1['y−axis (g)']∗9.81, 2)
xAxis2 = round(data2['y−axis (g)']∗9.81, 2)
xAxisBoot = round(data_boot['y−axis (g)']∗9.81, 2)

 get some info about extracted data
print('xAxis1:',xAxis1)
print('xAxis2:',xAxis2)
print('xAxis3:',xAxisBoot)

 show extracted data in diagramm
plt.rcParams["figure.figsize"] = (20,3)
plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(xAxis1, 'b−', xAxis2, 'r−', xAxisBoot, 'g−')

 get value data should be shifted, if the Starting point is different
correlated = np.correlate(xAxis1, xAxis2)
plt.plot(correlated)
print("get best fitting position")
bestFittingPosition = np.argmax(correlated)
print(bestFittingPosition)

 shift data to the best fitting
xAxis1_shift = np.roll(xAxis1, bestFittingPosition)
plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(xAxis1_shift, 'b−', xAxis2, 'r−')

ectract around 3 Strokes to analyse them
xAxis1_3Strokes = xAxis1[80:450]
xAxis2_3Strokes = xAxis2[80:450]
plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(xAxis1_3Strokes, 'b−', xAxis2_3Strokes, 'r−')
delete every nth value to get the same fequence in data
xAxis2_3Strokes = dropNElem(xAxis2_3Strokes, 80) 60 fits well
```

```
correct indexes from cut list - starts with 800 without correcting
xAxis1_3Strokes_withIndex = []
for value in xAxis1_3Strokes:
    xAxis1_3Strokes_withIndex.append(value)

DfXAxis1_3Strokes_withIndex = pandas.DataFrame(xAxis1_3Strokes_withIndex)

xAxis2_3Strokes_withIndex = []
for value in xAxis2_3Strokes:
    xAxis2_3Strokes_withIndex.append(value)

DfXAxis2_3Strokes_withIndex = pandas.DataFrame(xAxis2_3Strokes_withIndex)

DfXAxis1_3Strokes_withIndex.head()

DfXAxis2_3Strokes_withIndex.describe()

 use correlation to get count of shifts needed ToDo : see why value 52 is not shown
correlated_3Strokes = np.correlate(xAxis1_3Strokes_withIndex,
    xAxis2_3Strokes_withIndex, mode='full')
plt.plot(correlated_3Strokes)
correlated_3Strokes.shape[0]/2

print("get best fitting position")
bestFittingPosition_3Strokes = np.argmax(correlated_3Strokes)
print(bestFittingPosition_3Strokes)
valueToShift = correlated_3Strokes.shape[0]//2 - np.argmax(correlated_3Strokes)

 shift data to the best fitting
xAxis1_shift_3Strokes = np.roll(xAxis1_3Strokes_withIndex, 4)
plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(xAxis1_shift_3Strokes, 'b-', xAxis2_3Strokes_withIndex, 'r-')
plt.plot(difference)
```

```
 entweder kreuzkorrelation oder timestamps anpassen
alle schlaege
print('Stroke 1-5: ', np.corrcoef(xAxis1_shift_3Strokes[80:264],
    xAxis2_3Strokes_withIndex[80:264])[1,0])
einzelne schlaege
todo genaue peaks vom Schlagmann nehmen
print('Stroke 1: ', np.corrcoef(xAxis1_shift_3Strokes[80:119],
    xAxis2_3Strokes_withIndex[80:119])[1,0])
print('Stroke 2: ', np.corrcoef(xAxis1_shift_3Strokes[119:158],
    xAxis2_3Strokes_withIndex[119:158])[1,0])
print('Stroke 3: ', np.corrcoef(xAxis1_shift_3Strokes[158:194],
    xAxis2_3Strokes_withIndex[158:194])[1,0])
print('Stroke 4: ', np.corrcoef(xAxis1_shift_3Strokes[194:229],
    xAxis2_3Strokes_withIndex[194:229])[1,0])
print('Stroke 5: ', np.corrcoef(xAxis1_shift_3Strokes[229:264],
    xAxis2_3Strokes_withIndex[229:264])[1,0])
schlag zerlegen peaks vergleichen -> umkehrpunkte
abweichungen lokalisieren

 pandify values
DFxAxis1_shift_3Strokes = pandas.DataFrame(integrate.cumtrapz(
    xAxis1_shift_3Strokes))
DFxAxis2_3Strokes_withIndex = pandas.DataFrame(integrate.cumtrapz(
    xAxis2_3Strokes_withIndex))
DFxAxis1_shift_3Strokes = pandas.DataFrame(xAxis1_shift_3Strokes)
DFxAxis2_3Strokes_withIndex = pandas.DataFrame(xAxis2_3Strokes_withIndex)

plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(DFxAxis1_shift_3Strokes.rolling(3).mean(), 'b-',
    DFxAxis2_3Strokes_withIndex.rolling(3).mean(), 'r-')
plt.plot(difference)
xAxisBoot = xAxisBoot[80:450]
plt.plot(xAxisBoot)

difference = DFxAxis2_3Strokes_withIndex -DFxAxis1_shift_3Strokes
```

```
plt.plot(difference)

df1 = pd.DataFrame(DFxAxis1_shift_3Strokes[190:325])
df2 = pd.DataFrame(DFxAxis2_3Strokes_withIndex[190:325])

plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(df1.rolling(3).mean(), 'b-', df2.rolling(2).mean(), 'r-')
plt.plot( diff(diff(df1.rolling(1).mean())), 'b-',df1.rolling(1).mean(), 'r-')

np.random.seed(0)
rs = np.random.randn(500)
xs = [0]
for r in rs:
    xs.append(xs[-1] * 0.9 + r)
df1 = pd.DataFrame(xs, columns=['data'])
df2 = pd.DataFrame(xs, columns=['data'])

n = 5 number of points to be checked before and after
.rolling(3).mean()
DFxAxis1_shift_3Strokes = DFxAxis1_shift_3Strokes.rename(columns={0: 'data'})
DFxAxis1_shift_3Strokes = DFxAxis1_shift_3Strokes[50:400]

 Find local peaks
df1['min'] = DFxAxis1_shift_3Strokes.iloc[argrelextrema(DFxAxis1_shift_3Strokes.
    data.values, np.less_equal,
                    order=n)[0]]['data']
df1['max'] = DFxAxis1_shift_3Strokes.iloc[argrelextrema(DFxAxis1_shift_3Strokes.
    data.values, np.greater_equal,
                    order=n)[0]]['data']

 Plot results

plt.scatter(df1.index, df1['min'], c='r')
plt.scatter(df1.index, df1['max'], c='g')
```

```
plt.xlabel("Zeit in sec/10")
plt.ylabel("Beschleunigung in m/s^2")
zeros = DFxAxis1_shift_3Strokes.copy()*0
plt.plot(DFxAxis1_shift_3Strokes.index, DFxAxis1_shift_3Strokes['data'], zeros['data
    '])
for i, label in enumerate(df1['min']):
    if not(np.isnan(label)):
        plt.text(i, label, '({}, {})'.format(i, label))

for i2, label2 in enumerate(df1['max']):
    if not(np.isnan(label2)):
        plt.text(i2, label2, '({}, {})'.format(i2, label2))

 Nullstellen ermitteln
interZ1 = (np.argwhere(np.diff(np.sign(zeros['data'] - DFxAxis1_shift_3Strokes['data
    ']))).flatten())+50
plt.plot(interZ1, np.zeros_like(interZ1), 'yo')
plt.plot(difference[150:400])
plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")

plt.show()

np.zeros_like(DFxAxis1_shift_3Strokes.index)
zeros = DFxAxis1_shift_3Strokes.copy()*0
print(zeros)

np.zeros_like(DFxAxis1_shift_3Strokes.index)
zeros = DFxAxis1_shift_3Strokes.copy()*0

 Generate a noisy AR(1) sample

np.random.seed(0)
rs2 = np.random.randn(500)
xs2 = [0]
for r in rs2:
```

```
    xs2.append(xs2[-1] * 0.9 + r)
df2 = pd.DataFrame(xs2, columns=['data'])

n = 5 number of points to be checked before and after
.rolling(3).mean()
DFxAxis2_3Strokes_withIndex = DFxAxis2_3Strokes_withIndex.rename(columns
    ={0: 'data'})
DFxAxis2_3Strokes_withIndex = DFxAxis2_3Strokes_withIndex[50:400]

 Find local peaks
df2['min'] = DFxAxis2_3Strokes_withIndex.iloc[argrelextrema(
    DFxAxis2_3Strokes_withIndex.data.values, np.less_equal,
                    order=n)[0]]['data']
df2['max'] = DFxAxis2_3Strokes_withIndex.iloc[argrelextrema(
    DFxAxis2_3Strokes_withIndex.data.values, np.greater_equal,
                    order=n)[0]]['data']

 Plot results

plt.scatter(df2.index, df2['min'], c='r')
plt.scatter(df2.index, df2['max'], c='g')
plt.xlabel("Zeit in ms")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(DFxAxis1_shift_3Strokes.index, DFxAxis1_shift_3Strokes['data'],
    DFxAxis2_3Strokes_withIndex.index, DFxAxis2_3Strokes_withIndex['data'],
    zeros['data'])

for i, label in enumerate(df2['min']):
    if not(np.isnan(label)):
        plt.text(i, label, '({}, {})'.format(i, label))

for i2, label2 in enumerate(df2['max']):
    if not(np.isnan(label2)):
        plt.text(i2, label2, '({}, {})'.format(i2, label2))

 Nullstellen ermitteln
```

```
interZ2 = (np.argwhere(np.diff(np.sign(zeros['data'] - DFxAxis2_3Strokes_withIndex
    ['data']))).flatten())+50
plt.plot(interZ2, np.zeros_like(interZ2), 'yo')

plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.savefig('/Users/jonasningelgen/Documents/Uni/BachelorArbeit/Quellen/peaksSync
    .eps', format='eps')
plt.show()
nullstellen der f' sind extremstellen von f

entweder kreuzkorrelation oder timestamps anpassen
alle schlaege
print('Stroke 1-5: ', np.corrcoef(xAxis1_shift_3Strokes[80:264],
    xAxis2_3Strokes_withIndex[80:264])[1,0])
einzelne schlaege peaks vom Schlagmann nehmen
print('Stroke 1: ', np.corrcoef(xAxis1_shift_3Strokes[80:119],
    xAxis2_3Strokes_withIndex[80:119])[1,0])
print('Stroke 2: ', np.corrcoef(xAxis1_shift_3Strokes[119:158],
    xAxis2_3Strokes_withIndex[119:158])[1,0])
print('Stroke 3: ', np.corrcoef(xAxis1_shift_3Strokes[158:194],
    xAxis2_3Strokes_withIndex[158:194])[1,0])
print('Stroke 4: ', np.corrcoef(xAxis1_shift_3Strokes[194:229],
    xAxis2_3Strokes_withIndex[194:229])[1,0])
print('Stroke 5: ', np.corrcoef(xAxis1_shift_3Strokes[229:264],
    xAxis2_3Strokes_withIndex[229:264])[1,0])
schlag zerlegen peaks vergleichen -> umkehrpunkte
abweichungen lokalisieren

len(DFxAxis1_shift_3Strokes)
df1.iloc[150:400]
len(DFxAxis2_3Strokes_withIndex)
df2.iloc[150:400]

DataFram
mins = df1.index[df1['min'] == df1['min']].tolist()
```

```
print('mins: ')
print(mins)
mins2 = df2.index[df2['min'] == df2['min']].tolist()
print('mins: ')
print(mins2)

maxs = df1.index[df1['max'] == df1['max']].tolist()
print('maxs: ')
print(maxs)
maxs2 = df2.index[df2['max'] == df2['max']].tolist()
print('maxs: ')
print(maxs2)

print('zeros: ')
print(interZ1)
print('zeros: ')
print(interZ2)

test = diff(df1.rolling(3).mean())
plt.xlabel("Messpunkte aufgezeichnet mit 12,5Hz")
plt.ylabel("Beschleunigung in m/s^2")
plt.plot(test, 'b-')
}
```

Listing A.2: Colab Notebook Pose Estimation Openpose

```
{
https://colab.research.google.com/github/tugstugi/dl-colab-notebooks/blob/master/
    notebooks/OpenPose.ipynb#scrollTo=oNASdyyiO65I Accessed: 2022-06-14
https://github.com/CMU-Perceptual-Computing-Lab/openpose - Accessed:
    2022-06-14

This notebook uses an open source project CMU-Perceptual-Computing-Lab/
    openpose to detect/track multi person poses on a given youtube

import os
from os.path import exists, join, basename, splitext
```

```
git_repo_url = 'https://github.com/CMU-Perceptual-Computing-Lab/openpose.git
    '
project_name = splitext(basename(git_repo_url))[0]
if not exists(project_name):
   see: https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/949
   install new CMake becaue of CUDA10
  !wget -q https://cmake.org/files/v3.13/cmake-3.13.0-Linux-x86_64.tar.gz
  !tar xfz cmake-3.13.0-Linux-x86_64.tar.gz --strip-components=1 -C /usr/local
   clone openpose
  !git clone -q --depth 1 $git_repo_url
  !sed -i 's/execute_process(COMMAND git checkout master
       WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}\/3rdparty\/caffe)/
       execute_process(COMMAND git checkout
       f019d0dfe86f49d1140961f8c7dec22130c83154 WORKING_DIRECTORY ${
       CMAKE_SOURCE_DIR}\/3rdparty\/caffe)/g' openpose/CMakeLists.txt
   install system dependencies
  !apt-get -qq install -y libatlas-base-dev libprotobuf-dev libleveldb-dev
       libsnappy-dev libhdf5-serial-dev protobuf-compiler libgflags-dev libgoogle-
       glog-dev liblmdb-dev opencl-headers ocl-icd-opencl-dev libviennacl-dev
   install python dependencies
  !pip install -q youtube-dl
   build openpose
  !cd openpose && rm -rf build || true && mkdir build && cd build && cmake .. &&
       make -j`nproc`

from IPython.display import YouTubeVideo

 YOUTUBE_ID = '1PXzCjeheoU?t=185'
YOUTUBE_ID = 'oqpeTG0BaoM?t'

YouTubeVideo(YOUTUBE_ID)

!rm -rf youtube.mp4
 download the youtube with the given ID
```

```
!youtube-dl -f 'bestvideo[ext=mp4]' --output "youtube.%(ext)s" https://www.
    youtube.com/watch?v=$YOUTUBE_ID
 cut the first 5 seconds
!ffmpeg -t 0:13 -i youtube.mp4 -ss 0:18 video.mp4
!ffmpeg -y -loglevel info -i youtube.mp4 -t 10 video.mp4
 detect poses on the these 5 seconds
!rm openpose.avi
!cd openpose && ./build/examples/openpose/openpose.bin --video ./sample_data/
    rowvid.mp4 --write_json ./output/ --display 0 --write_video ../openpose.avi
 convert the result into MP4
!ffmpeg -y -loglevel info -i openpose.avi output.mp4

def show_local_mp4_video(file_name, width=1920, height=1080):
  import io
  import base64
  from IPython.display import HTML
  video_encoded = base64.b64encode(io.open(file_name, 'rb').read())
  return HTML(data='''<video width="{0}" height="{1}" alt="test" controls>
                        <source src="data:video/mp4;base64,{2}" type="video/mp4"
                             />
                      </video>'''.format(width, height, video_encoded.decode('ascii')
                         ))

show_local_mp4_video('output.mp4', width=1920, height=1080)

!zip /content/openpose/output /content/openpose_json.zip

!zip -r '/content/openpose_json.zip' '/content/openpose/output'
}
```

Listing A.3: Notebook Video Pipeline

```
{
imports
import pandas as pd
import json
import csv
```

```
import os
from pandas import *
import matplotlib.pyplot as plt
import numpy as np
from scipy import interpolate

 get files in directory
directory = 'Messungen/openpose/output/'

OpenPoseOutputFiles = os.listdir

header = ['wrist_x-coordinate', 'wrist_y-coordinate', 'wrist_percent','foot_x-
    coordinate', 'foot_y-coordinate', 'foot_percent', 'wrist_count']

with open('person1.csv', 'w', encoding='UTF8', newline='') as f:
    writer = csv.writer(f)

     write the header
    writer.writerow(header)

with open('person2.csv', 'w', encoding='UTF8', newline='') as f:
    writer = csv.writer(f)

     write the header
    writer.writerow(header)

for i, file in enumerate(OpenPoseOutputFiles):
    inputFromFile = open(directory + file)
    dataFromFile = json.load(inputFromFile)
    keypointsPerson = dataFromFile['people'][0]['pose_keypoints_2d']
    data = [keypointsPerson[18],keypointsPerson[19],keypointsPerson[20],
        keypointsPerson[30],keypointsPerson[31],keypointsPerson[32],i]
     Other wrist 9 10 11
    with open('person1.csv', 'a', encoding='UTF8', newline='') as f:
        writer = csv.writer(f)
```

```
            write the data
            writer.writerow(data)

for i, file in enumerate(OpenPoseOutputFiles):
    inputFromFile = open(directory + file)
    dataFromFile = json.load(inputFromFile)
    keypointsPerson = dataFromFile['people'][1]['pose_keypoints_2d']
    data = [keypointsPerson[18],keypointsPerson[19],keypointsPerson[20],
        keypointsPerson[30],keypointsPerson[31],keypointsPerson[32],i]
     Other wrist 9 10 11
    with open('person2.csv', 'a', encoding='UTF8', newline='') as f:
        writer = csv.writer(f)

            write the data
            writer.writerow(data)

 Read data from csv-files
data1 = read_csv("person1.csv", delimiter=",")
data2 = read_csv("person2.csv", delimiter=",")

correct values beeing zero
previousVal = 0
for i, value in enumerate(data1['foot_x-coordinate']):
    if (value == 0.0):
        data1['foot_x-coordinate'][i] = (data1['foot_x-coordinate'][i-1])+ data1['
            foot_x-coordinate'][i+1])/2
    previousVal = value

previousVal = 0
for i, value in enumerate(data2['foot_x-coordinate']):
    if (value == 0.0):
        data2['foot_x-coordinate'][i] = (data2['foot_x-coordinate'][i-1])+ data2['
            foot_x-coordinate'][i+1])/2
    previousVal = value

previousVal = 0
```

```
for i, value in enumerate(data1['foot_y-coordinate']):
    if (value == 0.0):
        data1['foot_y-coordinate'][i] = (data1['foot_y-coordinate'][i-1])+ data1['
            foot_y-coordinate'][i+1])/2
    previousVal = value

previousVal = 0
for i, value in enumerate(data2['foot_y-coordinate']):
    if (value == 0.0):
        data2['foot_y-coordinate'][i] = (data2['foot_y-coordinate'][i-1])+ data2['
            foot_y-coordinate'][i+1])/2
    previousVal = value

correct values beeing zero
previousVal = 0
for i, value in enumerate(data1['wrist_x-coordinate']):
    if (value == 0.0):
        data1['wrist_x-coordinate'][i] = (data1['wrist_x-coordinate'][i-1])+ data1['
            wrist_x-coordinate'][i+1])/2
    previousVal = value

previousVal = 0
for i, value in enumerate(data2['wrist_x-coordinate']):
    if (value == 0.0):
        data2['wrist_x-coordinate'][i] = (data2['wrist_x-coordinate'][i-1])+ data2['
            wrist_x-coordinate'][i+1])/2
    previousVal = value

correct values beeing zero
previousVal = 0
for i, value in enumerate(data1['wrist_y-coordinate']):
    if (value == 0.0):
        data1['wrist_y-coordinate'][i] = (data1['wrist_y-coordinate'][i-1])+ data1['
            wrist_y-coordinate'][i+1])/2
    previousVal = value
```

```
previousVal = 0
for i, value in enumerate(data2['wrist_y-coordinate']):
    if (value == 0.0):
        data2['wrist_y-coordinate'][i] = (data2['wrist_y-coordinate'][i-1])+ data2['
            wrist_y-coordinate'][i+1])/2
    previousVal = value

plt.rcParams["figure.figsize"] = (20,3)
plt.plot(data1['foot_x-coordinate'], 'b-',data2['foot_x-coordinate'], 'r-')

xAxisPerson1Wrist = data1['wrist_x-coordinate']
xAxisPerson2Wrist = data2['wrist_x-coordinate']

yAxisPerson1Wrist = data1['wrist_y-coordinate']
yAxisPerson2Wrist = data2['wrist_y-coordinate']

 show extracted data in diagramm
plt.rcParams["figure.figsize"] = (20,3)
plt.plot(xAxisPerson1Wrist.rolling(10).mean(), 'b-', xAxisPerson2Wrist.rolling(10).
    mean(), 'r-')

xAxisPerson1Foot = data1['foot_x-coordinate']
xAxisPerson2Foot = data2['foot_x-coordinate']

yAxisPerson1Foot = data1['foot_y-coordinate']
yAxisPerson2Foot = data2['foot_y-coordinate']


 show extracted data in diagramm
plt.rcParams["figure.figsize"] = (20,3)
plt.plot(xAxisPerson1Foot, 'b-', xAxisPerson2Foot, 'r-')

plt.plot(xAxisPerson1Wrist, 'b-', xAxisPerson1Foot, 'r-')
```

showing the distance between wrist and foot,
to see if you can read back and front of stroke per person, and from that get synchrony

```
 try only by subtrating axes

wristMinusFoot1 = np.array([])
for i, value in enumerate(xAxisPerson1Wrist):
    wristMinusFoot.append(xAxisPerson1Foot[i]- xAxisPerson1Wrist[i])
    wristMinusFoot1 = np.append(wristMinusFoot1, xAxisPerson1Wrist[i] -
        xAxisPerson1Foot[i])
wristMinusFoot2 = np.array([])
print(xAxisPerson1Wrist[73])
print(xAxisPerson1Foot[73])
for i, value in enumerate(xAxisPerson2Wrist):
    wristMinusFoot.append(xAxisPerson1Foot[i]- xAxisPerson1Wrist[i])
    wristMinusFoot2 = np.append(wristMinusFoot2, xAxisPerson2Wrist[i] -
        xAxisPerson2Foot[i])

print(xAxisPerson2Wrist[73])
print(xAxisPerson2Foot[73])

df1 = pd.DataFrame(wristMinusFoot1[73:364])
df2 = pd.DataFrame(wristMinusFoot2[73:364])
plt.plot(df1.rolling(15).mean(), 'b-')
plt.plot(df2.rolling(15).mean(), 'r-')

 try with 1,0,-1
showing the distance between wrist and foot,
to see if you can read back and front of stroke per person, and from that get synchrony
 try only by subtrating axes

wristMinusFoot1 = np.array([])
for i, value in enumerate(xAxisPerson1Wrist):
    wristMinusFoot.append(xAxisPerson1Foot[i]- xAxisPerson1Wrist[i])
    if (0 < xAxisPerson1Wrist[i] - xAxisPerson1Foot[i]):
        wristMinusFoot1 = np.append(wristMinusFoot1, 1)
    else:
        wristMinusFoot1 = np.append(wristMinusFoot1, 0)
wristMinusFoot2 = np.array([])
```

```
for i, value in enumerate(xAxisPerson2Wrist):
    if (0 < xAxisPerson2Wrist[i] - xAxisPerson2Foot[i]):
        wristMinusFoot2 = np.append(wristMinusFoot2, 1)
    else:
        wristMinusFoot2 = np.append(wristMinusFoot2, -1)
df1 = pd.DataFrame(wristMinusFoot1[73:364])
df2 = pd.DataFrame(wristMinusFoot2[73:364])
plt.plot(df1.rolling(1).mean(), 'b-')
plt.plot(df2.rolling(1).mean(), 'r-')

 try obove thing with eukleadian distance
disHandToFootP1 = np.sqrt((yAxisPerson1Foot - yAxisPerson1Wrist)**2 + (
    xAxisPerson1Foot - xAxisPerson1Wrist)**2)
df1Euklid = pd.DataFrame(disHandToFootP1[73:344])
plt.plot(df1Euklid.rolling(5).mean(), 'b-')

 try obove thing with eukleadian distance
disHandToFootP2 = np.sqrt((yAxisPerson2Foot - yAxisPerson2Wrist)**2 + (
    xAxisPerson2Foot - xAxisPerson2Wrist)**2)
df2Euklid = pd.DataFrame(disHandToFootP2[73:344])
plt.plot(df2Euklid.rolling(5).mean(), 'r-')

distanz zwischen Handgelenken
disWrist = np.sqrt((yAxisPerson1Wrist - yAxisPerson2Wrist)**2 + (
    xAxisPerson1Wrist- xAxisPerson2Wrist)**2)
plt.plot(disWrist[0:100], 'b-')

distanz zwischen Fuessen
disFoot = np.sqrt((yAxisPerson1Foot - yAxisPerson2Foot)**2 + (xAxisPerson1Foot
    - xAxisPerson2Foot)**2)
plt.plot(disFoot)
plt.plot(disWrist)
}
```

A.2 Erklaerung zur selbststaendigen Bearbeitung

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der ist erfolgt durch:

[redacted]

Ort | Datum | Unterschrift im Original