OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Bachelorarbeit

Automatisierte Wiederholungs- und Übungserkennung mithilfe einer Smartwatch

Entwicklung einer Software zur Erkennung von Fitnessstudio Übungen sowie automatisches Zählen der Wiederholungen

Betreuer: Prof. Dr. Wolfgang Mauerer
Zweitbetreuer: Dr. Ralf Ramsauer
Zweitprüfer: Prof. Dr. Markus Kucera

Eingereicht von
Markus Schottenhammer
Matrikelnummer: 3270732
Fakultät: Informatik und Mathematik

Eingereicht am
13.03.2024

Abkürzungsvereichnis

CNN Convolutional Neural Network

SVM Support Vector Machine

KNN K-Nearest-Neighbor

DTW Dynamic Time Warping

LIFO Last in - first out

PCO Principal Component Analysis

RNN Recurrent Neural Networks

TN Teilnehmer

Inhaltsverzeichnis

1 Abstract

Ziel dieser Arbeit war es eine Anwendung zu entwickeln welche die Wiederholungszahlen von Fitnesstudioübungen und die Übung selbst korrekt erkennen kann. Wichtig dabei war es neu angelegte Übung sofort ohne weiteren Trainingsaufwand erkennen zu können. Außerdem sollte die Anwendung flexibel erweiterbar sein und die Zeit zwischen Wiederholung und Ausgabe der Ergebnisse möglichst gering halten.

Erreicht werden konnten diese Anforderungen durch die Kombination des von Zhang et al. vorgestellten Algorithmus mit dynamischen Grenzwert mit dem Dynamic Time Warping (DTW)-Algorithmus. [1]

Der Algorithmus mit dynamischen Grenzwert trifft dabei eine Vorauswahl an möglichen Wiederholungsbereichen, welche im Anschluss vom DTW-Algorithmus mit der zuvor gespeicherten Vorlage der Übung verglichen wurde. Unterschreitet der berechnete Wert einen, während der Kalibrierung festgelegten, Grenzwert wird dieser Bereich als gültig Wiederholung erkannt.

Von insgesamt 96 durchgeführten Sätzen mit gesamt 1018 Wiederholungen wurden 28.125 % exakt erkannt, 64.583 % mit einer maximalen Abweichung von 1 und 82.291 % mit einer maximalen Abweichung von 2 Wiederholungen. Weitere Verbesserungen sind somit vor einem kommerziellen Einsatz notwendig. Die Tests zeigten das sehr häufig durch das Wechseln von Ruhe- in Ausgangsposition bereits eine Wiederholung erkannt wurde, was die gesamten Messungen stark verschlechterte. Mögliche Lösungsansätze dafür werden unter Unterunterabschnitt 10.2.1 vorgestellt.

Die umgesetzte Lösung erfüllt die gestellten Anforderungen, bietet aber noch umfangreiche Erweiterungsmöglichkeiten. Besonders der Einsatz vom Machine Learning, welches auf Basis der korrekt erkannten Wiederholungsbereiche arbeitet, scheint vielversprechend. Auf diese und weitere Verbesserungsmöglichkeiten wird im Ausblick eingegangen.

2 Motivation

Beim Sporttraining im Fitnessstudio ist es von entscheidender Bedeutung, die Leistung kontinuierlich und progressiv zu steigern, um langfristig Fortschritte zu erzielen. Dies gilt nicht nur für professionelle Sportler, sondern auch für Freizeitsportler, die ihre Kraft und Ausdauer verbessern möchten. Es gibt zahlreiche Apps, die bei der Umsetzung dieses Ziels unterstützen, indem sie Wiederholungszahlen und Gewichte speichern können. Allerdings haben diese Apps ein gemeinsames Problem: Viele Nutzer haben Schwierigkeiten, die Daten konsequent einzugeben, da die Verwendung des Smartphones ablenkend oder störend sein kann. Außerdem kann es leicht passieren, dass das Eintragen der Daten vergessen wird, was zu Lücken in der Dokumentation führen kann.

Die große Auswahl an verschiedenen Apps mit vielen Downloads zeigt ein breites Interesse an diesem Bereich. Die Apps unterscheiden sich bis auf kleinere Unterschiede kaum voneinander. Im Vorfeld dieser Arbeit wurde bereits eine unveröffentlichte Anwendung zum manuellen Erfassen von Übungen und Wiederholungen erstellt. Das Ziel dieser Arbeit ist es, ein Alleinstellungsmerkmal für diese Anwendung zu entwickeln und sie anschließend zu vermarkten.

3 Zielsetzung

In den letzten Jahren hat sich die Verbreitung von Smartwatches signifikant erhöht. [2], [3] Diese Geräte verfügen über Sensoren, die zur Erkennung von Bewegungen eingesetzt werden können. Das Ziel dieser Arbeit ist es, diese Sensoren zu nutzen, um sowohl die Übung als auch die Anzahl der Wiederholungen korrekt zu erkennen.

Aufgrund der Vielzahl an Übungen und Variationen ist es notwendig, das System flexibel zu gestalten. Es ist nicht vertretbar, alle diese Übungen von Anfang an in das System einzupflegen.

Das System soll direkt nach der Erstellung neuer Übungen eine anfängliche Bewertung abgeben können.

Die Arbeit beschäftigt sich konkret mit folgenden Forschungsfragen:

- Wie können die Anzahl der Wiederholungen von Fitnessstudioübungen und die Übung selbst auf Basis von Smartwatch-Sensordaten möglichst präzise ermittelt werden?
- Wie kann eine flexible Erweiterbarkeit erreicht werden, die es ermöglicht, beliebige neue Übungen zu integrieren?

Optimalitätskriterien

Präzision

Der Fokus der Arbeit liegt darauf, die Anzahl der Wiederholungen genau zu erkennen und anzuzeigen.

Es ist anzunehmen, dass bereits geringe Abweichungen bei der Wiederholungszählung aufgrund der hohen Standards, an die Endnutzer gewöhnt sind, die Akzeptanz stark senken würden.

Performance

Das System soll die Verzögerung zwischen der letzten Wiederholung und der Ausgabe des Ergebnisses minimieren.

Dies ist wichtig, da der Nutzer möglicherweise die Wiederholungen selbst zählt und überprüfen möchte, ob die Wiederholungserkennung korrekt funktioniert hat,

um sie gegebenenfalls anzupassen. Wenn der Nutzer nach der letzten Wiederholung lange warten muss, würde dies ebenfalls die Nutzerakzeptanz senken.

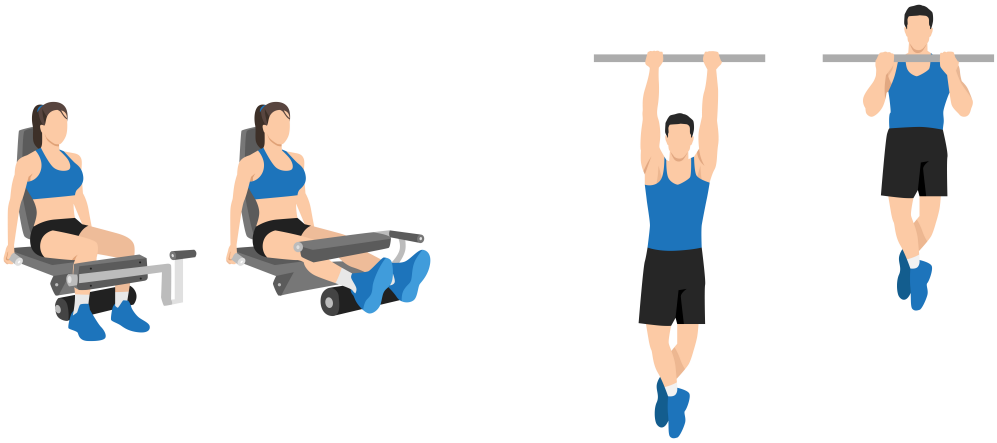
Flexibilität / Skalierbarkeit

Das System soll so konzipiert sein, dass es einfach erweiterbar ist und neue Übungen integriert werden können.

Wenn dem Nutzer einige Übungen zum Aufzeichnen nicht zur Verfügung stehen, muss er zwischen der Verwendung der Anwendung und manuellem Erfassen wechseln. Aus diesem Grund ist eine Begrenzung der Übungen nicht akzeptabel.

Limitierungen

Einige der Übungen, insbesondere Beinübungen, erfordern keine wesentlichen Handgelenksbewegungen (Beispiele: Abbildung 1). Eine Erkennung dieser Bewegungen ist ohne weitere Sensoren sehr wahrscheinlich nicht möglich. Keiner der bekannten Ansätze geht auf dieses Problem ein (Siehe Literaturrecherche). Die Verwendung zusätzlicher Sensoren würde die Akzeptanz unter den Nutzern deutlich verringern und ist deshalb keine Option. Ein möglicher Lösungsansatz wird unter Unterunterabschnitt 10.2.5 vorgestellt.



(a) (b)

Abbildung 1: Beispiele für nicht oder sehr schlecht erkennbare Übungen. 1a: Beinstrecker, 1b: Klimmzug

4 Grundlagen

4.1 DTW

Um die Ähnlichkeit zweier Datenreihen zu messen, kann z.B. die Euklidische Distanz verwenden und jeweils genau übereinander liegende Datenpunkte vergleichen, wie in Abbildung 2 gezeigt. Die Summe der Abstände bietet einen Anhaltspunkt wie weit die Daten übereinstimmen.

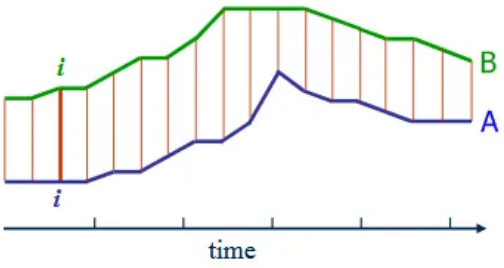


Abbildung 2: Vergleich Datenreihen, übereinander liegender Datenpunkte (Quelle: [4])

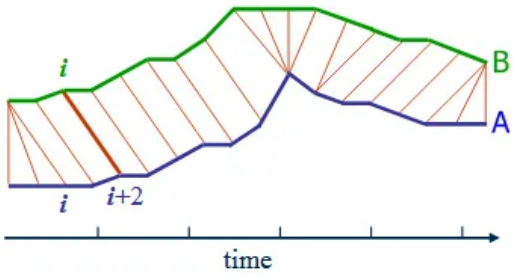


Abbildung 3: Vergleich Datenreihen mit dem DTW-Algorithmus. (Quelle: [4])

Der Ansatz mit der euklidischen Distanz funktioniert nur, wenn die Datenreihen identisch sind und lediglich geringe Abweichungen in vertikaler Richtung aufweisen. Sobald die Datenreihen jedoch zeitlich um nur einen Datenpunkt verschoben sind, verliert dieser Ansatz seine Aussagekraft.

Der DTW-Algorithmus kann diese Aufgabe lösen, indem er, wie in Abbildung 3 dargestellt, jeden Punkt mit allen Punkten der anderen Datenreihen vergleicht und nach der kürzesten Gesamtstrecke sucht.

Für die Berechnung wird eine Matrix genutzt (siehe Abbildung 4), in der die Abstände zwischen jedem Punkt berechnet werden. Anschließend wird der kürzeste Pfad durch die Matrix gesucht. Der Pfad muss am Anfang beider Reihen starten und am Ende beider Reihen enden, um sicherzustellen, dass jeder Punkt betrachtet wird. Außerdem müssen beide Zählervariablen monoton erhöht werden. Im Falle, dass beide Datenreihen exakt gleich sind, ergibt sich eine gerade, diagonale Linie in der Matrix.

Um den optimalen Pfad durch die Matrix zu finden, müssen alle Werte der Matrix berechnet werden. Hierfür wird ein Ansatz der dynamischen Programmierung genutzt. Der Wert einer Zelle ergibt sich aus dem Minimum der niedrigeren anliegenden Zellen sowie dem Pfad zwischen den in der aktuellen Zelle betrachteten Punkten. Oder formell ausgedrückt:

$$D(i,j) = Dist(i,j) + min[D(i-1,j), D(i,j-1), D(i-1,j-1)]$$

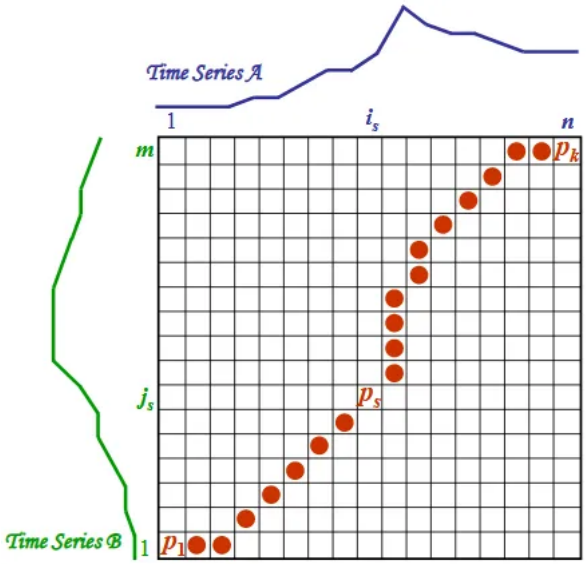


Abbildung 4: Darstellung der Kostenmatrix beim Vergleich (Bildquelle: [4])

Aufgrund dessen muss die Matrix von unten nach oben und von links nach rechts befüllt werden.[5]

Alle Zellen müssen genau einmal mit konstanter Zeit befüllt werden. Dadurch hat der Algorithmus eine Zeit- und Raumkomplexität von $|X| * |Y|$ was $O(N^2)$ entspricht wenn $N = |X| = |Y|$.

4.2 Fast DTW

Es gibt drei Ansätze zur Verbesserung der Performance des DTW-Algorithmus:

- Beschränkungen für die zu berechnende Matrixzellen
- Abstraktion der Eingangsdaten
- Indizierung durch das Finden von ähnlichen Datenabschnitten

Beschränkungen für die zu berechnende Matrixzellen

Die Beschränkung der Zellen basiert darauf, dass in den meisten Fällen nur geringe Verschiebungen zwischen den Datenreihen vorliegen. Deshalb ist es unwahrscheinlich, dass der ideale Weg durch die Kostenmatrix stark von der Diagonalen abweicht. Aus diesem Grund wird nicht die komplette Kostenmatrix berechnet, sondern nur die Felder innerhalb festgelegter Beschränkungen. Die bekanntesten Ansätze sind das Itakura-Parallelogramm und das Sakoe-Chiba-Band (Siehe: Abbildung 5). Bei starken Verschiebungen kann ein Pfad durch die Matrix ideal sein, der stark von der Diagonalen abweicht.

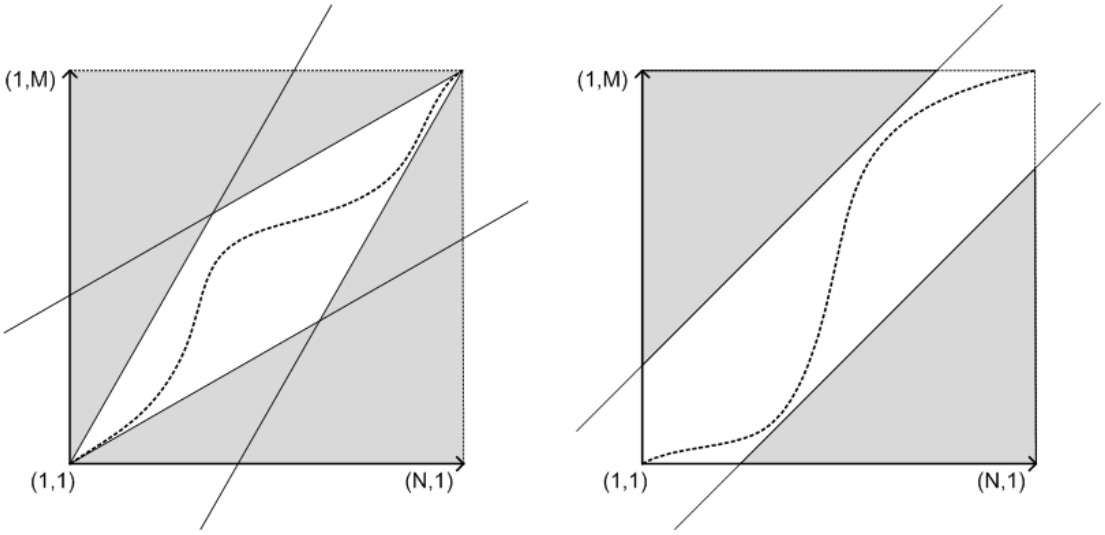


Abbildung 5: Beschränkung der Zellen der Berechnungsmatrix. Links: Itakura-Parallelogramm, Rechts: Sakoe-Chiba-Band (Quelle: [6])

Abstraktion der Eingangsdaten

Die Abstraktion baut darauf auf, die Datenreihen durch Zusammenfassen stark zu reduzieren und auf der reduzierten Datenmenge die Berechnungen durchzuführen. Der gefundene Pfad wird dann genutzt, um die Zellen mit den Originaldaten stark einzuschränken (siehe Abbildung 6). Dies hat allerdings den gleichen Nachteil wie die Beschränkungen, dass eine starke Abweichung vom idealen Pfad entstehen kann.

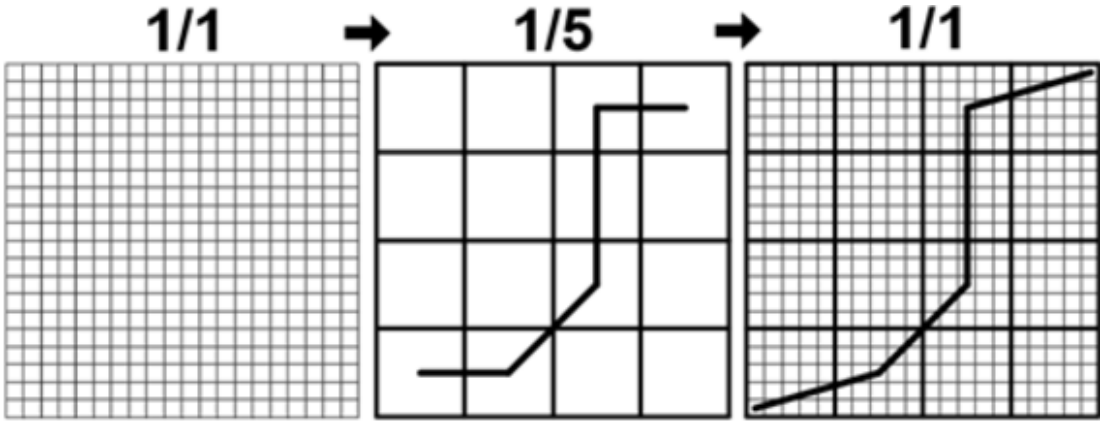


Abbildung 6: Berechnungen mit den reduzierten Daten, anschließend im eingeschränkten Bereich auf allen Daten (Bildquelle: [5])

Indizierung durch das Finden von ähnlichen Datenabschnitten

Die Indizierung ermöglicht es, die Anzahl der Ausführungen des DTW-Algorithmus zu verringern, wenn mehrere Datensätze durchsucht werden müssen. Hierbei werden sogenannte 'lower bounding functions' verwendet, welche mit deutlich weniger Berechnungen eine Einschätzung des DTW-Werts liefern können. Diese Einschätzung ist immer niedriger als der tatsächliche DTW-Wert. Nur in Fällen, die besser als der aktuell Beste tatsächliche DTW-Wert sind, muss die komplette DTW-Berechnung ausgeführt werden. Ein Beispiel für die Verwendung einer solchen Funktion ist unter Listing 1 ersichtlich. Es gibt verschiedene Ansätze zur Berechnung dieser Untergrenze. [7]–[9]

Für den gegebenen Anwendungsfall ist dieser Ansatz jedoch nicht geeignet, da die aktuellen Daten nur mit einer aufgezeichneten Wiederholung verglichen werden. Dieser Ansatz ist nur sinnvoll, um Berechnungen zu sparen, wenn es darum geht, die beste Übung aus dem aktuell gewählten Trainingsplan vorzuschlagen. Weitere Informationen siehe Ausblick.

Listing 1: Pseudocode für den Ausschluss von Datensätzen aus der DTW-Berechnung. (Quelle Code: [8])

```
best_so_far = infinity;
for all sequences in database
    LB_dist = lower_bound_distance(Ci,Q);
    if LB_dist < best_so_far
        true_dist = DTW(Ci,Q);
        if true_dist < best_so_far
            best_so_far = true_dist;
            index_of_best_match = i;
        endif
    endif
endfor
```

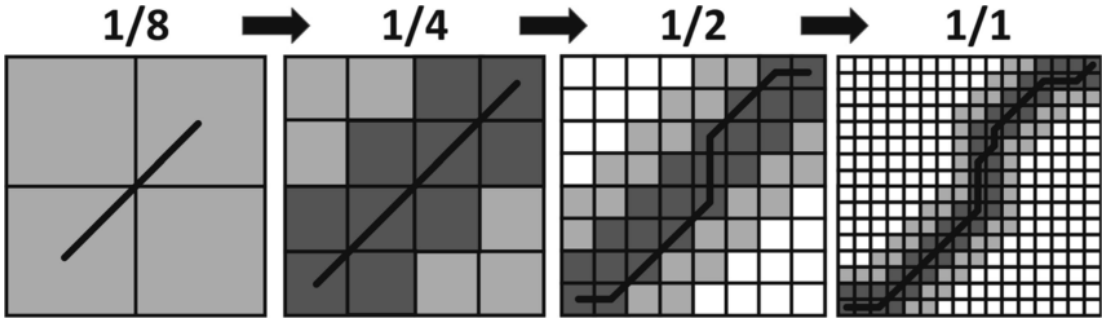


Abbildung 7: Vier Auflösungen die im FastDTW-Algorithmus nacheinander durchlaufen werden (Bildquelle: [5])

Um die Performance zu verbessern, greift man auf eine Kombination aus Beschränkung und Abstraktion zurück. Dieser Ansatz wurde von Stan Salvador und Phillip Chan in [5] vorgestellt.

Dabei werden zwei nebeneinander liegende Werte über den Durchschnitt zusammengefasst, sodass sich die Datenmenge halbiert. Dieser Schritt wird mehrmals wiederholt, um verschiedene Auflösungen zu erhalten. Anschließend wird in der niedrigsten Auflösung der DTW-Algorithmus durchgeführt und der gefundene Pfad an die nächsthöhere Auflösung übergeben. Dieses Vorwissen wird dann dort genutzt, um nur die anliegenden Felder dieses Pfades zu betrachten. Die Menge der angrenzenden Felder wird durch einen Betrachtungsradius bestimmt. Dieser ist in Abbildung 7 durch die hellgrauen Felder gekennzeichnet.

Der FastDTW-Algorithmus garantiert nicht, den optimalen Pfad zu finden, aber die Wahrscheinlichkeit, stark vom idealen Wert abzuweichen, ist deutlich geringer als bei der alleinigen Beschränkung.

4.3 Sich anpassender Grenzwert

Der Algorithmus passt den Grenzwert anhand der maximalen und minimalen Werte an, wie in Abbildung 8 sichtbar. Zhang et al. stellen zwei Bedingungen für eine Erkennung. Sobald der aktuelle Wert unter den Grenzwert fällt und dabei mindestens die adaptive Genauigkeit überschreitet, wird eine Wiederholung erkannt. [1] In dieser Arbeit wurde außerdem eine neue Anforderung definiert. Es wird durch den Algorithmus festlegt, dass ein Mindestabstand zwischen zwei Wiederholungen liegen muss, da zwei Wiederholungen innerhalb weniger Millisekunden nicht möglich sind. Der Abstand beträgt die Hälfte der durchschnittlichen Übungslänge.

$$BedingungI : |Value_{neu} - Value_{alt}| > AdaptiveGenauigkeit$$

$$BedingungII : Value_{neu} < AdaptiverGrenzwert < Value_{alt}$$

$$BedingungIII : time_{neu} - time_{alt} > time_{letzeErkennung} + Mindestabstand$$

Die adaptive Genauigkeit wird mit $\alpha * (MaximumWerte - MinimumWerte)$ berechnet. Der in der urspünglichen Arbeit verwendete Faktor α von 0.3 bis 0.4 wurde erneut verwendet. Diese Präzision ist notwendig, da die Sensoren im Ruhezustand ein Rauschen aufweisen. Wenn dieser Wert im Bereich des adaptiven Grenzwerts liegt, kann es dazu führen, dass falsche Wiederholungen erkannt werden. Die Präzision verhindert dieses Problem. Eine weitere Änderung im Vergleich zum ursprünglichen Algorithmus besteht darin, dass die adaptive Präzision im Bereich zwischen 0.5 und 1 liegen muss. Andernfalls wird der jeweilige Grenzwert verwendet. In Ausnahmefällen kann es vorkommen, dass der Wert entweder extrem niedrig ist und somit durch Rauschen ausgelöst wird oder sehr hoch ist und nicht mehr reagiert. Die Werte wurden auf Basis der durchschnittlichen adaptiven Grenzwerte festgelegt.

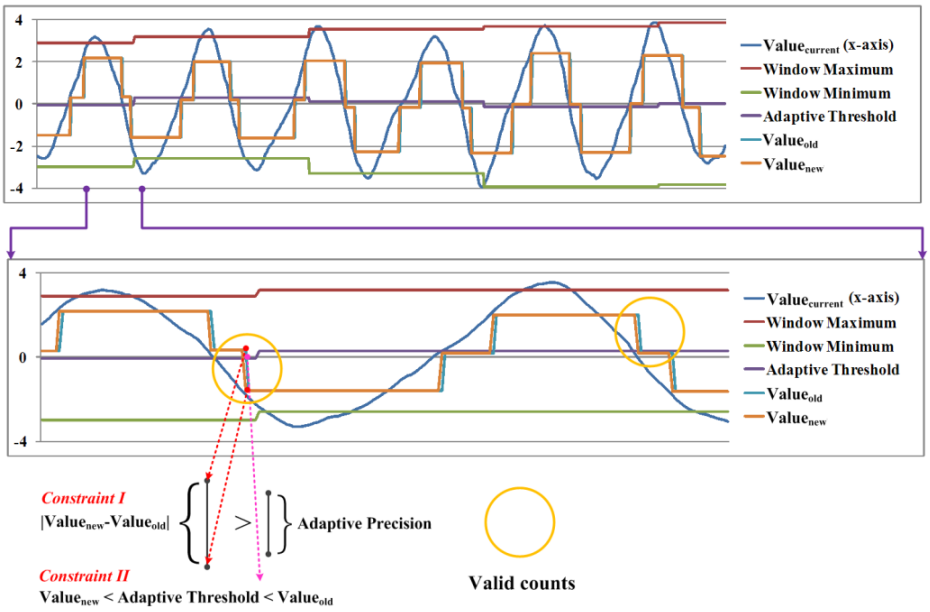


Abbildung 8: Erkennung gültiger Wiederholungen anhand einer Achse (Quelle: [1])

Listing 2: Pseudocode Identifikation möglicher Wiederholungen (Quelle: [1])

```
var newValue = newEntry.getValueForCounting();
//Ober- und Untergrenzen setzen und adaptive Werte anpassen
if (newValue > _max) { ... }
if (newValue < _min) { ... }
// Wenn der Unterschied groesser als erlaubte Toleranzen,...
if ((newValue.abs() - _lastValidInput.abs()).abs() >
    adaptivePercision) {
  // ...& Grenzwert unterschritten & Mindestabstand eingehalten wird
  if (newValue < _threshold &&
      _lastValidInput > _threshold &&
      newEntry.time - _lastPossibleRepTime >= exercise.minDistance) {
      //...setze Werte und gib eine moegliche Wiederholung zurueck
      _lastPossibleRepTime = newEntry.time;
      _lastValidInput = newValue;
      return PossibleRepetition(exercise, newEntry.time);
  } else {
      //... der Wert nicht die Randbedingungen erfuellt
      _lastValidInput = newValue;
  }
} else { return null; }
return null;
```

5 Literaturrecherche

Im Folgenden sollen die relevantesten bestehenden Arbeiten betrachtet werden. Die Hauptauswahlkriterien waren die grundsätzliche Möglichkeit, die Systeme mit geringem Aufwand um neue Übungen zu erweitern, eine hohe Erkennungsgenauigkeit und die Möglichkeit, Wiederholungen zu zählen.

5.1 RecoFit

RecoFit dient als eine der wichtigsten Grundlagen für die Arbeiten von Shen et al., Taewoong et al. und Soro et al. weshalb es hier genauer betrachtet wird. [10]–[13]

Übersicht

Die Autoren von RecoFit bezeichnen die Erkennung von Übungen und Pausen als eine der größten Schwierigkeiten auf diesem Gebiet. [10]

Es wurde festgestellt, dass bei stationären Übungen insbesondere die Rotationsachse, die vom Beschleunigungssensor aufgezeichnet wird, ein sich wiederholendes Muster aufweist. Zur Unterscheidung von Übungen und Pausen wird Autokorrelation genutzt, da Übungen typischerweise ein sich stark wiederholendes Muster aufweisen. Autokorrelation beschreibt in einer Zeitreihe die statistische Abhängigkeit zwischen den Werten zu verschiedenen Zeitpunkten. Um periodische Aktionen während der Pausen wie Gehen oder Dehnübungen auszufiltern, wird Machine Learning eingesetzt. Zur Datensammlung wurde ein typischer Trainingsraum nachgebaut, um eine echte Umgebung zu simulieren. Die erfassten Daten wurden manuell von einem Beobachter markiert. Es wurde ein Sliding-Window mit einer Länge von 5 Sekunden und einer Überschneidung von 4,8 Sekunden verwendet. Für jedes dieser Fenster wurden aus den Beschleunigungs- und Gyroskopdaten 224 Werte berechnet und als Featurematrix genutzt für eine Support Vector Machine (SVM), um zwischen Trainings- und Pausenzeiten zu unterscheiden. Um kurzzeitige falsche Erkennungen auszugleichen, wird nur in den Trainingszustand gewechselt, wenn ein definierter Grenzwert an Trainingserkennungen erreicht wird. Um die Übungen zu erkennen, werden aus drei Eingangssignalen jeweils 20 Merkmale berechnet. Anschließend entscheidet eine SVM, um welche Übung es sich handelt. Das Zählen der Wiederholungen basiert, nach einigen Filterungen, auf dem Zählen der Spitzen innerhalb eines Zeitfensters. Eine genauere Beschreibung siehe 6.3. Die Segmentierungsgenauigkeit mit einer maximalen Abweichung von zwei Sekunden zu Beginn und Ende betrug 85,6 Prozent. Die Erkennungsrate der Übungen lag offline im Durchschnitt bei 98,2 Prozent. Bei der Verwendung des Segmentierers wurden Wiederholungen in 94 Prozent der Fälle erkannt, wobei die maximale Abweichung eins betrug.

Stärken und Schwächen

Das manuelle Markieren der Übungen hat zur Folge, dass der Prozess wiederholt werden muss, um Daten für neue Übungen hinzuzufügen. Der Nachbau der Trainingsumgebung kann nicht alle typischen Szenarien abbilden, wie zum Beispiel Wartezeiten auf ein belegtes Gerät. Eine weitere Schwäche ist die hohe Anzahl an Features, die aus den Signalen berechnet werden. Es wurde nicht berücksichtigt, ob die entsprechenden Berechnungen auf einem durchschnittlichen Smartphone innerhalb einer angemessenen Zeit durchgeführt werden können.

5.2 PUSH

PUSH kann zwar nur durch erneutes Training erweitert werden, besitzt jedoch eine Datenbank mit 300 Übungen, von denen bereits 50 unterstützt werden. Daher sind Erweiterungen fast nicht notwendig. [12]

Übersicht

Die Arbeit verwendet den nicht mehr verfügbaren PUSH-Sensor sowie Datensätze, die von Athleten manuell markiert wurden und über 300 Übungen unterscheiden. [14] Es wurde sich gegen Recurrent Neural Networks (RNN) entschieden da die Datensätze sehr kurz sind. Als relevanteste Arbeit wird RecoFit angeführt. Als negative Punkte von RecoFit werden die kontrollierte Experimentumgebung, die minimale Länge von 20 Sekunden pro Datensatz, die begrenzte Übungsauswahl und die großen Unterschiede der gewählten Übungen angeführt.

Zur Erkennung wurden nur die 50 häufigsten Übungen trainiert. Das System geht von 784 Signalen aus, da eine Wiederholung in 99 Prozent der Datensätze kürzer als 3.92 Sekunden oder 784 Signale ist. Da Convolutional Neural Networks (CNNs) eine fixe Datenlänge benötigen werden die Datensätze auf 784 Werte aufgefüllt. Längere Datensätze werden nach dieser Signallänge unterteilt. Als Signaldaten werden die drei Achsen des Beschleunigungssensors im lokalen und globalen Raum sowie die drei Achsen des Eulerwinkels genutzt.

Für die Umwandlung in 2D-Bilder, die von CNNs verarbeitet werden können, wurden drei Varianten untersucht: (1) ein rechteckiges Signal mit 9 mal 784, (2) die Verwendung der RGB-Kanäle pro Signal oder (3) die Umwandlung der ersten Variante in ein quadratisches Signal. Dadurch können in der ersten Variante unzusammenhängende Eigenschaften direkt nebeneinander liegen, während in der dritten Variante zusammenhängende Eigenschaften auseinandergerissen werden können. Um dies in Variante 3 zu vermeiden, wurde als alternativer Ansatz (4) das Modell mit einer Schrittgröße von 3 festgelegt. Die Verwendung von Farbkanälen (2) hat den Vorteil, dass dies nicht passieren kann. Die Testergebnisse zeigten, dass Variante 2 mit einer Erkennungsrate von 90,47 % die besten Ergebnisse liefert. Weitere Tests zeigten, dass CNN Modelle mit 3 Layern die besten Ergebnisse erzielen (92,14 %).

Stärken und Schwächen

Die Auswahl der Signalquellen wird nicht begründet. Das System ist nicht flexibel erweiterbar und es wird nicht berücksichtigt, wie es sich unter realen Bedingungen verhält. Stattdessen wird nur auf den zuvor gesammelten Daten gearbeitet. Es wird auch nicht untersucht, ob die benötigten Berechnungen in angemessener Zeit auf einem Smartphone durchgeführt werden können.

5.3 MiLift

Übersicht

MiLift weist als Schwäche von RecoFit unter anderem auf die limitierte Anzahl unterstützter Übungen hin. [11] Die Arbeit implementiert zwei Phasen. In der ersten Phase wird energiesparend entschieden, ob eine Trainingsaktivität vorliegt. Ist dies der Fall, wird die zweite Phase gestartet, in der das Krafttraining überwacht wird. Als Sensor wird der Gravitationssensor verwendet. Zur Erkennung von Übungsabschnitten wird ähnlich wie in RecoFit zunächst eine Autokorrelation eingesetzt. Allerdings werden anstatt die aktuellen Zeitfenster mit vordefinierten Vorlagen pro Übung zu vergleichen, die Fenster mit dem vorhergehenden Fenster verglichen, um repetitive Muster zu erkennen. Da der beschriebene Ansatz eine hohe Rechenlast erzeugt, stellen die Autoren einen alternativen Ansatz zur Erkennung von sich wiederholenden Übungen vor, der deutlich weniger Berechnungen benötigt. Eine genauere Erklärung des Algorithmus siehe Abschnitt 6.2.

Um Wiederholungen zu erkennen, wird ein Zählen der Spitzen auf den rohen Gravitationsdaten durchgeführt. Zur Erkennung der Übung wird eine SVM verwendet (siehe Abschnitt 6.4). Wenn keine Übereinstimmung gefunden wird, wird dies als neue Übung definiert und der Nutzer wird nach einer Bezeichnung für die Übung gefragt.

Insgesamt wurde mit dem 'Rivisit'-Ansatz eine Genauigkeit von 95,7 % erreicht. Dies ist um 0,2 % besser als der deutlich rechenaufwendigere Autokorrelationsansatz. Die Wiederholungserkennung hat pro Satz eine Fehlerrate von circa 1,1. Die durchschnittliche Satzlänge betrug 9,65 Wiederholungen. Diese Fehlerrate ist auf den Teil der Übungen zurückzuführen, die kaum Änderungen am Gravitationssignal verursachen. Dies liegt daran, dass manche Übungen wie Pec Flys (Siehe Abbildung 9) zwar ausladende Armbewegungen beinhalten, diese sich jedoch senkrecht zum Boden befinden und somit kaum Auswirkungen auf das Gravitationssignal haben. Die Genauigkeit der Übungserkennung beträgt 89,71 Prozent.

Stärken und Schwächen

Eine der größten Stärken von MiLift ist die Nutzerunabhängigkeit bei der Segmenterkennung und der Wiederholungszählung. Es ist jedoch anzumerken, dass die neu erkannten Übungen nicht automatisch in das System integriert werden

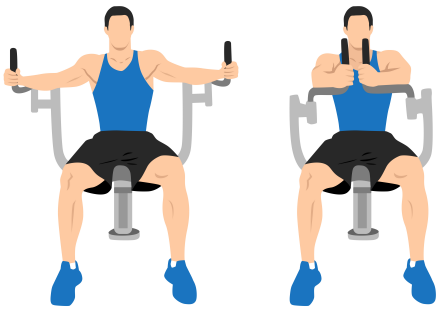


Abbildung 9: Übung Pec Flys

und Teilnehmer daher erneut über die externe Anwendung die Übung und Wiederholungen markieren müssen, um das System mit neuen Übungen zu erweitern. Die Fehlerrate bei der Wiederholungserkennung ist sehr hoch. Die Verwendung des Gravitationssignals erlaubt zwar ein konsistenteres Signal, hat aber nachweislich Probleme beim Zählen der Wiederholungen von Bewegungen, die parallel zum Boden ausgeführt werden.

5.4 Erkennung und repetitives Zählen von komplexen physischen Übungen mit Deep Learning

Übersicht

Soro et al. konzentrieren sich auf die Erkennung von 10 CrossFit-Übungen. Dabei wurden zwei Smartwatches eingesetzt. [13]

Tests haben gezeigt, dass dieses System auch mit nur einer Smartwatch verwendet werden kann. Bei Verwendung einer einzigen Smartwatch wurde eine Erkennungsrate von 98,91 % erzielt. Die Wiederholungen wurden in 91 % der Fälle mit einer Abweichung von maximal Eins erkannt. Für die Datenerfassung wurde ein begrenztes Training verwendet, bei dem die Startzeiten einer Wiederholung durch Vibration signalisiert wurden. Dadurch konnte eine größere Gruppe von Testnutzern erreicht werden. Die Rohdaten wurden mithilfe eines neuronalen Netzwerks genutzt, um die Übungen zu erkennen. Zur Zählung der Übungen wurden separate neuronale Netzwerke verwendet. Eine korrekte Erkennung der Übung ist hierfür jedoch notwendig. Es wurde festgestellt, dass der Beschleunigungssensor die meisten Informationen liefert, das Gyroskop die Genauigkeit weiter erhöhen kann und die Orientierung keinen Einfluss auf die Ergebnisse hat.

Stärken und Schwächen

Der Einsatz des beschränkten Trainings erscheint sinnvoll, um den Kreis der Tester zu erweitern. Eine Schwäche besteht darin, dass die Methode zum Zählen vollständig von der korrekten Erkennung der Übung abhängt. Wenn eine Übung falsch erkannt wird, wird ein falsches neuronales Netzwerk verwendet, was die Wiederholungserkennung erheblich beeinträchtigen kann. Obwohl bei den verwendeten 10 Übungen eine sehr hohe Erkennungsrate von etwa 99 % erreicht wurde, ist dies bei vielen Übungen deutlich unwahrscheinlicher. Um das System zu erweitern, müssen erneut neue Datensätze gesammelt werden.

5.5 Beschleunigungsensor-basierter Algorithmus zur Segmentierung und Erkennung von Repetitiven Bewegungen während dem Training

Diese Arbeit von Džaja et al. bietet grundsätzlich eine einfache Erweiterbarkeit um neue Übungen. Allerdings wurden nur 85,7 % der Wiederholungen korrekt erkannt. [15] Wenn man die Erfolgsrate des Kandidaten, der die Vorlagen zur Erkennung mit dem DTW-Algorithmus erstellte, herausrechnet, ergibt sich eine Erkennungsrate von nur noch 78,53 % für die anderen drei Kandidaten. Es wurden drei Sensoren an unterschiedlichen Körperstellen verwendet, was diesen Ansatz für den alltäglichen Gebrauch zusätzlich unbrauchbar macht.

Wenn man nur die Ergebnisse des Kandidaten betrachtet, der die Vorlagen erstellt hat, ergibt sich eine Rate von 95,8 %. Dies legt nahe, dass separate Vorlagen pro Nutzer die Ergebnisse erheblich verbessern können.

5.6 GymApp

GymApp verwendet ebenfalls den DTW-Algorithmus und ist dynamisch erweiterbar. Die Anwendung baut darauf auf, dass die Nutzer ihre eigenen Vorlagen pro Übung erstellen und diese dann zur Erkennung nutzen. Was diese Arbeit von anderen abhebt, ist die Verwendung von Quaternionen zur Verhinderung des Gimbal-Lock-Problems bei den Rotationssignalen. Die korrekt ausgeführten Wiederholungen wurden bei neun Teilnehmern mit einer Wahrscheinlichkeit von 92 bis 98 Prozent sicher erkannt. [16]

6 Stand der Forschung

Bisherige Arbeiten haben verschiedene Ansätze verfolgt, die im Folgenden separat betrachtet werden. Die Aufgabenstellung lässt sich in vier Hauptaufgaben unterteilen.

6.1 Auswahl der Signale

Im Ersten Schritt muss die Auswahl der Eingangsignale und die richtige Kombination dieser ausgewählt werden.

Beschleunigungssensor und Gyroskop

Nahezu alle betrachteten Arbeiten verwenden als Eingangsdaten den Beschleunigungssensor und den Gyroskopsensor. [1], [10], [13], [17]–[20]

Taewoong et al. unterscheiden zusätzlich zwischen den Beschleunigungsdaten im lokalen Raum und im globalen Raum. [12] Ishii et al. und Džaja et al. nutzen nur den Beschleunigungssensor. [15], [20] Viana et al. nutzen in Ihrer Arbeit nur den Beschleunigungssensor und zusätzlich den Rotation Vector. [16]

Gravitationssensor

Als Eingangsignal nutzen Shen et al. nur den Gravitationssensor der die Daten von Beschleunigungsensor und das Gyroskop zusammenfasst. [11] Die Autoren von LEAN nutzen zusätzlich zu Rotation und Beschleunigung die Gravitation. [18]

Pulssensor

Maheedhar et al. nutzen den Pulssensor um die Muskelkraft der Personen zu verfolgen. [17]

6.2 Auswahl der Abschnitte zur Segmentierung

Im zweiten Schritt müssen die Sensordaten in Abschnitte unterteilt werden, um eine Weiterverarbeitung zu ermöglichen. Dabei müssen Pausen zwischen den Sätzen sowie Änderungen von Einstellungen oder der Wechsel zu anderen Geräten gefiltert werden.

Sliding-Window

Die meistverwendete Methode ist der Sliding-Window-Ansatz mit einem überlappenden Fenster. Dabei werden die eingehenden Sensordaten in gleichmäßige Zeitfenster unterteilt. Die Arbeit ExerSense verwendet 0.25 Sekunden Zeitfenster zur Erkennung von Spitzen. [20]

Zhang et al. und Coates et al. nutzen ein Zeitfenster von 4 Sekunden mit einer Überschneidung von 50 Prozent. [1], [18] Die Autoren von RecoFit und StayFit verwenden ein 5 Sekunden Zeitfenster das sich 4.8 Sekunden beziehungsweise 4 Sekunden mit dem vorhergehen Fenster überschneidet. [10], [17] Die durchschnittliche Wiederholungsdauer pro Übung anhand der gesammelten Trainingsdaten wurde von Mortazavi et al. manuell berechnet. [19] Jedes Fenster wird dabei einzeln überprüft, indem Blöcke dieser Größe aus den eingehenden Daten extrahiert

werden. Ebenfalls übungsabhängige Zeitfenster werden von Soro et al genutzt und zwischen 1 und 10 Sekunden festgelegt. [13]

Rivisit-based Algorithmus

Im Rahmen des MiLift-Papers wird ein Autokorrelationsalgorithmus mit einem Algorithmus namens 'Revisit-based Approach' verglichen. Dieser neue Ansatz erfordert deutlich weniger Berechnungen und ermöglicht es, wiederkehrende Muster in einem Signalstrom zu erkennen. [11]

Der 'Revisit-based'-Algorithmus basiert auf diskretisierten Sensordaten sowie einer Hashmap. Die Hashmap verwendet die diskretisiert Sensordaten als Schlüssel und die aktuelle Zeit als Wert. Anschließend wird die Anzahl der Hashkollisionen der vergangenen Sekunde genutzt, um die 'Rivisit-rate' zu erkennen. Es werden nur Kollisionen gezählt, bei denen der Zeitabstand geringer als 6 Sekunden war. Überschreitet diese 'Rivisit-rate' einen Grenzwert, wird dies als sich wiederholendes Muster erkannt. Wenn der Sensor nicht bewegt wird, führt dies zu einem konstant hohen Wert. Um dieses Problem zu lösen, werden alle eingehenden Werte, die den gleichen diskretisierten Wert wie der unmittelbare Vorgänger haben, verworfen.

Frequenzbasiert

In der Arbeit von Džaja et al. basiert die Erkennung der Abschnitte auf einer erkannten gleichbleibenden Frequenz. Dabei wird die beste Frequenz erkannt und gefiltert. Solange diese Signalfrequenz besteht, wird dies als ein Segment festgelegt. Die beste Frequenz entspricht dabei nicht unbedingt der Frequenz mit der maximalen Amplitude. [15]

6.3 Auswahl der Ansätze zum Zählen

Machine Learning

Soro et al. verwenden separate neuronale Netzwerke für jeden Übungstyp. Die Netzwerke wurden so entworfen, dass der Start einer Wiederholung innerhalb des übergebenen Zeitfensters erkannt wurde. Nach einer Filterung konnten die zusammenhängenden Start-Erkennungen gezählt werden, um die Anzahl der Wiederholungen zu ermitteln. In 73 Prozent der Fälle wurde die Anzahl exakt erkannt und in 91 Prozent der Fälle mit einer maximalen Abweichung von einer Wiederholung. [13]

Die Modelle, die in der Untersuchung von Mortazavi et al. entwickelt wurden, zielen darauf ab, exakt eine Durchführung einer Übung zu identifizieren. Sobald eine Wiederholung erkannt wird, erfolgt eine Erhöhung der Wiederholungsanzahl. Zur Vermeidung von Doppelzählungen wird das betrachtete Fenster am Ende jeder erkannten Wiederholung positioniert. [19]

In der Studie von PUSH wurde ein Datensatz mit über 400.000 Übungsdurchführungen verwendet, um ein neuronales Netzwerk zu trainieren, das die Anzahl der Wiederholungen erkennt. Die Genauigkeit der Identifikation von Datensätzen mit nur einer Durchführung betrug 42,32 Prozent, während sie bei Datensätzen mit 20 Durchführungen 97,23 Prozent erreichte. [12]

Zählen der Spitzen

Die Autoren von MiLift verwenden, nach Erkennung der Sätze, die Achse mit der größten Differenz zwischen Maximum und Minimum. Die Entscheidung, ob die Spitzen oder die Täler gezählt werden hängt davon ab, welches der beiden im Durchschnitt innerhalb eines kleinen Zeitdeltas um den Extrempunkt den größeren vertikalen Unterschied aufweist. Anschließend wird die Anzahl ermittelt. Die durchschnittliche Fehlerrate beim Zählen beträgt 1,12 Wiederholungen bei durchschnittlich 9,65 Wiederholungen. [11] Die Achsen werden in RecoFit in ein eindimensionales Signal umgewandelt, von dem der Durchschnitt abgezogen wird. Anschließend wird ein Filter angewendet, um sehr hohe und niedrige Frequenzen zu entfernen. Alle lokalen Maxima werden gesammelt und zeitlich sortiert. Maxima, die innerhalb einer zeitlichen Untergrenze nach einem bereits akzeptierten Maxima auftreten, werden entfernt. Um weitere falsche Erkennungen zu entfernen, wird in dem Zeitintervall zwischen minimaler und maximaler Wiederholungsdauer um eine Spitze die Autokorrelation berechnet. Der Abstand zum Wert mit der höchsten Autokorrelation wird als Wiederholungsdauer festgelegt. Alle Werte, die näher als 0,75 mal der Wiederholungsdauer an einer akzeptierten Wiederholung liegen, werden entfernt. Im letzten Schritt wird die Spitze gesucht, die an der 40-Prozent-Grenze aller Ausschläge liegt. Alle Spitzen, die kleiner als die Hälfte dieser Spitze sind, werden entfernt und die verbleibenden Spitzen gezählt. Die Zählgenauigkeit wird mit 70 Prozent exakter Erkennung auf den Trainingsdaten angegeben und mit 93 Prozent bei maximal einer Abweichung von einer Wiederholung. [10] In LEAN wird die Achse mit der größten Varianz verwendet. Zur Erkennung werden die letzten drei Wendepunkte der Achse verwendet. Wenn die vertikalen Werte der Wendepunkte den Grenzwert überschreiten, wird dies als Wiederholung gezählt und die ersten beiden Wendepunkte werden aus der Liste entfernt. Der Grenzwert wird aus dem Durchschnitt zwischen dem maximalen und minimalen Wert der Achse berechnet. Von 9 Sätzen wurden 2 aufgrund eines Fehlers nicht richtig erkannt. [18] Maheedhar et al. bereinigen das Signal mit einem Butterworth-Filter. Anschließend werden alle Extrempunkte, die eine geringere Magnitude als der Durchschnitt haben, mit einem Faktor ausgefiltert. Danach wird erneut gefiltert und die Anzahl der Wiederholungen gezählt. Laut Angaben wurde die Anzahl der Wiederholungen in mehr als 95 Prozent der Fälle richtig erkannt. Es wurde keine detaillierte Beschreibung der Datenerhebung gegeben. [17]

Sich anpassender Grenzwert

Anstatt die Spitzen zu zählen, werden hier die Schnittpunkte des Signals mit einem dynamischen Grenzwert gezählt. Es müssen mehrere Bedingungen an diesen Schnittpunkten erfüllt sein um als gültige Wiederholung zu zählen. Die Erkennungsrate für Wiederholungen beträgt 98 Prozent. Der Algorithmus ist im Detail in Unterabschnitt 4.3 beschrieben.

DTW Algorithmus

In mehreren Arbeiten wird jede einzelne Wiederholung durch eine Vorlage erkannt. [15], [16], [20] Die Übungsvorlage wird initial mit einer Ausführung der Übung erstellt und die einzelnen Segmente werden mithilfe des DTW-Algorithmus verglichen. Wenn eine hohe Übereinstimmung erkannt wird, wird das Segment als Wiederholung gezählt. Um die Anzahl der Wiederholungen festzulegen, werden die separat erkannten Übungen gezählt.

ExerSense gibt keine expliziten Auskünfte über die Erkennungsgenauigkeit der Wiederholungen. Allerdings wird die Übungserkennung mit einer Genauigkeit von 97 Prozent angegeben. [20] Džaja et al. erreichten im Schnitt eine Zählgenauigkeit von 85,7 Prozent. Der Teilnehmer, der die Vorlagen aufzeichnete, hatte das mit Abstand beste Ergebnis von 95,8 Prozent. [15] GymApp erkennt einzelne saubere Wiederholungen mit einer Genauigkeit von 91,53 bis 97,41 Prozent. [16]

6.4 Auswahl Ansätze zum Erkennen der Übung

Algorithmischer Ansatz

In den Arbeiten von Džaja et al. und Ishii et al. wird der DTW-Algorithmus genutzt, um das aktuelle Segment mit allen gespeicherten Vorlagen zu vergleichen. [15], [20] Wenn der Wert unter eine Grenze fällt, wird der Abschnitt als Übung gekennzeichnet. Dabei wurde eine Übungerkennungsgenauigkeit von 95 Prozent und 85,7 Prozent erreicht. Die geringere Prozentrate der zweiten Arbeit lässt sich auf eine Übung zurückführen, die häufig mit einer anderen, sehr ähnlichen Übung verwechselt wurde. Ohne diese Übung beträgt die Genauigkeit etwa 91,5 Prozent.

In GymApp wird ebenfalls der DTW-Algorithmus genutzt, um saubere Wiederholungen mit einer Genauigkeit zwischen 92 und 98 Prozent zu erkennen. Die Anzahl der Übungen ist hier nicht beschränkt, da der Nutzer zuvor pro Übung sein eigenes Template aufzeichnet. [16]

Machine Learning

SVM

MiLift verwendet einen SVM-Classifier, der bei 15 Übungen eine Genauigkeit von ca. 90 Prozent erzielt hat. In RecoFit werden mehrere binäre SVMs trainiert

und das Ergebnis der SVM mit der höchsten Übereinstimmung genutzt. [10] Die Berechnungen erfolgen für jedes Fenster. Die verwendete Methode sieht vor, dass bei Online-Anwendungen die bereits in den ersten drei Sekunden erkannte Übung beibehalten wird, falls die Gewichtung zugunsten einer anderen Übung überwiegt. Laut Zhang et al. ist eine SVM aufgrund der Erkennungsrate von 96,3 Prozent gegenüber Entscheidungsbäumen und neuronalen Netzwerken zu bevorzugen. [1]

CNN

In der Arbeit von Gjoreski et al. wurde die Performance von verschiedenen Machine Learning Modellen zur Bewegungserkennung verglichen. Dabei schnitten Deep Learning Convolutional Neural Networks am besten ab. Es wurde auch festgestellt, dass der Ansatz mit Random Forest mit einem kleinen Datensatz genauer arbeitet. Mit einem ausreichend großen Datensatz funktioniert jedoch ein CNN besser. [21] Soro et al. verwenden ebenfalls ein CNN, welches aus einer Reihe von 2D Faltungsschichten und zwei vollständig verbundenen Schichten besteht. Die Feineinstellungen wurden mit einer Gittersuche nach Hyperparametern durchgeführt. Insgesamt wurden die Übungen mit einer Genauigkeit von 99,96 Prozent erkannt. Es wurden dabei zwei Smartwatches und zehn Übungen verwendet, wobei die Wiederholungen durch vorgegebene Zeitintervalle per Vibration signalisiert wurden. [13] Ähnlich nutzt PUSH ein CNN mit zwei Faltungsschichten und einer voll verbundenen Schicht mit 1024 Knoten. Es wurde sich gegen 'Recurrent Neural Networks' entschieden. Insgesamt wurde eine Genauigkeit von 92,14 Prozent über 50 Übungen erreicht. [12]

Da sich Zhuang et al. auf allgemeine Sportarten konzentriert, wurde zwischen nicht-periodischen und schwach periodischen Sportarten unterschieden. Fitnessstudio-Übungen sind periodisch, daher ist nur der Abschnitt mit den schwach periodischen relevant. Die Arbeit vergleicht CNN, K-Nearest-Neighbor (KNN), Naive Bayes, Random Forest und SVMs. Aufgrund der besten Ergebnisse wurde ein CNN gewählt. Es wurde zwischen vier verschiedenen Schwimmarten unterschieden. Insgesamt wurde eine Genauigkeit von 93,99 Prozent erreicht. Um die Ergebnisse weiter zu verbessern, wurde ein periodischer Matching-Algorithmus genutzt, der auf Autokorrelation basiert. Wenn keine Übereinstimmung gefunden wird, werden konstante Zeitabstände für das CNN verwendet. Da die Abschnittslänge variiert, wird eine bilineare Interpolation verwendet, um auf die für die vollständig verbundenen Schichten benötigte konstante Länge zu kommen. Durch die periodische Erkennung konnte die Erkennungsrate um 3,89 Prozent auf 97,88 Prozent gesteigert werden. Die oben genannten anderen Machine-Learning-Ansätze wurden ebenfalls verglichen. Durch die periodische Erkennung konnte die Erkennungsrate im Durchschnitt um 3,78 Prozent gesteigert werden. [22]

In ihrer Arbeit nutzten Sivakumar et al. ebenfalls ein online CNN zur Klassifizierung von 19 Übungen und erreichten insgesamt eine Genauigkeit von 82,29 Prozent. [23] Nils et al. empfehlen auch die Verwendung von CNNs zur Erkennung von physischen Aktivitäten empfohlen. [24]

Andere Ansätze

StayFit arbeitet mit einem Künstlichen Neuronalen Netzwerk (ANN). Die Übungen wurden mit einer Genauigkeit von etwa 96 Prozent bei acht Übungen erkannt. [17]

Die Autoren von LEAN verwenden Gradient-Boosted-Classifier, die ähnlich wie Random Forest mehrere Entscheidungsbäume kombinieren, um ein genaueres Modell zu erhalten. Für jede der drei betrachteten Übungen wurden separate Modelle erstellt, welche zwischen guter und schlechter Ausführung unterscheiden können. Die Erkennungsrate beträgt durchschnittlich 98 Prozent für alle drei Übungen. [18]

7 Begründetes Vorgehen dieser Arbeit

7.1 Auswahl der Sensoren und berechnete Features

Im Folgenden werden die Sensoren festgelegt, die zur Erkennung der Übungen und zum Zählen der Wiederholungen genutzt werden.

Sensoren

Die am häufigsten verwendeten Sensoren zur Bewegungserkennung im Kontext der Wiederholungserkennung sind eine Kombination aus Beschleunigungssensor und Gyroskop [1], [10], [13], [17]–[19]. Die Arbeit von Mortazavi et al. setzte sich mit der Frage auseinander, welcher Sensor und welche Achse am besten geeignet sind. Dabei wurden nur der Beschleunigungssensor und das Gyroskop einbezogen. Insgesamt wurde eine übungsspezifische optimale Achse ermittelt. Als bestes Signal wurde der Beschleunigungssensor festgelegt. Es wurde jedoch angemerkt, dass das genaueste Ergebnis mit einer Kombination aus beiden Sensoren erzielt werden kann. [19] Aufgrund dieser Erkenntnisse sollen in dieser Arbeit sowohl das Gyroskop als auch der Beschleunigungssensor verwendet werden.

Der Gravitationssensor kombiniert lediglich die Daten des Beschleunigungssensors und des Gyroskops. Zudem gibt es laut Shen et al. Schwierigkeiten bei der Erkennung von Übungen, die parallele Bewegungen zum Boden erfordern. [11] Aufgrund dieser bekannten Probleme wird der Gravitationssensor nicht verwendet.

7.1.1 Berechnete Sensorwerte

Rotationsvektoren

Rotationsvektoren wurden bereits in den Quellen [13], [16] verwendet. Sie geben Aufschluss über die Rotation im globalen Raum, da die Werte des Magnetometers mit einbezogen werden. In einer anderen Arbeit wurde eine Untersuchung der verschiedenen Kombinationsmöglichkeiten der Sensoren durchgeführt, die ergab, dass dieser Orientierungssensor die Performance des Systems weder verbessern

noch verschlechtern konnte. [13] Aus diesem Grund sollten diese Sensorwerte nicht verwendet werden.

Winkelgeschwindigkeit

Maheedhar et al. berechnen den Betrag der Winkelgeschwindigkeit sowie den Anteil davon in der YZ-Ebene. Diese Ebene wurde aufgrund der betrachteten Übungen verwendet. Dieser Wert ist sinnvoll, erfordert jedoch zusätzliche Berechnungen. Wenn andere Werte, die ohne zusätzlichen Aufwand ausreichend sind, kann auf diesen Wert verzichtet werden. [17]

7.1.2 Zusammenfassen der Achsen pro Sensor

Es gibt verschiedene Ansätze, um die Achsen jedes Sensors in eine einzelne Achse umzuwandeln und eine einfachere Weiterverarbeitung zu ermöglichen.

Hauptachse anhand der größten Integrationsfläche

Um den Zählalgorithmus zu verbessern, wird von Zhang et al. die Hauptachse ausgewählt. Diese wird anhand der Achse mit der größten Integrationsfläche unter der Signalkurve jedes Fensters bestimmt. Dadurch wird die Qualität des Signals nicht durch die Verwendung irrelevanter Achsen verschlechtert und ein dynamischer Wechsel der Achse wird ermöglicht. Um zwischen zwei sehr ähnlichen Achsen nicht ständig wechseln zu müssen, muss der Wert für eine bestimmte Zeit konstant auf einer anderen Achse liegen, bevor ein Wechsel erfolgt. [1] Dies führt zu einer Verzögerung bei der Nutzung der idealen Achse, dadurch ist dieser Ansatz nicht ideal und sollte vermieden werden.

Beschleunigungsmagnitude

Mehrer Arbeiten berechnet die Magnitude des Beschleunigungssensors. Diese fasst die drei Achsen zu einem Wert zusammen, der die Gesamtrichtung der Bewegung angibt. [10], [15], [17] Das Zusammenfassen von Informationen führt zu einem Informationsverlust, weshalb dieser Ansatz als nicht ideal betrachtet wird.

Hauptkomponentenanalyse

RecoFit geht davon aus, dass nur die X-Achse der Sensoren fixiert ist und sich parallel zum Unterarm befindet. Zur Übungserkennung werden die 3D-Punkte des Beschleunigungsensors entlang der Achse projiziert, auf der sich die meisten Punkte befinden. Dies wird mithilfe der Hauptkomponentenanalyse (engl. Principal Component Analysis (PCO)) ermöglicht. Ein Beispiel für eine solche Projektion ist unter Abbildung 10 zu sehen. Diese Achse wird aufgrund ihrer Eigenschaften als die interessanteste Achse betrachtet. Die gleiche Berechnung wird erneut nur für die Y- und Z-Achse durchgeführt, da dies die Bewegungen senkrecht zum Arm darstellt. Sowohl für den Beschleunigungssensor als auch für das Gyroskop werden beide Hauptkomponentenanalysen durchgeführt. Morris et al. machen keine expliziten Angaben dazu, auf welcher Hardware die Daten mit welcher Performance verarbeitet werden. Die Ausstattung zur Erkennung besteht aus

Sensoren sowie der nötigen Ausstattung zur Übertragung per Bluetooth. Daher wird angenommen, dass die Berechnungen ausschließlich auf dem PC durchgeführt werden, an den die Daten übertragen werden. [10] Die Berechnungen für die Hauptkomponentenanalyse sind sehr rechenintensiv. Aus diesem Grund wird dieses Verfahren zur Erkennung in dieser Arbeit nicht verwendet.

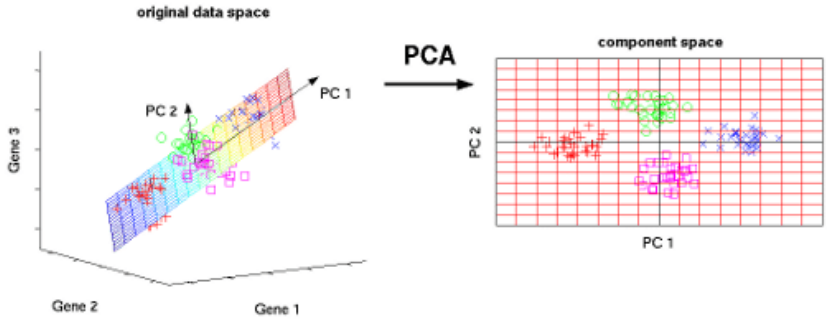


Abbildung 10: Umwandlung von 3D-Daten in 2D mithilfe der Hauptkomponentenanalyse (Quelle: [25])

7.2 Ansatz Abschnittsunterteilung

Sliding-Window

Sliding-Window bezeichnet ein Verfahren, bei dem ein Fenster mit einer zuvor definierten Länge schrittweise über einen Datensatz bewegt wird. Diese Abschnitte können dann für den Vergleich mit einem Muster verwendet werden. Der Sliding-Window Ansatz ist am häufigsten zur Erkennung der Trainingsabschnitte verwendet. Wie in Sliding-Window aufgelistet, verwendet jeder der genannten Ansätze eine andere Fenster- und Überschneidungsgröße. Für jedes dieser Fenster wird dann entschieden, ob dort eine Übung enthalten ist. Um eine dynamische Erweiterbarkeit der Übungen zu ermöglichen, sind individuelle Größen pro Übung, wie in anderen Arbeiten, nicht möglich. [13], [19]. Es ist nicht möglich zu entscheiden, welche Fenstergrößen und Überschneidungen der betrachteten Arbeiten die beste darstellen. [1], [10], [17], [18], [20] Die Sensordaten werden kontinuierlich gesammelt, in Fenster aufgeteilt und pro Fenster berechnet, was zu einer hohen Rechenlast auf dem Gerät führt. Aufgrund dieser beiden Schwächen wird dieser Ansatz nicht verwendet.

Frequenzbasiert

Die frequenzbasierte Erkennung von Džaja et al. nutzt die Extrempunkte eines Wertes, um zwischen Segmenten abhängig von der Übung zu unterscheiden. [15] Obwohl dies bei den 9 betrachteten Übungen einfach umzusetzen ist, ist es bei einer dynamischen Erweiterbarkeit nicht möglich. Aus diesem Grund wird dieser Ansatz in dieser Arbeit nicht verwendet.

Rivisit-based Algorithmus

Der in Rivisit-based Algorithmus vorgestellte Algorithmus weist keine Schwächen der anderen aufgeführten Ansätze auf. Die eingehenden Daten werden kontinuierlich betrachtet und komplexere Berechnungen werden erst ausgeführt, sobald ein wiederkehrendes Muster erkannt wurde. Dadurch kann die Anzahl der benötigten Berechnungen gering gehalten werden.

7.3 Ansatz Übungserkennung

In dieser Arbeit wird die automatische Erkennung von Übungen nicht als Hauptkriterium betrachtet. Eine automatische Übungserkennung könnte zu falschen Erkennungen führen. Wenn dann Wiederholungen zu der falschen Übung eingetragen werden, müssen diese aus der Übung gelöscht, zur richtigen Übung gewechselt und dort anschließend die korrekte Wiederholungsanzahl erfasst werden. Fehleinschätzungen bezüglich der Übung könnten dazu führen, dass der Nutzer im schlimmsten Fall nach jedem Satz oder jeder Wiederholung die Daten anpassen muss. Aus diesem Grund ist der Ansatz dieser Arbeit darauf ausgerichtet, dass der Nutzer manuell die aktuelle Übung auswählt. Um den Benutzer zu warnen, falls er vergisst, die Übung auszuwählen, wird bei einer hohen Fehlerquote eine Warnung ausgegeben, um den Benutzer darauf hinzuweisen, dass eine andere Übung wahrscheinlicher ist. Es gibt zwei verschiedene Ansätze zur Erkennung der Übung.

Klassischer Ansatz

In vorangegangenen Arbeiten hat der klassische Ansatz mit dem DTW eine gute Erkennungsrate von 85,7 bis 98 Prozent erzielt. Es wird lediglich eine Vorlage einer einzelnen Wiederholung benötigt. Džaja et al. geben eine Erkennungsrate von insgesamt 85,7 Prozent nach den vier Teilnehmern aufgeschlüsselt an. Der Teilnehmer, der die Vorlagen aufzeichnete, erzielte mit Abstand das beste Ergebnis von 95,8 Prozent. Die anderen Teilnehmer, die lediglich die Vorlagen verwendeten, erzielten nur eine Erkennungsrate von 71,1 bis 85,4 Prozent. [15] Daher wird dieser Ansatz als Ausgangslage verwendet, um den Nutzern sofort ein Ergebnis liefern zu können, ohne dass Trainingsdaten benötigt werden. Um genauere Ergebnisse zu erzielen, sollte jeder Nutzer bei der ersten Verwendung der Übung eine eigene Vorlage erstellen.

Machine Learning Ansatz

Die Verwendung von Machine Learning ist weit verbreitet (Siehe: Abschnitt 6.4). Die häufigsten Ansätze sind SVMs und Convolutional Neural Networks (CNNs). Unterschiedliche Studien kommen zu dem Schluss, dass CNNs am besten geeignet sind (vgl. [21], [24]). Laut Zhuang et al. liefern SVMs und CNNs die genauesten Ergebnisse. [22] Angesichts der Empfehlungen und der breiteren Wissensbasis aus anderen Studien scheint der Einsatz von CNNs präziser zu sein. Der Machine-

Learning-Ansatz hat jedoch den großen Nachteil, dass vor der Benutzung eine große Anzahl markierter Datensätze vorhanden sein muss.

Der klassische Ansatz scheint zwar für den Einsatz mit einer kleinen Anzahl an Übungen sinnvoll zu sein, benötigt aber bei sehr vielen Übungen aufgrund der hohen Anzahl an Mustern, die verglichen werden müssen, viele Berechnungen. Für eine hohe Anzahl an Übungen ist somit vermutlich ein CNN am besten geeignet. Dies setzt jedoch voraus, dass Datensätze zuvor markiert wurden, was eine einfache Erweiterung der Anwendung unmöglich macht. Aus diesem Grund wird zunächst versucht, die Problemstellung durch den klassischen Ansatz zu lösen.

7.4 Ansatz Wiederholungserkennung

Überblick Wiederholungserkennung

Die in Machine Learning beschriebenen Ansätze sind universell einsetzbar und können leicht verschiedene Sensoren einbeziehen. Allerdings sind die angegebenen exakten Zählraten zwischen 42,32 und 73 Prozent eher schlecht.

Von den in Zählen der Spitzen beschriebenen Arbeiten geben nur ungenaue Angaben über die Erkennungsrate und können deshalb nur schwer verglichen werden. Die Arbeit RecoFit erreicht nur eine Rate von 70 Prozent genauer Erkennungen. [10] Die höchsten Werte werden von Stay-Fit mit durchschnittlich über 95 Prozent angegeben. Es werden keine genauen Angaben darüber gemacht, auf welcher Grundlage die Prozentwerte berechnet wurden. [17] Zudem basiert dieser Ansatz auf einer manuellen Auswahl der besten Achse pro Übung, was nicht verallgemeinerbar ist. Daher scheint dieser Ansatz nicht optimal zu sein. Der Algorithmus, der von Zhang et al. verwendet und in Abschnitt 6.3 vorgestellt wird, gibt eine exakte Erkennungsrate von 98 Prozent an. Um Bewegungen auszufiltern, die nicht zu den Wiederholungen zählen, wird erst ab 10 aufeinanderfolgenden gleichen Signalwellen gezählt. Obwohl es nicht explizit erwähnt wird, legt dies nahe, dass Sätze mit weniger als 10 Wiederholungen nicht erkannt werden können. [1]

Der unter DTW Algorithmus vorgestellte Ansatz liefert ebenfalls sehr gute Ergebnisse. Der Nutzer, der die Vorlage erstellt hat, erreichte eine Genauigkeit von 95,8 Prozent (vgl. [15]). Ein weiterer Vorteil dieses Ansatzes ist, dass bereits einzelne Wiederholungen sehr genau erkannt werden können. Die Autoren von GymApp geben die Erkennungsraten pro Wiederholung an, welche zwischen 91,53 und 97,41 Prozent liegen. [16] Ein Nachteil dieses Ansatzes ist, dass jedes Zeitfenster mit der hinterlegten Vorlage verglichen werden muss, was viele Berechnungen erfordert.

Vorgehen Wiederholungserkennung

Während der Counting-Ansatz 6.3 die höchste Genauigkeit angibt und nur relativ wenige Berechnungen benötigt, kann er kleine Wiederholungsbereiche nur schlecht abdecken. Der DTW Algorithmus-Ansatz deckt auch einzelne Wiederho-

lungen sehr gut ab, benötigt allerdings viele Berechnungen. Der in dieser Arbeit verfolgte Ansatz kombiniert beide Varianten, um die Vorteile aus beiden zu nutzen und die Nachteile auszugleichen. Grundsätzlich wird der Sich anpassender Grenzwert genutzt, um potenzielle Wiederholungen zu erkennen. Anschließend wird der DTW Algorithmus verwendet, um den Bereich mit der Vorlage zu vergleichen und die Wiederholung zu bestätigen. Auf diese Weise können viele Berechnungen eingespart werden und gleichzeitig die hohe Genauigkeit des DTW-Algorithmus genutzt werden.

7.5 Zusammenfassung Vorgehen

Wie in Unterabschnitt 7.4 beschrieben, scheint eine Kombination des Algorithmus mit anpassenden Grenzwert und des DTW-Algorithmus am sinnvollsten zum Erkennen der Wiederholungen zu sein, da diese beiden Ansätze die höchsten Erkennungsraten in vorangegangenen Arbeiten gezeigt haben. Eine Kombination gleicht die Schwächen beider Ansätze aus. Außerdem kann, wie in Unterabschnitt 7.3 beschrieben, der DTW-Algorithmus genutzt werden, um zu überprüfen, ob die ausgewählte Übung noch immer aktiv ist. Wenn der Wert der aktuell gewählten Übung zu niedrig ist, kann man die Vorlagen der anderen Übungen aus dem aktuellen Trainingsplan vergleichen und dem Nutzer bei entsprechender Übereinstimmung einen Vorschlag machen, ob er die Übung wechseln möchte.

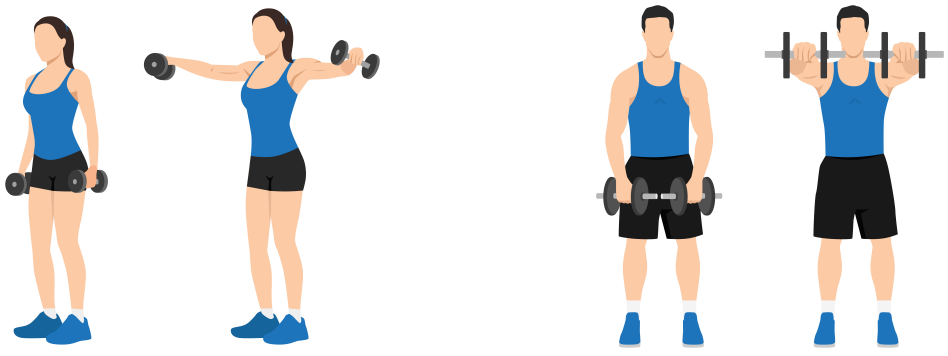
Für die Wiederholungserkennung scheint es sinnvoll, einen Sensor auszuwählen, der nur wenige oder keine Berechnungen benötigt, um seine Werte zu liefern. Bei sehr ähnlichen Übungen haben zusätzliche Informationen, wie beispielsweise die gewählte Achse, wenig Bedeutung. Viel wichtiger ist es, möglichst klare Daten zu haben, um den größtmöglichen Bewegungsradius abzubilden. Somit ist es sinnvoller, eine aussagekräftige Achse auszuwählen, anstatt alle Achsen zu betrachten oder zu vermischen. Dies kann durch die Auswahl der Hauptachse, wie in der Arbeit von Zhang et al., erreicht werden. [1] Zum Beispiel liefert die vertikale Beschleunigung beim Zählen von Wiederholungen wahrscheinlich sehr genaue Informationen für die sehr ähnlichen Übungen Front- und Seitheben (Siehe: Abbildung 11). Eine Entscheidung über die Art der Übung kann bei beiden genannten Übungen nur anhand der vertikalen Beschleunigung nicht getroffen werden, da beide das gleiche Signal liefern. Bei der Erstellung des DTW muss entschieden werden, welche der verarbeiteten oder unverarbeiteten Daten das unterschiedlichste Bewegungsmuster ergibt. Eine Möglichkeit besteht darin, die Achsen zu einem Beschleunigungswert zu kombinieren. [10], [15], [17] Alternativ kann bei ausreichender Rechenleistung eine separate Betrachtung der verschiedenen Achsen erfolgen, gefolgt von einer Kombination dieser. Es muss geprüft werden, ob der in Abschnitt 7.2 vorgeschlagene Rivisit-based Algorithmus Einsparungen bei den Berechnungen erzielen kann, ohne die Erkennungsgenauigkeit des Systems zu stark zu beeinträchtigen. Alternativ kann auch geprüft werden, ob der Ansatz gemäß Abschnitt 6.3 bereits ausreichend Einsparungen ermöglicht.

8 Implementierung

8.1 Auslesen der Sensorwerte

Die zum Testen verwendete Smartwatch verfügt über verschiedene Sensoren, die den aktuellen Sensorwert zum Zeitpunkt der Messung liefern. Der Standardwert für diese Abtastrate liegt bei 200 Millisekunden. Manuelle Auswertungen der aufgezeichneten Datenreihen ergaben keine erkennbaren Ausschläge durch kurze, abgehackte Bewegungen. Solche Bewegungen treten z.B. auf, wenn eine Wiederholung Seitheben (Abbildung 11a) sehr schnell ausgeführt wird. Eine schrittweise Reduktion der Abtastrate ergab eine notwendige Abtastrate von 10 Millisekunden, um auch solche kurzen Bewegungen zu erkennen.

Der Benutzer-Beschleunigungssensor nutzt die Daten des Beschleunigungssensors entfernt aber die Erdbeschleunigung von $9.81\,\mathrm{m/s^2}$. Wie in Abbildung 12 sichtbar ist, zeigen die Werte der X- und Y-Achse des Benutzer-Beschleunigungssensors nur geringe Schwankungen. Bei den Daten des Beschleunigungssensors ist bei Datenpunkt 200 und 600 ein deutlicher Ausschlag zu erkennen. Das stärker variierende Bewegungsmuster ermöglicht eine bessere Erkennung durch den DTW-Algorithmus. Aus diesem Grund wurde der Beschleunigungssensor und nicht der Benutzer-Beschleunigungssensor verwendet.



(a) (b)

Abbildung 11: Beispiel ähnliche Übungen. (11a: Seitheben, 11b: Frontheben)

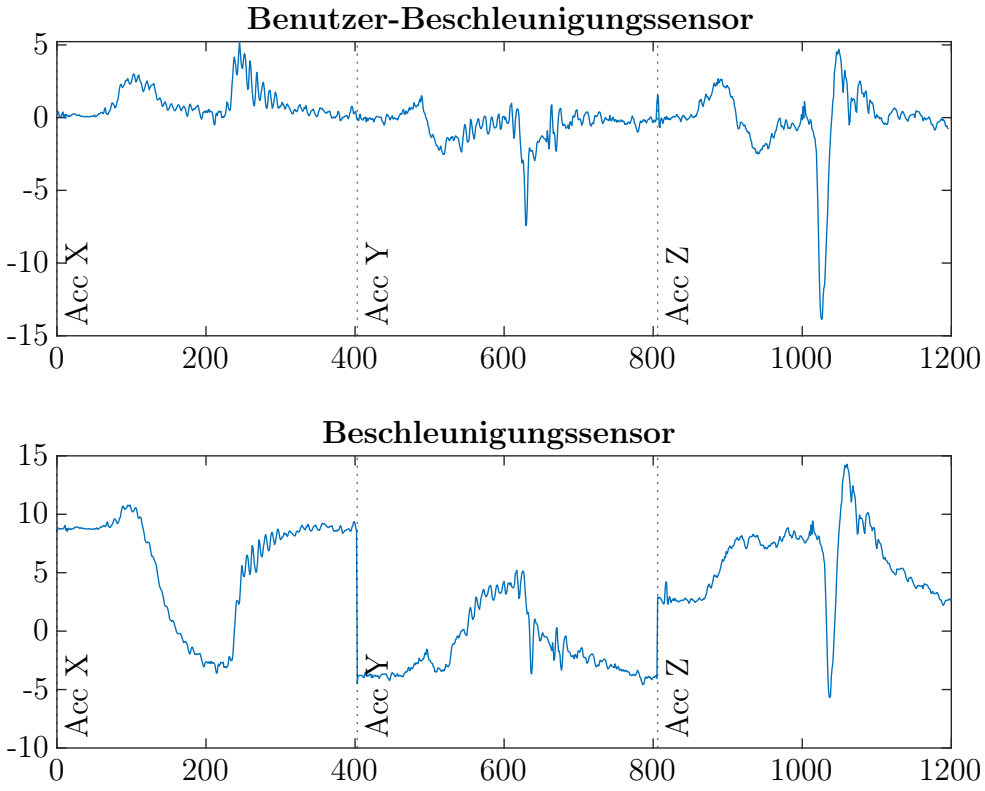


Abbildung 12: Erfasste Daten des Benutzer-Beschleunigungssensors und dem Beschleunigungssensors. Die Daten wurden mit einer einzigen Wiederholung erfasst.

8.2 DTW

8.2.1 Umsetzung DTW

Um den in Unterabschnitt 4.1 beschriebenen Algorithmus zu verwenden, wird eine Vorlage benötigt, mit der verglichen werden kann. Zur Aufzeichnung der Vorlage markierte die Uhr am Handgelenk über Vibration den Start der Ausführung. Nach zwei Sekunden wurde die Position der maximalen Kontraktion oder Dehnung markiert, um den Zeitpunkt zu kennzeichnen, an dem wieder in die Ausgangsposition zurückgekehrt werden sollte. Das Ende der Ausführung wurde ebenfalls durch eine Vibration signalisiert. Das Ergebnis dieser Aufzeichnung für die Übung Seitheben ist in Abbildung 13 zu sehen.

Der Plan, für die Übungserkennung eine einzelne Achse auszuwählen, und diese für den DTW-Algorithmus zu nutzen, musste geändert werden, da er nur schlechte Ergebnisse lieferte und oft die falsche Achse gewählt wurde.

Der Ansatz, die Achse mit der größten Fläche zum jeweiligen Ausgangswert (0-Achse bzw. $9.81\,\mathrm{m/s^2}$ im Fall der Beschleunigungsachse auf der Achse zum

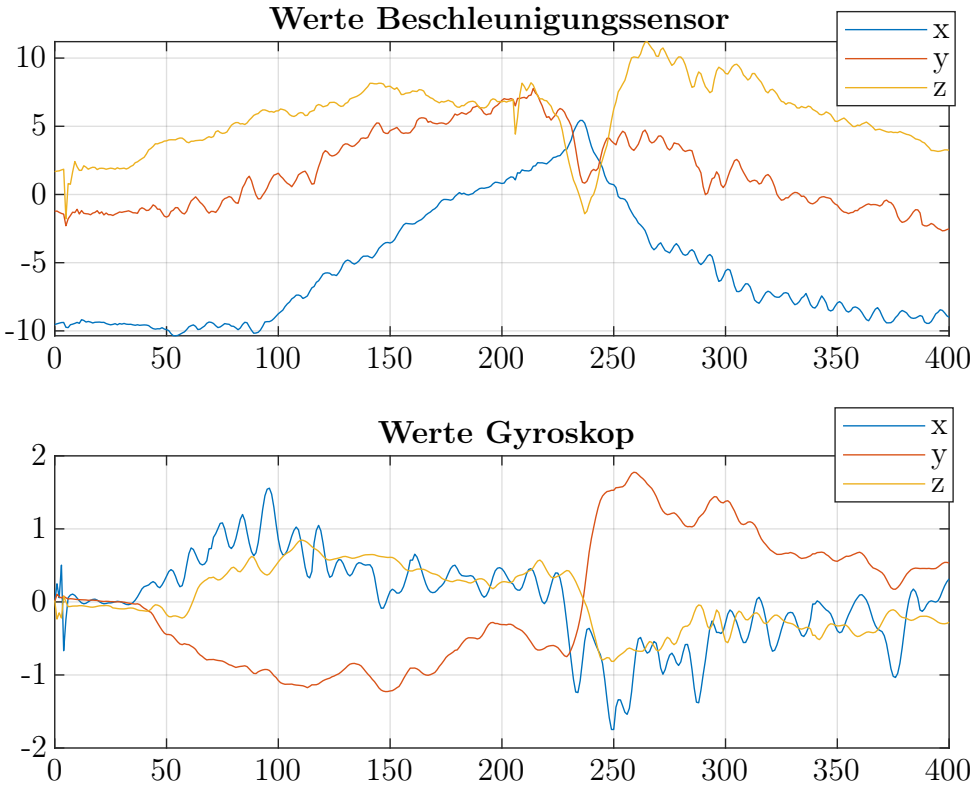


Abbildung 13: Datenreihen einer Wiederholung Seitheben

Boden), ermöglicht keine Vergleichbarkeit zwischen Beschleunigungswerten und Gyroskopwerten. Wie in Abbildung 13 ersichtlich, bewegten sich die Beschleunigungswerte im Bereich von -10 bis 10, während die Gyroskopwerte nur zwischen -2 und 2 lagen.

Das zweimalige Aufzeichnen der Übung und die Berechnung der DTW-Werte pro Achse führten zu Problemen. Oft besitzen Übungen eine oder mehrere Achsen, die keinerlei oder nur geringfügige Schwankungen um den Ausgangswert aufweisen. Aufgrund dieser geringen Schwankungen ergeben diese Achsen die besten DTW-Werte, obwohl sie die geringste Aussagekraft über die Übung besitzen.

Die beschriebenen Probleme konnten gelöst werden, indem für den DTW-Algorithmus alle Achsen genutzt wurden.

Der Ansatz, die DTW-Werte für jede Achse separat zu berechnen und davon die Summe zu nutzen, führt dabei zu dem Problem, dass sich Werte gegenseitig ausgleichen können. Angenommen alle Achsen zweier Übungen sind nahezu null, mit Ausnahme davon, dass die erste Übung einen eindeutigen Ausschlag bei den X-Werten des Beschleunigungssensors zeigt und die zweite Übung einen eindeutigen, gleich starken Ausschlag bei den Y-Werten zeigt. Die Summe der DTW-

Werte bei beiden Übungen würde die gleichen Werte liefern und wäre nicht mehr zu unterscheiden.

Um dies zu vermeiden, wurden alle Sensorwerte miteinander verkettet. Dies ermöglicht es, das genannte Problem zu beheben, da aus dem zusammengefassten Datensatz ein einzelner Wert mit dem DTW-Wert berechnet wurde. Eine solche Verkettung ist in Abbildung 14 zu sehen. Die Daten entsprechen den gleichen Werten wie in Abbildung 13.

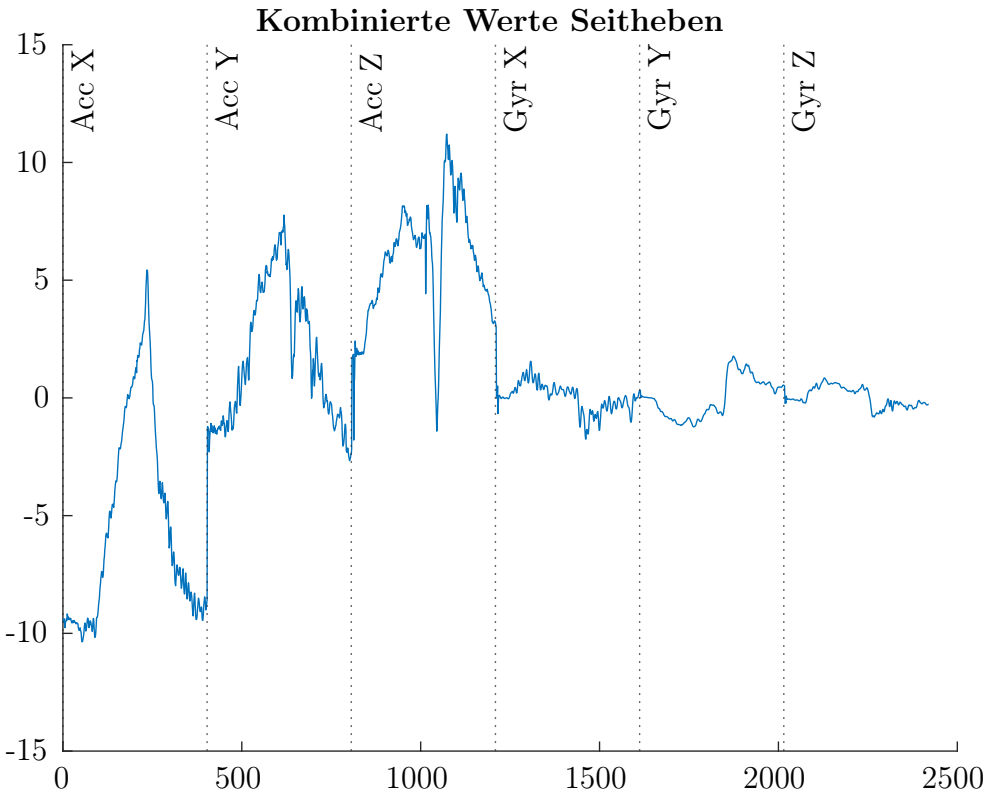


Abbildung 14: Datenreihe einer Wiederholung Seitheben mit kombinierten Werten

8.2.2 Performance DTW

Keine Reaktion des UI Threads

Für die Entwicklung wurde Flutter und Dart verwendet. In diesem Framework werden standardmäßig alle Berechnungen im Frontend-Thread ausgeführt. Aufgrund dessen führten die rechenintensiven Berechnungen des DTW-Algorithmus dazu, dass die Benutzeroberfläche auf Nutzereingaben nicht reagierte und Ladeanimationen keine Bewegung mehr zeigten.

Dieses Problem wurde durch die Verwendung von Isolates gelöst, welche die Berechnungen im Hintergrund durchführten und nur das Rechenergebnis zurücklieferten. Die Kommunikation ist in Listing 3 sichtbar.

Listing 3: Erstellung und Verwendung eines Isolates

```
//Variablen definieren...
Future<void> init() async {
    _isolate = await Isolate.spawn(_backgroundWorker,
        ReceivePort().sendPort);
    events = StreamQueue<dynamic>(receivePort);
    // Die Erste Nachricht die geantwortet wird enthaelt den Port
        zum senden
    sendPort = await events.next;
}

Future<void> _backgroundWorker(SendPort p) async {
    // Senden des Ports zum senden von Nachrichten
    final commandPort = ReceivePort();
    p.send(commandPort.sendPort);
    // Wait for messages from the main isolate.
    await for (final message in commandPort) {
      if (message is DtwParams) {
        //Berechnungen...
        p.send(result);
      }
    }
}

public void main(){
    // Sicherstellen das init() aufgerufen wurde....
    sendPort.send(DtwParams(...));
    double result = await events.next;
}
```

State Managment

Die Verwendung von Isolates ermöglicht eine Weiterverwendung der App während der DTW Berechnung. Die Erstellung eines neuen Isolates führte aber weiterhin zum Einfrieren der Anwendung, da diese aufwendig ist.

Dieses Problem wurde durch die Verwendung eines Providers gelöst, der über den gesamten Lebenszyklus der App das gleiche Isolate zur Verfügung stellt. Somit muss nur beim Starten der Anwendung einmalig das Isolate generiert werden. Die Erstellung und Verwendung des Providers wird in Listing 4 gezeigt. Die Anwendung ist hierarchisch aufgebaut, und der Provider kann jederzeit unterhalb der Stelle, an der er erstellt wurde, abgerufen werden.

Listing 4: Verwendung des Provider Patterns

```
// Erstellung des Providers im Wurzelverzeichnis der Anwendung
// sowie Initialisierung des Isolates
Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => DtwIsolateHelper()..init(),...
}

// Abruf der Instanz in jeder beliebigen Stelle der Anwendung
final helper = Provider.of<DtwIsolateHelper>(context, listen: false);
```

FastDTW

Die Verwendung der genannten Änderungen führte zu einer Verbesserung der Performance. Allerdings war diese allein nicht ausreichend. Wie beschrieben, wurde die Abtastrate auf 10 Millisekunden gesetzt. Die Berechnungen dauerten im Schnitt mehrere Sekunden, was zu einer starken Verzögerung bei der Ausgabe führte.

Durch die Verwendung des Fast DTW Algorithmus konnten weitere Verbesserungen erzielt werden. Die Berechnungsdauer einer möglichen Wiederholung konnte auf circa eine Sekunde gesenkt werden. Allerdings ist dies mit einem neuen Datensatz alle 10 Millisekunden deutlich zu langsam. Um die Ergebnisse in derselben Geschwindigkeit wie die Erfassung neuer Daten zu berechnen, müsste die Performance des Algorithmus um etwa den Faktor 100 verbessert oder die Anzahl der Datenpunkte zur Berechnung reduziert werden.

Aufgrund der begrenzten Rechenleistung der Smartwatch sind Ansätze wie parallele Berechnungen nicht sinnvoll. Obwohl eine Verbesserung wahrscheinlich ist, würde sie nicht in dem benötigten Ausmaß eintreten.

8.3 Sich anpassender Grenzwert

Wie im vorangegangenen Abschnitt beschrieben, reicht der DTW-Algorithmus auch mit Verbesserungen nicht aus, um Wiederholungen in vertretbarer Zeit zu erkennen. Aus diesem Grund wurde der Ansatz von Zhang et al. mit dem Ansatz kombiniert. [1]

Wie in den Grundlagen unter Unterabschnitt 4.3 beschrieben, ist dieser sehr performant, benötigt aber mehrere Wiederholungen, um korrekt zu funktionieren und ist somit nicht ideal für einen geringen Wiederholungsbereich geeignet. Der genau Ablauf der Kombination der beiden Algorithmen ist unter Unterabschnitt 8.4 erklärt.

8.4 Wiederholungserkennung

Um Wiederholungen zu erkennen, wählt der Nutzer eine zuvor angelegte Übung aus und startet das Tracking. Die eingehenden Sensorwerte werden an den Algorithmus für den sich anpassenden Grenzwert übergeben. Zusätzlich werden die Daten in einer Last in - first out (LIFO)-Struktur gespeichert. Um zu vermeiden, dass die Speicherkapazität überlastet wird, wurde eine Datenstruktur gewählt, die eine maximale Größe begrenzen kann und dabei immer die ältesten Daten verwirft. Die Länge der LIFO-Struktur wurde auf 2000 festgelegt, was bei einer eingehenden Datenrate von 10 Millisekunden einem Zeitfenster der letzten 20 Sekunden entspricht. Wenn der Algorithmus eine mögliche Wiederholung erkennt, werden die Datenpunkte um die mögliche Wiederholung aus der Struktur entnommen. Die Anzahl der Datenpunkte entspricht dabei der durchschnittlichen Dauer einer Wiederholung bei der Aufzeichnung. Da bei der Aufzeichnung die Wiederholung in den meisten Fällen langsam und konzentriert ausgeführt wird, ist anzunehmen, dass die Bewegung bei der tatsächlichen Ausführung schneller erfolgt. Es ist daher vorauszusetzen, dass die gesamte Wiederholung in den entnommenen Daten enthalten ist, sofern der Algorithmus die Wiederholung korrekt erkannt hat. Anschließend wird der Datenabschnitt an den DTW-Algorithmus übergeben, der mit den kombinierten Werten einen Wert berechnet. Je niedriger dieser Wert ist, desto größer ist die Übereinstimmung mit der aufgezeichneten Vorlage. Es ist nicht möglich, eine fixe Größe zu definieren, welche Werte als gültige Wiederholung zählen und welche abgelehnt werden, da diese abhängig von der Übung teils starke Schwankungen aufweist. Diese Größe wird anhand eines Schritts zum Kalibrieren festgelegt. Wenn der aktuell berechnete Wert diese Grenze unterschreitet, wird der Datenabschnitt als gültige Wiederholung erkannt und die Anzahl der Wiederholungen erhöht.

8.4.1 Anlegen einer neuen Übung

Wenn der Nutzer eine neue Übung aufzeichnen möchte, wählt er diese aus und hat dann fünf Sekunden Zeit, um in die Startposition zu gelangen. Bei Seitheben mit Kurzhanteln sollten die Arme parallel zum Oberkörper sein (Ausführung siehe Abbildung 11a). Die Smartwatch signalisiert den Start der Wiederholung durch Vibration. Nach der Hälfte der Zeit (2 Sekunden) sollte die Hälfte einer Wiederholung abgeschlossen sein, was durch erneutes Vibrieren angezeigt wird. Beim Seitheben sollten die Arme etwa parallel zum Boden sein. Anschließend wird die Übung fortgesetzt und in der Ausgangsposition beendet. Das Ende wird durch eine dritte Vibration signalisiert. Die Werte der Achsen werden separat für jeden der beiden Sensoren gespeichert. Außerdem wird eine Kombination der Sensordaten für den DTW-Algorithmus abgespeichert. Dabei wird die Reihenfolge $Acc_x + Acc_y + Acc_z + Gyr_x + Gyr_y + Gyr_z$ eingehalten. Um falsche Wiederholungen auszuschließen wird ein Mindestabstand genutzt. Dieser wird aus der Hälfte der Länge eines Achsendatensatzes hier berechnet.

8.4.2 Kalibrieren

Wie im Unterabschnitt 8.4 beschrieben, muss ein Grenzwert für das Ergebnis des DTW-Algorithmus definiert werden, unterhalb dessen eine Wiederholung als gültig angesehen wird. Der Kalibrierungsschritt wird nach der Aufzeichnung der Übung durchgeführt und muss vor der eigentlichen Erkennung erfolgen (Screenshot siehe Abbildung 16b). Ähnlich wie bei der Erfassung in Unterabschnitt 5.4 wird nach einem initialen Countdown, um sich in die Startposition zu begeben, der Start durch Vibration angezeigt. Der Nutzer hat insgesamt 13 Sekunden Zeit, um exakt drei vollständige Wiederholungen der aufgezeichneten Übung auszuführen. Anschließend werden die aufgezeichneten Daten an den in Unterabschnitt 8.4 beschriebenen Algorithmus übergeben. Sollten weniger als drei Wiederholungen erkannt werden, wird der Nutzer aufgefordert, die Kalibrierung erneut durchzuführen. Bei genau drei oder mehr erkannten Wiederholungen wird für jede Wiederholung der DTW-Wert berechnet. Die Ausgangslage für die Auswahl des Grenzwertes bilden die drei niedrigsten DTW-Werte. Der Grenzwert wird durch den höchsten der drei gefundenen Werte multipliziert und mit einem Faktor von 2 festgelegt. Vorher wird überprüft, ob der höchste der gefundenen Werte kleiner als 3500 ist. Bei zu schneller Ausführung wurde vereinzelt eine Wiederholung nicht erkannt. Wenn andere Bewegungen vor Ende der Aufzeichnung ausgeführt wurden, führte dies zu falschen Erkennungen mit einem viel zu hohen DTW-Wert. Um dies zu vermeiden, wurde eine Limitierung eingeführt. Sollte der Wert dennoch zu hoch sein, muss die Kalibrierung erneut durchgeführt werden.

Außerdem werden die niedrigsten drei DTW-Werte genutzt, um die Achse festzulegen, auf deren Daten die Berechnungen des Algorithmus mit dynamischen Grenzwerten durchgeführt werden. Es wurde bewusst nicht der Ansatz aus Zhang et al. verwendet. Dort wird kontinuierlich zur Laufzeit die Integrationsfläche berechnet. Wenn der höchste Wert für eine gewisse Zeit konstant auf einer Achse liegt, wird diese zur Berechnung verwendet. [1]

Dies hat den Nachteil des zusätzlichen Rechenaufwands. Außerdem reagiert das System somit nur verzögert auf geänderte Daten. Da das Ziel dieser Arbeit nicht das gleichzeitige Überwachen aller Übungen ist, kann die Achse bei der Kalibrierung anhand der Daten gesetzt werden.

8.4.3 Anpassung des kalibrierten Werts

Sollten bei der Erkennung Fehler auftreten und zu viele oder zu wenige Wiederholungen erkannt werden, hat der Nutzer die Möglichkeit, die Anzahl der Wiederholungen manuell anzupassen. Dabei wird im Hintergrund der durch die Kalibrierung festgelegte Grenzwert um jeweils 10 Prozent erhöht oder verringert. Sollte die Änderung um mehr als 50 Prozent des ursprünglich erkannten Wertes abweichen, wird davon ausgegangen, dass die aufgezeichneten Daten nicht sauber genug erfasst wurden. Der Nutzer wird gebeten, die Übung erneut aufzuzeichnen. Die Meldung ist in Abbildung 16c zu sehen.

8.5 Übungserkennung

Wenn ein Nutzer vergisst, die Übung manuell zu wechseln oder eine falsche auswählt, wird er durch eine Benachrichtigung darauf hingewiesen (Siehe: Abbildung 16c. Fällt die Rate der erfolgreichen Erkennungen bei den letzten 4 Wiederholungen unter 50 Prozent, wird für jede gespeicherte Übung der DTW-Wert berechnet. Wenn eine andere Übung als die aktuell ausgewählte einen niedrigeren DTW-Wert aufweist, gilt diese Übung als wahrscheinlich korrekt. Um zu überprüfen, ob der Nutzer darauf hingewiesen werden sollte, wird geprüft, ob der berechnete DTW-Wert unter dem maximalen DTW-Wert der Kalibrierung dieser Übung liegt. Da es möglich ist, dass der Algorithmus mit dynamischen Grenzwerten eigentlich eine andere Achse zur Berechnung verwenden sollte, kann es zu Verschiebungen kommen, was den DTW-Wert erhöhen kann. Aus diesem Grund wird der maximale Wert mit dem Faktor fünf erhöht.

Wenn nach dem beschriebenen Vorgehen eine andere Übung erkannt wird, wird dem Nutzer eine Benachrichtigung angezeigt. Außerdem vibriert das Gerät, um den Nutzer darauf aufmerksam zu machen.

9 Testergebnisse

Zur Bewertung der entwickelten Software wurde eine Testreihe durchgeführt. Die Teilnehmer wurden bei der Bedienung des Prototyps unterstützt. Die Aufzeichnung erfolgte nach erfolgreicher Kalibrierung der Übung. Insgesamt wurden drei Sätze aufgezeichnet. Nach jedem Satz wurde - falls erforderlich - die Anzahl der Wiederholungen angepasst.

Betrachtet wurden die Übungen:

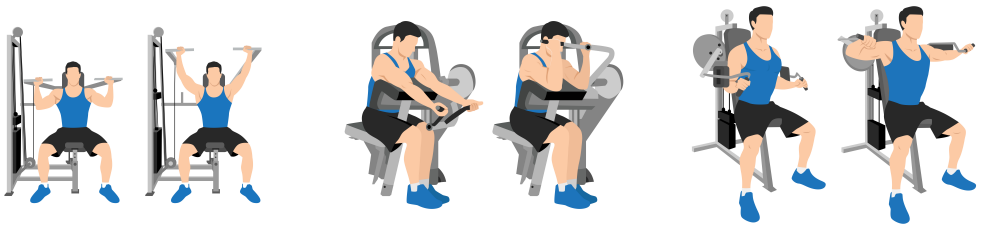
- Seitheben Maschine Abbildung 15c
- Schulterdrücken Abbildung 15a
- Preachercurls geführte Maschine Abbildung 15b

Insgesamt haben 11 Teilnehmer (TN) an den Tests mitgewirkt. Die Ergebnisse der Tests sind in Tabelle 1 abgebildet.

Von insgesamt 96 durchgeführten Sätzen mit gesamt 1018 Wiederholungen wurden 28.125 % exakt erkannt, 64.583 % mit einer maximalen Abweichung von 1 und 82.291 % mit einer maximalen Abweichung von 2 Wiederholungen.

Während der Testerfassung traten zwei Probleme auf:

Die Bewegung, um sich in die Ausgangsposition zu begeben, wurde sehr oft bereits als eine Wiederholung erfasst. Eine mögliche Begründung ist die große Ähnlichkeit der Bewegung zur tatsächlichen Ausführung. Bei Betrachtung der Übung Schulterdrücken (Abbildung 15a) fällt auf, dass bei den meisten Personen in den Pausen die Arme auf den Beinen liegen oder sie die Uhr kontrollieren und dann



(a) (b) (c)

Abbildung 15: Übungen für Tests. (15a: Schulterdrücken am Gerät, 15b: Preacher Curls, 15c: Seitheben am Gerät)

Tabelle 1: Testergebnisse

	Schulterdrücken			Seitheben			Preacher Curls		
TN 1	9/10	6/10	8/10	11/10	12/10	9/10	11/10	9/10	9/10
TN 2	9/10	9/10	11/10	8/10	10/10	10/10	11/10	7/10	10/10
TN 3	6/10	5/10	5/10	11/10	10/10	10/10	12/10	10/10	11/10
TN 4	6/10	8/8	7/9	4/10	12/10	10/10	9/10	11/10	10/10
TN 5	6/10	10/10	5/10	10/10	7/10	6/10	8/10	6/10	5/10
TN 6	16/18	15/15	16/15	13/15	16/16	12/12			
TN 7	14/12	18/17	16/16	17/16	13/15	13/15	8/10	9/10	10/10
TN 8	11/10	9/10	11/10	10/10	11/10	11/10	10/10	8/10	10/10
TN 9	8/10	8/10	9/10	11/10	10/10	11/10	10/10	11/10	10/10
TN 10	11/10	11/10	10/10	10/10	8/11	10/10	9/12	9/10	11/10
TN 11	11/8	9/9	9/10	9/10	10/10	9/10	11/10	10/10	11/9

nach oben greifen, um in die Ausgangsposition zurückzukehren. Dadurch entsteht eine starke vertikale Beschleunigung welche der eigentlichen Wiederholung ähnelt. Auch am Ende eines Satzes, bei der Rückkehr von der letzten Wiederholung in die Ruheposition zwischen den Sätzen, wurden vereinzelt Wiederholungen beobachtet. Diese Beobachtung lässt sich auch in den Testergebnissen erkennen. Die beiden TN 6 und 7 führten deutlich höhere Wiederholungszahlen als die anderen TN aus, wiesen jedoch keine höhere Fehlerquote auf. Obwohl diese Ergebnisse die insgesamt eher schlechten Testergebnisse nicht vollständig erklären können, würde das Beheben dieser Problematik die Gesamtergebnisse im Schnitt vermutlich um eine Wiederholung genauer machen. Dies würde, ausgehend von den gesammelten Daten, zu circa 64 % exakter und 82 % Erkennungsgenauigkeit mit einer Abweichung von 1 führen. Obwohl diese Werte immer noch nicht ausreichen, um die Anwendung kommerziell zu vermarkten, bieten sie eine gute Ausgangslage, um durch weitere Verbesserungen die Werte auf ein gutes Niveau

zu erhöhen. Ein möglicher Ansatz, wie dies erreicht werden kann, wird in Unterunterabschnitt 10.2.1 vorgestellt.

Das zweite Problem betrifft das Aufzeichnen der Vorlage. Aktuell wird die Vorlage durch Vibration und einen definierten zeitlichen Abstand vorgegeben, wie in Unterunterabschnitt 8.4.1 beschrieben. Viele Teilnehmer hatten Schwierigkeiten, von ihrer gewohnten Wiederholungsgeschwindigkeit abzuweichen, wodurch sie entweder zu langsam oder zu schnell waren. Eine möglichst genaue Aufzeichnung der Vorlage ist jedoch enorm wichtig, wie sich auch an den Testergebnissen erkennen lässt. Wenn eine Vorlage sehr genau aufgezeichnet wurde, konnte die Übung in der Regel sehr präzise erkannt werden. Bei Teilnehmer 9 wurden Preacher Curls sehr gut erkannt und bei Teilnehmer 3 Seitheben. Dies legt nahe, dass die Vorlage dort sehr gut erkannt wurde. Dass die genaue Erkennung nicht vom Teilnehmer abhängt, sondern von der Übung, wird bei Teilnehmer 3 deutlich. Seitheben wurden dort sehr präzise erkannt, aber das Schulterdrücken sehr schlecht. Eine Lösung dieses Problems könnte eine Verbesserung der Aufzeichnungen bewirken, bei denen jeder Satz sehr schlecht erkannt wurde. Beispiele für solche sehr schlechten Erkennungen sind bei Teilnehmer 3 Schulterdrücken oder Teilnehmer 5 Preacher Curls zu finden. Zwei mögliche Lösungsansätze für dieses Problem werden in Unterunterabschnitt 10.2.2 vorgestellt.

10 Ausblick

10.1 Weiteres Vorgehen

Diese Arbeit zeigt, dass Übungen korrekt erkannt werden können, ohne dass vorheriges aufwendiges Training erforderlich ist. Um den entwickelten Prototypen zu verbessern, sind jedoch einige weitere Erweiterungen notwendig.

Im ersten Schritt ist es notwendig, die gesammelten Vorlagen und Wiederholungen abzuspeichern und zur erleichterten Verwaltung auf einem verbundenen Smartphone zur Verfügung zu stellen. Die in Tabelle 1 vorgestellten Ergebnisse zeigten, dass besonders die Erkennung des Satzanfangs und -endes, sowie das Aufzeichnen der Vorlage weiter verbessert werden müssen (Siehe: Unterunterabschnitt 10.2.2, Unterunterabschnitt 10.2.1). Anschließend muss getestet werden, ob der Einsatz von Machine Learning die Erkennungsrate verbessert. Wie im Unterabschnitt 6.4 beschrieben, scheint dafür ein CNN am vielversprechendsten. Die Datensätze, die aktuell an den DTW-Algorithmus übergeben werden, könnten als Datengrundlage verwendet werden. Daten, die einen bestimmten DTW-Grenzwert überschreiten, könnten aussortiert werden, um nur Datensätze für das Training zu verwenden die sauber erkannt wurden. Es könnte berücksichtigt werden, dass Datensätze, die nach der Erkennung vom Nutzer angepasst wurden, ausgesondert werden. Dadurch könnten automatisch im Hintergrund Datensätze für das Training des CNN von den Benutzern der Anwendung erstellt werden.

Um die Performance weiter zu verbessern, kann die Indexierung genutzt werden, wie unter Unterabschnitt 4.2 vorgestellt. Dadurch wird die Geschwindigkeit der Berechnungen bei der Überprüfung, ob der Wechsel von Übungen vergessen wurde, weiter optimiert. Sowohl die Indexierung als auch der FastDTW-Algorithmus könnten in diesem Fall genutzt werden, um die Berechnungen zu beschleunigen.

10.2 Weitere Verbesserungen

10.2.1 Präzisere Bestimmung des Startzeitpunkts

Wie in Tabelle 1 beschrieben, werden oft falsche Wiederholungen vor der ersten oder nach der letzten Wiederholung erkannt. Ein möglicher Ansatz, um dies zu verbessern, wäre die Einführung eines Zwischenschritts. Der Nutzer müsste in der Ausgangsposition für eine definierte Zeit von beispielsweise zwei Sekunden warten, um den Start des Satzes anzuzeigen. Die Erkennung kann einfach umgesetzt werden, indem ein Timer gestartet wird, sobald alle Sensorwerte sich innerhalb eines Toleranzbereichs bewegen, abgesehen von einem Grundrauschen. Erst nachdem ein solcher Start erkannt wurde, werden die Wiederholungen als gültig betrachtet. Die Erkennungen gelten so lange als gültig, bis wieder ein Warten am Ende des Satzes erkannt wird. Falsche Start-Erkennungen könnten verworfen werden, wenn innerhalb von 2 Minuten kein Ende erkannt wird. Weitere Einschränkungen könnten die Ergebnisse weiter verbessern, beispielsweise eine minimale Anzahl an Wiederholungen zwischen Anfang und Ende.

10.2.2 Verbesserte Aufzeichnung der Vorlage

Wie in Tabelle 1 dargestellt, ist die Erkennung der Vorlage ein wichtiger Schritt zur Identifizierung von Wiederholungen und kann bei schlechten Erfassungen zu starken Abweichungen bei der Erkennung führen.

Ein möglicher Ansatz zur Verbesserung wäre, ähnlich wie der Ansatz unter Unterunterabschnitt 10.2.1, die zeitliche Vorgabe mit dem Timer komplett zu entfernen und stattdessen nach dem Initialen Countdown darauf zu warten, dass die Uhr für beispielsweise zwei Sekunden nicht bewegt wurde. Wenn die Ausgangsposition erkannt wird, könnte die Uhr durch Vibration anzeigen, dass die Ausführung begonnen werden kann. Die Vorlage wird dann so lange aufgezeichnet, bis die Uhr wieder nicht bewegt wird. Dieser Ansatz hat den Vorteil, dass die Bewegung mit der normalen Geschwindigkeit des Nutzers durchgeführt werden kann und die Aufzeichnung des Beginn- und Endzeitpunkts genauer ist.

Eine Möglichkeit, das Ergebnis weiter zu verbessern, wäre, die Vorlage durch mehrere Aufzeichnungen zu erweitern. Anschließend könnte entweder ein Durchschnitt berechnet werden, um eine allgemeingültige Vorlage zu erhalten, oder die Vorlage genutzt werden, die mit den meisten anderen Aufzeichnungen übereinstimmt.

10.2.3 Verwendung eines zuvor definierten Trainingsplans

Um die Leistungsänderungen zwischen den Trainingseinheiten genau zu erfassen und leichte Steigerungen zu planen, wird meist ein Trainingsplan mit definierten Übungen verwendet. Dies hat den Vorteil, dass das Gewicht oder die Wiederholungszahl in kleinen Schritten erhöht werden kann. Aus diesem Grund werden typischerweise über einen längeren Zeitraum von mehreren Wochen bis Monaten dieselben Übungen zur Training der jeweiligen Muskelgruppe genutzt.

Diese Trainingspläne beschränken die Auswahl an Übungen deutlich, was die Erkennung erleichtert. Es müssen nicht alle, sondern nur noch eine niedrige zweistellige Anzahl an Übungen unterschieden werden.

RecoFit nutzt dies zur besseren Erkennung, indem eine Teilmenge der maximal 26 Übungen genutzt wird. [10]

10.2.4 Verwendung der zuletzt ausgeführten Übungen

Es kann vorkommen, dass aufgrund von Maschinendefekten, einem anderen Trainingsort oder einer hohen Belegung der Geräte vom Trainingsplan abgewichen werden muss. Um den Muskel weiterhin möglichst vergleichbar zu belasten, wird nach Möglichkeit eine ähnliche Übung genutzt.

Da Benutzer oft nur an den gleichen Trainingsorten trainieren, ist die Auswahl der Geräte beschränkt und nicht alle Übungen können durchgeführt werden. Außerdem werden oft bereits bekannte Übungen verwendet, die in vergangenen Trainingseinheiten oder alten Trainingsplänen genutzt wurden.

Dies kann hilfreich sein, wenn keine passenden Übungen im aktuellen Trainingsplan gefunden werden können. Denn kürzlich ausgeführte Übungen haben eine höhere Wahrscheinlichkeit, erneut ausgeführt zu werden.

10.2.5 Erweiterung mit Herzfrequenz

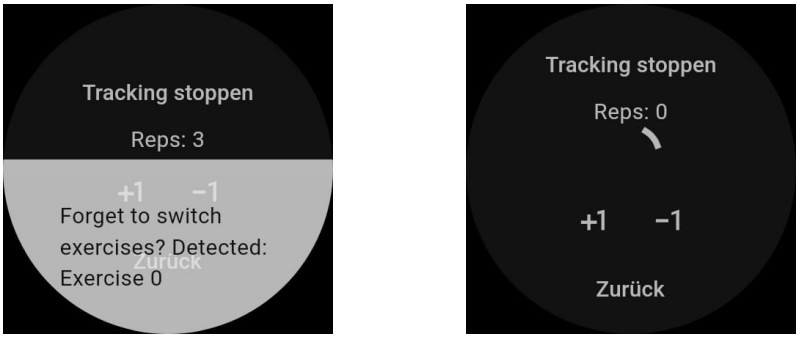
Wie in Abschnitt 3 dargestellt, werden bei einigen Übungen keine Bewegungen des Handgelenks ausgeführt und sind daher mit dem vorgestellten Ansatz nicht erkennbar. Um den Nutzer daran zu erinnern, dass der zuletzt ausgeführte Satz nicht erfasst wurde, sollte dieser darauf hingewiesen werden, sobald der Satz beendet ist.

Um zwischen einer im Sitzen ausgeführten Übung und einer Pause zu unterscheiden, ist es sinnvoll, auf den Puls des Nutzers zuzugreifen. In den Pausen zwischen den Übungen sollte der Puls tendenziell sinken, während während der Übungen eine tendenzielle Steigerung zu vermuten ist.

11 Anhang

(a) Startbildschirm Aufzeichnen neuer Übungen

(b) Kalibrieren und anschließendes Aufzeichnen



(c) Warnung Übungswechsel und Überprüfung mögliche Wiederholung

Abbildung 16: Screenshots der Anwendung

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Literatur

[1] S. Zhang, Z. Li, J. Nie, L. Huang, S. Wang und Z. Wei, „How to record the amount of exercise automatically? A general real-time recognition and counting approach for repetitive activities," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2016, S. 831–834. DOI: 10.1109/BIBM.2016.7822633.

[2] fortunebusinessinsights, *Smartwatch Market Size & Growth | Industry Outlook [2030]*, https://www.fortunebusinessinsights.com/smartwatch-market-106625, (Besucht am 02/26/2024), Aug. 2023.

[3] *Smartwatches - Worldwide | Statista Market Forecast*, https://www.statista.com/outlook/hmo/digital-health/digital-fitness-well-being/fitness-trackers/smartwatches/worldwide, (Besucht am 02/26/2024), Feb. 2024.

[4] C. Shyalika. „Dynamic Time Warping (DTW)." (15. Mai 2019), Adresse: https://medium.datadriveninvestor.com/dynamic-time-warping-dtw-d51d1a1e4afc (besucht am 14. 02. 2024).

[5] S. Salvador und P. Chan, „Toward Accurate Dynamic Time Warping in Linear Time and Space," Bd. 11, Jan. 2004, S. 70–80.

[6] P. Senin. „Dynamic Time Warping Algorithm Review." (Dez. 2008), Adresse: https://www.researchgate.net/publication/228785661_Dynamic_Time_Warping_Algorithm_Review#pf10.

[7] S.-W. Kim, S. Park und W. Chu, „An index-based approach for similarity search supporting timewarping in large sequence databases," Feb. 2001, S. 607–614, ISBN: 0-7695-1001-9. DOI: 10.1109/ICDE.2001.914875.

[8] E. Keogh und C. Ratanamahatana, „Exact indexing of dynamic time warping," *Knowledge and Information Systems*, Jg. 7, S. 358–386, Jan. 2005. DOI: 10.1007/s10115-004-0154-9.

[9] B.-K. Yi, H. Jagadish und C. Faloutsos, „Efficient retrieval of similar time sequences under time warping," in *Proceedings 14th International Conference on Data Engineering*, 1998, S. 201–208. DOI: 10.1109/ICDE.1998.655778.

[10] D. Morris, T. S. Saponas, A. Guillory und I. Kelner, „RecoFit: Using a Wearable Sensor to Find, Recognize, and Count Repetitive Exercises," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Ser. CHI '14, Toronto, Ontario, Canada: Association for Computing Machinery, 2014, S. 3225–3234, ISBN: 9781450324731. DOI: 10.1145/2556288.2557116. Adresse: https://doi.org/10.1145/2556288.2557116.

[11] C. Shen, B.-J. Ho und M. Srivastava, „MiLift: Efficient Smartwatch-Based Workout Tracking Using Automatic Segmentation," *IEEE Transactions on Mobile Computing*, Jg. 17, Nr. 7, S. 1609–1622, 2018. DOI: 10.1109/TMC.2017.2775641.

[12] V. B. Terry Taewoong Um und D. Kulic´. „Exercise Motion Classification from Large-Scale Wearable Sensor Data Using Convolutional Neural Networks." (), Adresse: https://ieeexplore.ieee.org/abstract/document/8206051?casa_token=1E5NSyNigVAAAAAA:4o8RNCZH51YvBwP6KS-HrHJ08sWqK5nb9eV-DH5-m8XBXyZg9CIcBBiGHM6MHBaK4otExjsmggU (besucht am 25. 10. 2023).

[13] A. Soro, G. Brunner, S. Tanner und R. Wattenhofer, „Recognition and Repetition Counting for Complex Physical Exercises with Deep Learning," *Sensors*, Jg. 19, Nr. 3, 2019, ISSN: 1424-8220. DOI: 10.3390/s19030714. Adresse: https://www.mdpi.com/1424-8220/19/3/714.

[14] *PUSH Design Solution Inc.* http://www.trainwithpush.com/, Besucht am 10/11/2023.

[15] G. Š. D. Džaja M. Čibarić und R. Magjarević, „Accelerometer-based algorithm for the segmentation and classification of repetitive human movements during workouts," *Automatika*, Jg. 64, Nr. 2, S. 211–224, 2023. DOI: 10.1080/00051144.2022.2121247. eprint: https://doi.org/10.1080/00051144.2022.2121247. Adresse: https://doi.org/10.1080/00051144.2022.2121247.

[16] P. Viana, T. Ferreira, L. Castro u. a., „GymApp: A Real Time Physical Activity Trainner on Wearable Devices," in *2018 11th International Conference on Human System Interaction (HSI)*, 2018, S. 513–518. DOI: 10.1109/HSI.2018.8431358.

[17] M. Maheedhar, A. Gaurav, V. Jilla, V. N. Tiwari und R. Narayanan, „StayFit: A wearable application for Gym based power training," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2016, S. 6290–6293. DOI: 10.1109/EMBC.2016.7592166.

[18] W. Coates und J. Wahlström, „LEAN: Real-Time Analysis of Resistance Training Using Wearable Computing," *Sensors*, Jg. 23, Nr. 10, 2023, ISSN: 1424-8220. DOI: 10.3390/s23104602. Adresse: https://www.mdpi.com/1424-8220/23/10/4602.

[19] B. J. Mortazavi, M. Pourhomayoun, G. Alsheikh, N. Alshurafa, S. I. Lee und M. Sarrafzadeh, „Determining the Single Best Axis for Exercise Repetition Recognition and Counting on SmartWatches," in *2014 11th International Conference on Wearable and Implantable Body Sensor Networks*, 2014, S. 33–38. DOI: 10.1109/BSN.2014.21.

[20] S. Ishii, A. Yokokubo, M. Luimula und G. Lopez, „ExerSense: Physical Exercise Recognition and Counting Algorithm from Wearables Robust to Positioning," *Sensors*, Jg. 21, Nr. 1, 2021, ISSN: 1424-8220. DOI: 10.3390/s21010091. Adresse: https://www.mdpi.com/1424-8220/21/1/91.

[21] H. Gjoreski, J. Bizjak, M. Gjoreski und M. Gams. „Comparing Deep and Classical Machine Learning Methods for Human Activity Recognition using Wrist Accelerometer." (), Adresse: https://sites.cc.gatech.edu/~alanwags/DLAI2016/2.%20(Gjoreski+)%20Comparing%20Deep%20and%20Classical%20Machine%20Learning%20Methods%20for%20Human%20Activity%20Recognition%20using%20Wrist%20Accelerometer.pdf (besucht am 25. 10. 2023).

[22] Z. Zhuang und Y. Xue, „Sport-Related Human Activity Detection and Recognition Using a Smartwatch," *Sensors*, Jg. 19, Nr. 22, 2019, ISSN: 1424-8220. DOI: 10.3390/s19225001. Adresse: https://www.mdpi.com/1424-8220/19/22/5001.

[23] S. Sivakumar, Y. J. Kun und A. A. Gopalai, „High-Intensity Interval Training Exercise Recognition using Smartwatch," in *2020 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 2021, S. 206–211. DOI: 10.1109/IECBES48179.2021.9398735.

[24] T. P. Nils Y. Hammerla Shane Halloran, „Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables," 2016. Adresse: https://doi.org/10.48550/arXiv.1604.08880.

[25] S. M, *Learn About Principal Component Analysis in Details! - Analytics Vidhya*, https://www.analyticsvidhya.com/blog/2022/03/learn-about-principal-component-analysis-in-details/, (Besucht am 02/26/2024), März 2022.

OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

ERKLÄRUNG
ZUR BACHELORARBEIT VON

Name: Schottenhammer

Vorname: Markus

Studiengang: Informatik

1. Mir ist bekannt, dass dieses Exemplar der Bachelorarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.

2. Ich erkläre hiermit, dass ich diese Bachelorarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Regensburg, den 13.03.2024

.....
Unterschrift

Diese Erklärung ist mit der Bachelorarbeit (eingeheftet) abzugeben.

Stand: 21.09.2018/Abt. III