JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Institut für Informatik

Shape-basierte Bewegungsklassifikation auf multidimensionalen Zeitreihen mit Hilfe elastischer Abstandsmaße

Dissertation

zur Erlangung des Grades
"Doktor der Naturwissenschaften"
am Fachbereich Physik, Mathematik und Informatik
der Johannes Gutenberg-Universität in Mainz

vorgelegt von
Frédéric Alexander Stein
geboren am 12.12.1982 in Mainz

Mainz, Juni 2017

Erstgutachter:
[Name ausgelassen in der elektronischen Version der Dissertation]
Zweitgutachter:
[Name ausgelassen in der elektronischen Version der Dissertation]

Datum der mündlichen Prüfung:
11. Oktober 2017

Erklärung

Ich erkläre hiermit, dass ich diese Arbeit selbstständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel angefertigt habe.

Mainz, den ....., ...

(Frédéric Alexander Stein)

Zusammenfassung

Bewegungs-Zeitreihen sind in der heutigen Zeit allgegenwärtig. Durch die günstig verfügbaren Messsysteme wie z.B. Inertialsensoren in Smartphones und Fitness-Trackern, sowie Bewegungserfassungs-Systeme in der Unterhaltungs-Industrie wie z.B. Microsofts Kinect™ oder Nintendos Wii Remote™ lassen sich Bewegungs-Zeitreihen einfach, kostengünstig und in großer Zahl generieren. Die Erkennung einzelner Teilbewegungen bzw. Gesten spielt dabei eine bedeutende Rolle, z.B. bei Videospiel-Steuerungen oder bei der automatisierten Überwachung industrieller Fertigungsprozesse.

Diese Arbeit beschäftigt sich im Wesentlichen mit der Shape-basierten Bewegungsklassifikation. Dies bedeutet, dass die Bewegungen (z.B. Gesten) in Form einzelner – im Allgemeinen mehrdimensionaler – Bewegungs-Zeitreihen (im Folgenden als „Shapes“ bezeichnet) vorliegen. Die Aufgabe besteht nun darin, ein Anfrage-Shape mit Hilfe von bekannten Bewegungsdaten einem entsprechenden Bewegungsmuster bzw. einer Geste zuzuordnen. Dieser Prozess wird auch als Klassifikation bezeichnet. Die One-Nearest-Neighbor-(1NN)-Klassifikation ist dabei das intuitivste und einfachste Vorgehen, welches zusätzlich eine einfache physikalische Interpretation der Bewegungs-Zeitreihen zulässt.

Nach einer kurzen Einleitung in die Thematik werden die in dieser Arbeit verwendeten Datensätze kurz vorgestellt, auf denen die untersuchten Verfahren getestet werden. Neben einigen öffentlich frei zugänglichen Datensätzen wurde im Rahmen dieser Arbeit auch ein eigener Datensatz erzeugt. Dabei wurde die Hand verschiedener Probanden während des Zeichnens von Ziffern in 6 Freiheitsgraden (6-DoF) getrackt.

Danach wird zunächst ein Verfahren vorgestellt, das mittels Continuous-Wavelet-Transformation einen Bewegungs-Datenstrom (semi-)automatisiert in einzelne Teilbewegungen (Shapes) segmentieren kann.

Der Hauptteil dieser Arbeit beschäftigt sich anschließend mit der Shape-basierten Klassifikation von Bewegungen mittels Dynamic Time Warping (DTW). Um die 1NN-Klassifikation effizienter durchführen zu können, werden zuerst neue Lower Bounds für Bewegungs-Zeitreihen eingeführt, indem zwei in der Literatur bekannte Lower Bounds auf Quaternionen-wertige Zeitreihen erweitert werden. In diesem Zusammenhang wird eine schnell zu berechnende Approximation der komplexeren Lower Bound vorgestellt. Anschließend werden verschiedene Zeitreihen-Abstandsmaße, sowie verschiedene punktweise Abstandsmaße hinsichtlich ihrer Eignung für die Klassifikation miteinander verglichen. Die „Punkte“ der Zeitreihen sind in diesem Fall verschiedene Darstellungen von Rotationen. Im Anschluss wird noch ein neuartiger, bisher noch nicht bekannter Ansatz zur Klassifikation anhand von inkrementellen Rotationen beschrieben.

Abschließend werden verschiedene Ansätze für eine Sensordaten-Fusion auf sehr unterschiedlichen Daten getestet und miteinander verglichen. Hierzu wird ein neuartiges – auf Dynamic Time Warping basierendes – Verfahren zur zeitlichen Synchronisation von Bewegungs-Zeitreihen vorgestellt, die in unterschiedlichen Koordinatensystemen dargestellt sind.

Abstract

Motion time series are ubiquitous at present. Thanks to low-priced measuring systems such as inertial sensors in smartphones and fitness trackers, as well as motion detection systems in the entertainment industry, e.g. Microsoft's Kinect™ or Nintendo's Wii Remote™ motion time series can be easily and cost-effectively generated in large numbers. The recognition of individual partial movements or gestures plays an important role, e.g. for video game controls or automated monitoring of industrial production processes.

This thesis focuses on the shape-based motion classification. This means that the movements (gestures) are represented in the form of single – generally multi-dimensional – motion time series, from here on referred to as "shapes". The task is to assign a query shape to a corresponding movement pattern or a gesture, with the help of known motion data. This task is also known as classification. The one nearest neighbor (1-NN) classification is the most intuitive and straightforward procedure which also allows for a simple physical interpretation of the motion time series.

After a brief introduction to the topic the datasets that will be used to test the investigated methods are presented. In addition to publicly accessible datasets a new dataset has been generated within this work. The hand of different test subjects was tracked with 6 degrees of freedom (6DoF) while drawing digits.

Subsequently, a method is presented which uses a continuous wavelet transform to (semi-) automatically segment a motion data stream into subsequences (shapes).

The core part of this thesis focuses on the shape-based classification of movements by means of dynamic time warping (DTW). To perform the 1-NN classification more efficiently new lower bounds are presented for motion time series by extending two well known lower bounds from the literature to quaternion-valued time series. In this context a fast approximation of the more complex lower bound is presented. Next, different time series distance measures as well as different pointwise distance measures are compared regarding their suitability for the classification. The "points" of the time series in this case are different representations of rotations. Subsequently, a novel approach to classification based on incremental rotations is described.

Finally, different approaches for a sensor data fusion process are compared and tested on very different kinds of data. To this end, a novel method based on dynamic time warping is introduced for the temporal synchronization of motion time series, which are represented in different reference frames.

INHALTSVERZEICHNIS

Einleitung

In unserer heutigen Welt sind Bewegungs-Zeitreihen allgegenwärtig. Große Mengen an Bewegungs-Daten können heutzutage auf viele Arten aufgezeichnet werden. Die meisten Leute nutzen inzwischen Smartphones, die in der Regel mit Inertialsensoren ausgestattet sind. Ebenso kommen vermehrt Fitness-Tracker zum Einsatz, die ebenfalls in der Lage sind, Bewegungen mit Hilfe von Inertialsensoren aufzuzeichnen. Daneben gibt es inzwischen auch in der Unterhaltungsindustrie viele Sensorsysteme, die Bewegungen erfassen können. Beispiele hierfür sind Microsofts Kinect™ und Nintendos Wii Remote™ . Damit lassen sich Bewegungs-Zeitreihen einfach, kostengünstig und in großer Zahl generieren. Je nach verwendetem Sensorsystem werden die Bewegungen dabei in 3 Freiheitsgraden (z.B. bei reinen Inertialsensoren) oder sogar in 6 Freiheitsgraden (bei Ultraschall- oder Kamera-basierten Sensorsystemen) aufgezeichnet. Die Erkennung und Klassifikation einzelner Bewegungen spielt dabei eine große Rolle, z.B. bei der Gestenerkennung. Diese ist z.B. für interaktive Video-Spiele relevant, bei denen statt eines herkömmlichen Spiele-Controllers der Kopf, die Arme oder sogar der ganze Körper des Spielers als „Eingabegerät“ dienen. Es gibt aber auch industrielle Anwendungen z.B. bei der Überwachung bestimmter Fertigungsschritte bei Fließbandarbeiten.

Diese Arbeit setzt sich aus diesem Grunde vor allem mit der Klassifikation von Bewegungs-Zeitreihen auseinander.

Dazu werden im ersten Kapitel zunächst die theoretischen Grundlagen erläutert. Im Anschluss daran werden die verwendeten Test-Datensätze vorgestellt. Auf diesen Test-Daten werden später die verschiedenen Algorithmen validiert und ausgewertet.

Im zweiten Kapitel wird dann zunächst ein Verfahren beschrieben, mit dem ein gegebener mehrdimensionaler Bewegungs-Datenstrom semi-automatisiert in einzelne Teilbewegungen segmentiert werden kann. Anschließend folgt der Hauptteil dieser Arbeit, der sich mit der Klassifikation von Bewegungen beschäftigt – also der Zuordnung einzelner Teilbewegungen zu einer Bewegungsklasse (z.B. Geste). Das intuitivste und einfachste Klassifikations-Verfahren ist die sogenannte One-Nearest-Neighbor- (1NN)-Klassifikation. Diese wird hier für die Bewegungs-Klassifikation verwendet, da sie neben einer intuitiven Vorgehensweise auch eine sinnvolle physikalische Interpretation der Zeitreihen-„Punkte“ (z.B. als Rotationen oder als starre Transformationen) zulässt. Hierfür werden zunächst sogenannte Lower Bounds eingeführt, welche den Klassifikations-Prozess entscheidend beschleunigen können. Anschließend werden verschiedene Abstandsmaße auf Bewegungs-Zeitreihen miteinander verglichen. Am Ende des zweiten Kapitels wird dann ein neuartiger Ansatz zur Bewegungs-Klassifikation anhand von inkrementellen Rotationen vorgestellt.

Das dritte Kapitel beschäftigt sich mit der Sensordaten-Fusion. Dabei geht es um die Fra-

gestellung, wie zwei Messreihen, die von einer einzigen Bewegung stammen, aber mit zwei unterschiedlichen Sensorsystemen aufgezeichnet wurden, miteinander fusioniert werden können. Dadurch lässt sich die Messgenauigkeit erhöhen und Messausfälle eines einzelnen Sensors können kompensiert werden. Dazu werden zwei bestehende Fusions-Verfahren miteinander verglichen. Daran anschließend wird ein neues Verfahren zur zeitlichen Synchronisation zweier Messreihen, die relativ zu unterschiedlichen Koordinatensystemen gemessen wurden, vorgestellt.

Zum Schluss werden die Ergebnisse der einzelnen Kapitel noch einmal zusammengefasst und ein kurzer Ausblick über mögliche weitere Forschung auf diesem Gebiet gegeben.

Hinweis: In dieser Arbeit wird als Dezimaltrennzeichen – wie in der überwiegend in englischer Sprache verfassten Literatur – der Punkt (.) verwendet.

KAPITEL 1

EINFÜHRUNG UND THEORETISCHE GRUNDLAGEN

1.1 Zeitreihen

Eine Zeitreihe ist eine diskrete Abfolge von Datenpunkten. Diese Datenpunkte werden in der Regel durch eine Messung, die über ein bestimmtes Zeitintervall durchgeführt wird, erzeugt.

Definition 1: Zeitreihe

Sei M eine (nicht leere) Menge und $f : \mathbb{R} \to M$ eine Funktion mit reellwertigem Definitionsbereich. Sei nun $T = \{t_0, t_1, \ldots, t_{n-1}\}$ eine abzählbare Untermenge von $\mathbb{R}$ dann nennt man eine Abbildung
$\Gamma := \{(t_0, f(t_0)), (t_1, f(t_1)), \ldots, (t_{n-1}, f(t_{n-1}))\}$ eine **(M-wertige) Zeitreihe**.

Hierbei wird gefordert, dass die Elemente einer Zeitreihe aufsteigend nach der Zeit sortiert sind: $t_0 < t_1 < \cdots < t_{n-2} < t_{n-1}$. Sind die Zeitschritte immer gleich groß, gilt also $|t_i - t_{i-1}| = |t_k - t_{k-1}| \forall i, k$, reicht es aus, nur die Messwerte $f(t_i)$ zu betrachten. Dieser Fall liegt z.B. vor bei Sensoren, die mit einer festen Rate (z.B. 50 Hz) Werte aufzeichnen.

Jeder Datenpunkt f_i stellt einen Messwert dar. Da $f_i \in M$, haben alle Datenpunkte die gleiche Dimension. Beispielsweise können dies Temperaturwerte (z.B in °C) sein, die Geschwindigkeit eines Fahrzeugs (z.B. in km/h), ein bestimmter Aktienkurs jeweils zu Börsenschluss (z.B. in Punkten) oder gemessene Spannungen (z.B. in mV) in einem EKG oder EEG.

Die von Bewegungssensoren aufgezeichneten Datenpunkte können z.B. Posen mit jeweils 6 Freiheitsgraden sein:

$$f_i = \{x_i, r_i\}$$

wobei x_i Positionen und r_i Rotationen (Orientierungen) im 3-dimensionalen Raum sind. Eine solche Pose lässt sich beispielsweise als Paar eines Positionsvektors $\vec{x} \in \mathbb{R}^3$ und eines Rotationsquaternions $q_i \in \mathbb{H}_1$ darstellen:

$$f_i = \{\vec{x}_i, q_i\}$$

Eine weitere Darstellungsmöglichkeit für eine Pose ist eine homogene 4×4-Matrix aus der speziellen Euklidischen Gruppe ($SE(3)$):

$$f_i = \begin{pmatrix} \Theta_{00} & \Theta_{01} & \Theta_{02} & t_x \\ \Theta_{10} & \Theta_{11} & \Theta_{12} & t_y \\ \Theta_{20} & \Theta_{21} & \Theta_{22} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mit dem Translationsvektor $\vec{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \in \mathbb{R}^3$ und der (orthonormalen) Drehmatrix

$$\Theta = \begin{pmatrix} \Theta_{00} & \Theta_{01} & \Theta_{02} \\ \Theta_{10} & \Theta_{11} & \Theta_{12} \\ \Theta_{20} & \Theta_{21} & \Theta_{22} \end{pmatrix} \in SO(3) \text{ mit } \Theta\Theta^\top = \Theta^\top\Theta = \mathbb{1} \text{ und } \det(\Theta) = +1.$$

Die Posen werden von verschiedenen Sensoren gemessen. Zum Einsatz kommt hierbei vor allem ein Infrarot-Tracker-System, das von der Hochschule Rhein Main in Wiesbaden entwickelt wurde. Hierbei wird ein aktiver Marker mit einer speziellen Anordnung von Infrarot-LEDs von einer Kamera gefilmt und dessen Position und räumliche Orientierung relativ zur Kamera bestimmt. Dieses Messsystem erreicht eine Aufzeichnungsgeschwindigkeit von bis zu 160 Hz [1]. Des Weiteren wird ein Ultraschall-Tracker-System verwendet, welches von der Firma soft2tec in Rüsselsheim entwickelt wurde. Bei diesem wird ein Ultraschall-Beacon, bestehend aus drei Ultraschall-Emittern von einem Mikrofon-Array aufgenommen, welches ebenfalls dessen räumliche Position und Orientierung relativ zum Mikrofon-Array bestimmt. Hiermit können Posen mit einer Frequenz von ca. 50 Hz aufgezeichnet werden [2].

Cylinder-Bell-Funnel

Ein Beispiel für synthetisch erzeugte Zeitreihen ist der CBF-Datensatz [3]. Dieser besteht aus drei verschiedenen Klassen: einer Rechteckfunktion (Cylinder), einer aufsteigenden Dreiecksflanke (Bell) und einer absteigenden Dreiecksflanke (Funnel). Diese drei Klassen werden durch folgende Funktionen generiert:

$$C(t) := (A+\eta)\cdot\chi_{[a,b]}(t) + \varepsilon(t) \quad , \tag{1.1}$$

$$B(t) := (A+\eta)\cdot\chi_{[a,b]}(t)\cdot\tfrac{t-a}{b-a} + \varepsilon(t) \quad , \tag{1.2}$$

$$F(t) := (A+\eta)\cdot\chi_{[a,b]}(t)\cdot\tfrac{b-t}{b-a} + \varepsilon(t) \quad . \tag{1.3}$$

A bezeichnet hierbei die Basis-Amplitude der generierten Shapes. Die Zeitachse t wird im Bereich von $t \in \{0,\ldots,L-1\}$ gewählt. a ist eine zufällig uniform aus dem Intervall $[\frac{L}{8},\frac{L}{4}]$ gewählte ganze Zahl. b wird ebenfalls als ganze Zahl zufällig uniform so bestimmt, dass $b-a$ im Bereich von $[\frac{L}{4},\frac{3L}{4}]$ liegt. $\chi_{[a,b]}(t)$ ist eine sogenannte Indikatorfunktion auf dem Intervall $[a,b]$:

$$\chi_{[a,b]}(t) = \begin{cases} 1 & \text{wenn} \quad t \in [a,b] \\ 0 & \text{sonst} \end{cases}$$

Durch den Term $\frac{t-a}{b-a}$ wird eine aufsteigende Flanke erzeugt. Analog erzeugt der Term $\frac{b-t}{b-a}$ eine absteigende Flanke. Die Parameter η und $\varepsilon(t)$ werden zufällig aus der Normalverteilung mit $\mu = 0$ und $\sigma = 1$ gewählt. Während η für eine variable Amplitude der erzeugten Shapes verantwortlich ist, wird durch $\varepsilon(t)$ ein Gaußsches Rauschen auf die erzeugte Zeitreihe aufmoduliert. Abb. 1.1 zeigt einige Beispiel-Shapes des CBF-Datensatzes.

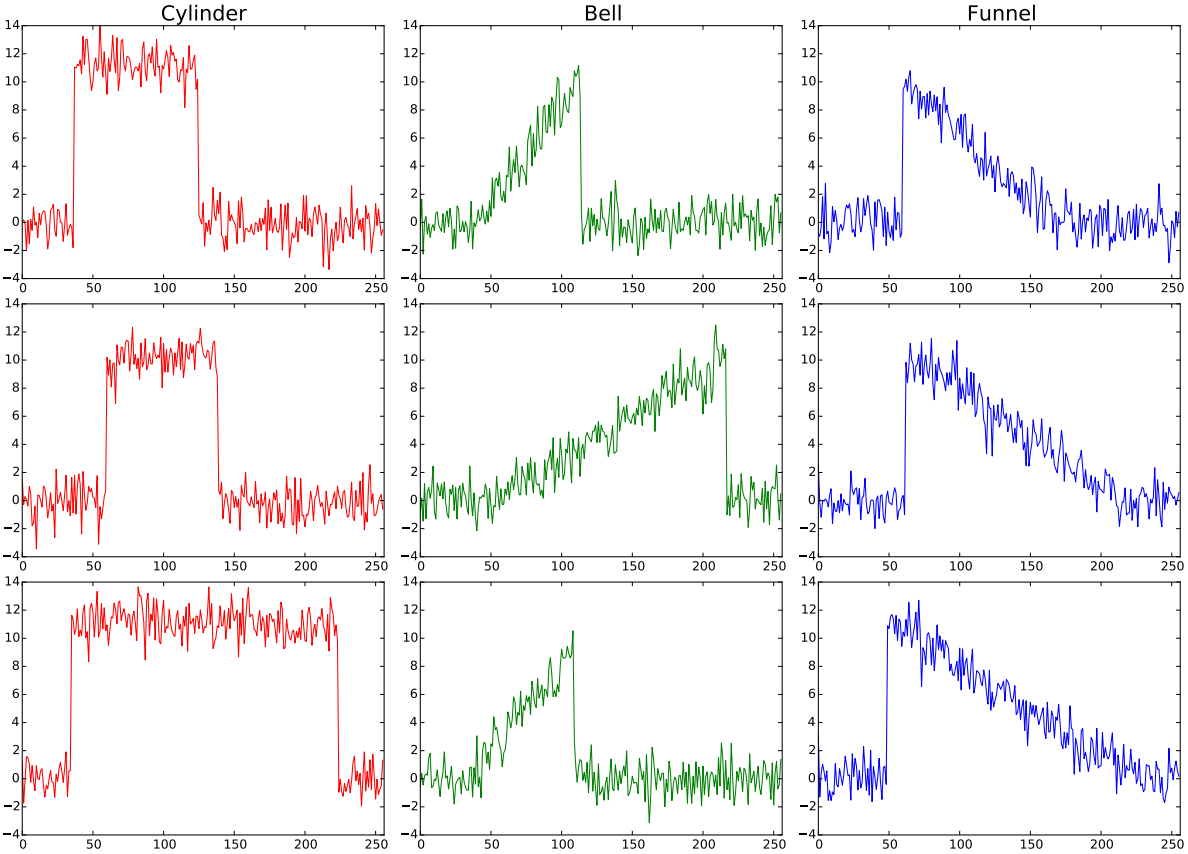


Abbildung 1.1: Beispiel-Shapes für Cylinder-Bell-Funnel. Zu jeder Klasse sind hier exemplarisch drei Beispiel-Shapes dargestellt. Als Basis-Amplitude wurde hier $A = 10$ gewählt. Die Länge beträgt $L = 256$ Zeitschritte. Wie man sehen kann, haben die Shapes alle unterschiedliche Ausdehnungen auf der Zeitachse. Deshalb eignet sich CBF sehr gut, die Fähigkeit des Dynamic Time Warping Algorithmus (siehe Kapitel 2) zu demonstrieren, Shapes mit unterschiedlicher Zeit-Skalierung aufeinander zu matchen.

1.2 Darstellungen von Orientierungen im 3-dimensionalen Raum

Während Orte (Positionen) bzw. Verschiebungen (Translationen) im 3-dimensionalen Raum in der Regel als Vektoren $t_i \in \mathbb{R}^3$ dargestellt werden, werden für Orientierungen bzw. Drehungen (Rotationen) unterschiedliche Möglichkeiten der Darstellung verwendet. Die verschiedenen Darstellungsmöglichkeiten sind zwar zueinander äquivalent, haben jedoch jeweils konzeptionelle Vor- und Nachteile. Vier der am häufigsten verwendeten Darstellungen sowie deren Vor- und Nachteile werden im Folgenden näher erläutert.

1.2.1 Rotations-Matrizen

Eine Möglichkeit ist die Darstellung von Rotationen als (orthonormale) **Rotationsmatrizen** $\Theta_i \in SO(3) \subset \mathbb{R}^{3\times 3}$. Für detaillierte Ausführungen und Herleitungen hierzu siehe z.B. [4].

Man kann jede Rotationsmatrix als Zerlegung in drei multiplikativ verknüpfte Matrizen darstellen, von denen jede jeweils eine Drehung um eine Achse eines kartesischen Koordinatensystems darstellt:

$$\Theta = \Theta_x \cdot \Theta_y \cdot \Theta_z \mid \Theta, \Theta_x, \Theta_y, \Theta_z \in SO(3)$$

$$\Theta_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}, \Theta_y = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}, \Theta_z = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

mit den Rotationswinkeln $\alpha, \beta, \gamma \in [0, 2\pi]$ um die jeweilige Koordinaten-Achse.

Alle Matrizen der speziellen orthogonalen Gruppe $SO(3)$ („Drehgruppe“) haben folgende **Eigenschaften**:

- Orthogonalität: $\Theta^\top \cdot \Theta = \Theta \cdot \Theta^\top = \mathbb{1}$. Daraus folgt auch, dass $\Theta^\top = \Theta^{-1}$ ist.
- Determinante: $\det(\Theta) = +1$. Hieraus folgt, dass die Händigkeit eines Koordinatensystems (rechtshändig oder linkshändig) unter Drehungen erhalten bleibt.
- Orthonormalität: alle Zeilen-Vektoren haben jeweils die Länge 1 und stehen paarweise senkrecht zueinander. Dies gilt entsprechend für die Spalten-Vektoren.
- Der resultierende Drehwinkel φ der Gesamtdrehung lässt sich aus der Spur berechnen: $\mathrm{Tr}(\Theta) = 1 + 2\cos(\varphi)$.

Eine durch eine Rotations-Matrix $\Theta \in SO(3)$ repräsentierte Drehung wirkt auf einen Punkt $\vec{p} \in \mathbb{R}^3$ durch einfache Matrix-Vektor-Multiplikation, wobei die Rotations-Matrix *von links* auf den Vektor wirkt:

$$\vec{p}\,' = \Theta \cdot \vec{p} \in \mathbb{R}^3$$

Die Hintereinander-Ausführung (Konkatenation) zweier Rotationen $\Theta_1 \in SO(3)$ und $\Theta_2 \in SO(3)$ wird durch das bekannte Matrizen-Produkt im $\mathbb{R}^{3\times3}$ realisiert:

$$\tilde{\Theta} = \Theta_2 \cdot \Theta_1 \in SO(3)$$

Hierbei wirkt zuerst die durch Θ_1 beschriebene Rotation auf einen Punkt $\vec{p}$ und anschließend die durch Θ_2 beschriebene Rotation:

$$\begin{aligned}\vec{p}'' &= \tilde{\Theta} \cdot \vec{p} \\ &= (\Theta_2 \cdot \Theta_1) \cdot \vec{p} \\ &= \Theta_2 \cdot (\Theta_1 \cdot \vec{p}) \\ &= \Theta_2 \cdot \vec{p}'\end{aligned}$$

Hierbei wurde die Assoziativität des Matrizen-Produktes ausgenutzt. Der Punkt $\vec{p}$ wird also wie erwartet zuerst per Θ_1 rotiert mit $\vec{p}' = \Theta_1 \cdot \vec{p}$. Dieser gedrehte Punkt wird anschließend per Θ_2 rotiert mit $\vec{p}'' = \Theta_2 \cdot \vec{p}'$. Dies entspricht der Konkatenation der durch Θ_1 und Θ_2 beschriebenen Drehungen.

Vorteile bei der Verwendung von Rotationsmatrizen:

+ Die Darstellung ist (innerhalb eines festgelegten Koordinatensystems) eindeutig, d.h. zu jeder Drehung im Raum gibt es *genau eine* Rotationsmatrix $\Theta \in SO(3)$, die diese Drehung beschreibt. Es besteht keine Antipodalität wie z.B. bei Quaternionen oder Rodrigues-Vektoren (siehe dort).
+ Die relative Drehung zwischen zwei Rotationsmatrizen lässt sich mit Hilfe von Logarithmen definieren.

Nachteile bei der Verwendung von Rotationsmatrizen:

- Man benötigt 9 Elemente zur Speicherung (für eine Drehung, die lediglich 3 Freiheitsgrade besitzt). Dies erfordert – gerade bei längeren Zeitreihen – entsprechend mehr Speicherplatz als andere Darstellungen.
- Der Rechenzeitaufwand z.B. bei der Multiplikation oder Inversen-Bildung ist recht hoch.
- Nach aufwändigeren Berechnungen wie z.B. der wiederholten Hintereinander-Ausführung von Drehungen durch Matrixmultiplikation muss die Ergebnismatrix erst wieder orthonormalisiert werden, um numerische Fehler bei der Berechnung zu korrigieren. Dies erfordert einen nicht unerheblichen Rechenaufwand.
- Eine Interpolation zwischen Drehungen bzw. eine Mittelwertbildung ist mit Drehmatrizen nicht sinnvoll möglich, da das Ergebnis nicht mehr zwingend eine gültige Rotationsmatrix wäre.

1.2.2 Rotations-Quaternionen

Eine weitere Möglichkeit der Darstellung von Rotationen im $\mathbb{R}^3$ ist die Verwendung von (normierten) Quaternionen. Für die formale Definition von Quaternionen und deren Eigenschaften sei auf das ursprüngliche Werk von W. R. Hamilton [5] verwiesen, der diese bereits im 19. Jahrhundert als neues mathematisches Konstrukt einführte. Für die Anwendung von normierten Quaternionen zur Darstellung von Rotationen siehe z.B. [6].

Hierbei wird eine Rotation folgendermaßen beschrieben:

Definition 2: Rotations-Quaternion

Sei $\varphi \in [0, 2\pi]$ der Rotationswinkel und $\hat{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \in \mathbb{R}^3$, $\|\hat{r}\| = 1$ die normierte Rotationsachse einer Drehung. Dann ist das zugehörige Rotations-Quaternion $q_{(\varphi,\hat{r})} \in \mathbb{H}_1$ definiert als:

$$q_{(\varphi,\hat{r})} := \begin{pmatrix} \cos(\frac{\varphi}{2}) \\ \sin(\frac{\varphi}{2}) \cdot \hat{r} \end{pmatrix} = \begin{pmatrix} \cos(\frac{\varphi}{2}) \\ \sin(\frac{\varphi}{2}) \cdot r_x \\ \sin(\frac{\varphi}{2}) \cdot r_y \\ \sin(\frac{\varphi}{2}) \cdot r_z \end{pmatrix} \in \mathbb{H}_1$$

Hierbei steht $\mathbb{H}$ für die Menge aller Quaternionen und $\mathbb{H}_1 = \{q \in \mathbb{H} \mid \|q\| = 1\}$ für die Menge aller normierten Quaternionen (Rotations-Quaternionen).

Das Rotations-Quaternion ist per Konstruktion normiert:

$$\begin{aligned} \|q_{(\varphi,\hat{r})}\|^2 &= \cos^2(\tfrac{\varphi}{2}) + \sin^2(\tfrac{\varphi}{2}) \cdot (r_x^2 + r_y^2 + r_z^2) \\ &= \cos^2(\tfrac{\varphi}{2}) + \sin^2(\tfrac{\varphi}{2}) \cdot \|\hat{r}\|^2 \\ &= \cos^2(\tfrac{\varphi}{2}) + \sin^2(\tfrac{\varphi}{2}) \\ &= 1 \end{aligned}$$

Die *Inverse* q^{-1} eines Quaternions q ist folgendermaßen definiert:

$$q^{-1} := \frac{q^*}{\|q\|}$$

q^* ist hierbei das *konjugierte* Quaternion:

$$q^* = \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix}^* := \begin{pmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{pmatrix}$$

Aus dieser Definition folgt direkt, dass die Inverse normierter Quaternionen gleich der Konjugierten ist:

$$q^{-1} = \frac{q^*}{\|q\|} = \frac{q^*}{1} = q^*$$

Das Produkt zweier Quaternionen ist wie folgt definiert:

Definition 3: Quaternionen-Produkt

Seien $q_1 = \begin{pmatrix} q_{1,w} \\ q_{1,x} \\ q_{1,y} \\ q_{1,z} \end{pmatrix} =: \begin{pmatrix} s_1 \\ \vec{v}_1 \end{pmatrix} \in \mathbb{H}$ und $q_2 = \begin{pmatrix} q_{2,w} \\ q_{2,x} \\ q_{2,y} \\ q_{2,z} \end{pmatrix} =: \begin{pmatrix} s_2 \\ \vec{v}_2 \end{pmatrix} \in \mathbb{H}$ zwei beliebige Quaternionen mit Skalar-Anteil $s_i = q_{i,w} \in \mathbb{R}$ und Vektor-Anteil $\vec{v}_i = \begin{pmatrix} q_{i,x} \\ q_{i,y} \\ q_{i,z} \end{pmatrix} \in \mathbb{R}^3$ dann ist das Quaternionen-Produkt $q_1 * q_2 \in \mathbb{H}$ wie folgt definiert:

$$q_1 * q_2 = \begin{pmatrix} s_1 \\ \vec{v}_1 \end{pmatrix} * \begin{pmatrix} s_2 \\ \vec{v}_2 \end{pmatrix} := \begin{pmatrix} s_1 \cdot s_2 - \langle \vec{v}_1 | \vec{v}_2 \rangle \\ s_1 \cdot \vec{v}_2 + s_2 \cdot \vec{v}_1 + \vec{v}_1 \times \vec{v}_2 \end{pmatrix} \in \mathbb{H}$$

Hierbei ist $\langle \cdot | \cdot \rangle$ das Skalarprodukt im $\mathbb{R}^3$ und $(\cdot \times \cdot)$ das Kreuzprodukt (Vektorprodukt) ebenfalls im $\mathbb{R}^3$.

Man beachte, dass das Quaternionen-Produkt zwar assoziativ ist:

$$(q_1 * q_2) * q_3 = q_1 * (q_2 * q_3) \quad \forall\, q_1, q_2, q_3 \in \mathbb{H}$$

jedoch im Allgemeinen nicht kommutativ:

$$\exists q_1, q_2 \in \mathbb{H} \mid q_1 * q_2 \neq q_2 * q_1$$

Ein Punkt im $\mathbb{R}^3$ wird mit Hilfe von Quaternionen nun wie folgt transformiert:

Definition 4: Rotationen im $\mathbb{R}^3$ mit Quaternionen

Sei $\vec{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \in \mathbb{R}^3$ ein beliebiger Punkt im 3-dimensionalen Raum. Dieser wird wie folgt in den Raum der Quaternionen $\mathbb{H}$ eingebettet:

$$q_p := \begin{pmatrix} 0 \\ \vec{p} \end{pmatrix} = \begin{pmatrix} 0 \\ p_x \\ p_y \\ p_z \end{pmatrix} \in \mathbb{H}$$

Sei nun $q_{(\varphi,\hat{r})}$ ein Quaternion, welches die Rotation um den Winkel φ um die Achse $\hat{r}$ beschreibt. Der um diese Achse $\hat{r}$ um den Winkel φ rotierte Punkt $\vec{p}'$ ist dann gegeben durch:

$$q_{p'} = \begin{pmatrix} 0 \\ \vec{p}' \end{pmatrix} = \begin{pmatrix} 0 \\ p'_x \\ p'_y \\ p'_z \end{pmatrix} = q_{(\varphi,\hat{r})} * q_p * q^{-1}_{(\varphi,\hat{r})} \in \mathbb{H}$$

$(*)$ bezeichnet hierbei das Quaternionen-Produkt in $\mathbb{H}$

Hieraus sieht man leicht, dass ein Quaternion q und das am Ursprung gespiegelte (hierzu *„antipodale“*) Quaternion $-q$ dieselbe Drehung beschreiben:

$$(-q_{(\varphi,\hat{r})}) * q_p * (-q^{-1}_{(\varphi,\hat{r})}) = (-1)^2 \cdot q_{(\varphi,\hat{r})} * q_p * q^{-1}_{(\varphi,\hat{r})} = q_{(\varphi,\hat{r})} * q_p * q^{-1}_{(\varphi,\hat{r})} = q_{p'}$$

Diese Eigenschaft wird als *Antipodalität* bezeichnet.

Die Hintereinander-Ausführung (Konkatenation) zweier Drehungen, die durch die Rotations-Quaternionen $q_1 \in \mathbb{H}_1$ und $q_2 \in \mathbb{H}_1$ beschrieben werden, ist hierbei einfach das Produkt der beiden Quaternionen:

$$\tilde{q} = q_2 * q_1$$

Hierbei wirkt auf einen gegebenen Punkt $\vec{p}$ zuerst die durch q_1 beschriebene Drehung und anschließend die durch q_2 beschriebene Drehung. Dies erkennt man leicht durch Einsetzen in Definition 4:

$$\begin{aligned} q_{p''} &= \tilde{q} * q_p * \tilde{q}^{-1} \\ &= (q_2 * q_1) * q_p * (q_2 * q_1)^{-1} \\ &= (q_2 * q_1) * q_p * (q_1^{-1} * q_2^{-1}) \\ &= q_2 * (q_1 * q_p * q_1^{-1}) * q_2^{-1} \quad = q_2 * q_{p'} * q_2^{-1} \end{aligned}$$

Der Punkt p wird also wie erwartet zuerst per q_1 rotiert mit $q_{p'} = q_1 * q_p * q_1^{-1}$. Dieser gedrehte Punkt wird anschließend per q_2 rotiert mit $q_{p''} = q_2 * q_{p'} * q_2^{-1}$. Dies entspricht der Konkatenation der durch q_1 und q_2 beschriebenen Drehungen.

Möchte man den relativen Drehwinkel zwischen zwei Rotations-Quaternionen q_a und q_b berechnen, so kann man folgenden Ansatz anwenden: Angenommen q_a sei die „Null-Rotation“, dargestellt durch das Einheits-Quaternion[1] $q_a = q_{\mathbb{1}} := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$. Das Skalarprodukt von q_a und q_b ist dann:

$$\begin{aligned} \langle q_a | q_b \rangle &= \langle \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} | \begin{pmatrix} \cos(\frac{\varphi}{2}) \\ \sin(\frac{\varphi}{2}) \cdot r_x \\ \sin(\frac{\varphi}{2}) \cdot r_y \\ \sin(\frac{\varphi}{2}) \cdot r_z \end{pmatrix} \rangle \\ &= 1 \cdot \cos(\tfrac{\varphi}{2}) + 0 \cdot \sin(\tfrac{\varphi}{2}) \cdot r_x + 0 \cdot \sin(\tfrac{\varphi}{2}) \cdot r_y + 0 \cdot \sin(\tfrac{\varphi}{2}) \cdot r_z \\ &= \cos(\tfrac{\varphi}{2}) \end{aligned}$$

Die Werte φ und $\hat{r}$ eines Rotations-Quaternions q parametrisieren im Allgemeinen die Rotation von der neutralen Lage $q_{\mathbb{1}}$ nach q. Wählt man nun wie in obigem Beispiel $q_a = q_{\mathbb{1}}$ dann ist q_b genau gleich der relativen Rotation zwischen q_a und q_b. Der relative Dreh-Winkel φ ist damit gegeben durch:

$$\varphi = 2 \cdot \arccos(\langle q_a | q_b \rangle) \tag{1.4}$$

Ist jetzt q_a ein beliebiges Rotations-Quaternion, so kann man sich zunutze machen, dass das Skalarprodukt invariant unter gleichzeitiger Rotation beider Quaternionen ist:

$$\langle q_x * q_a | q_x * q_b \rangle = \langle q_a | q_b \rangle$$

Damit lässt sich bei jedem beliebigen Paar von Rotations-Quaternionen durch gleichzeitige Anwendung einer Rotation auf beide Quaternionen o.B.d.A. jeweils eines auf die neutrale Rotation $q_{\mathbb{1}}$ drehen, ohne das resultierende Skalarprodukt zu verändern:

$$\langle q_a | q_b \rangle = \langle q_a^{-1} * q_a | q_a^{-1} * q_b \rangle = \langle q_{\mathbb{1}} | q_a^{-1} * q_b \rangle$$

So lässt sich der relative Winkel φ zwischen zwei beliebigen Rotations-Quaternionen q_a und $q_b \in \mathbb{H}$ mit Hilfe von Gleichung (1.4) berechnen.

Zusammengefasst besitzen Rotations-Quaternionen unter anderem folgende wichtige **Eigenschaften**:

[1] Da die Rotations-Quaternionen, ebenso wie die $SO(3)$, eine multiplikative Gruppe bilden, ist das neutrale Element die $\mathbb{1}$ und nicht – wie bei additiven Gruppen – die 0.

- Normierung: $\|q\|^2 = q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1$
- Antipodalität: q und $-q$ beschreiben dieselbe Drehung.
- Die Inverse eines normierten Quaternions ist gleich der Konjugierten: $q^{-1} = q^*$
- Zusammenhang zwischen Drehung um den Winkel φ um die (normierte) Achse $\hat{r}$: $q_{(\varphi,\hat{r})} = \begin{pmatrix} \cos(\frac{\varphi}{2}) \\ \sin(\frac{\varphi}{2}) \cdot \hat{r} \end{pmatrix}$

Vorteile bei der Verwendung von Quaternionen:

+ Es werden nur 4 Elemente zur Speicherung benötigt.
+ Eine Renormierung nach aufwändigeren Berechnungen zur Korrektur numerischer Fehler ist mit relativ geringem Rechenaufwand durchführbar: Das Quaternion muss lediglich durch seinen Betrag geteilt werden.
+ Auch wenn die Summe zweier Quaternionen nicht mehr notwendigerweise normiert ist und deshalb kein Rotations-Quaternion ist, kann man durch Addition und anschließende Renormierung eine Art Mittelwert zwischen Quaternionen definieren.
+ Eine Interpolation zwischen Quaternionen ist z.B. mit Hilfe von LERP oder SLERP möglich [6],[7].
+ Der Rechenzeitaufwand für die Inversen-Bildung ist minimal (lediglich drei Vorzeichen-Umkehrungen).
+ Die relative Drehwinkel zwischen zwei Quaternionen lässt sich über das Skalarprodukt berechnen.

Nachteile bei der Verwendung von Quaternionen:

- Auch wenn 4 Elemente zur Speicherung schon deutlich weniger sind, als z.B. die 9 Elemente bei der Verwendung von Rotations-Matrizen, ist immer noch ein redundantes Element dabei. Eine Drehung im 3-dimensionalen Raum hat nur 3 Freiheitsgrade.
- Um den relativen Drehwinkel zwischen zwei Quaternionen zu bestimmen, muss die arccos-Funktion aufgerufen werden, welche relativ viel Rechenzeit in Anspruch nimmt. Dies wirkt sich negativ auf die Gesamtlaufzeit der verwendeten Algorithmen aus – insbesondere bei längeren Zeitreihen.
- Durch die Antipodalität muss bei bestimmten Berechnungen wie z.B. der Interpolation zwischen zwei Quaternionen q_a und q_b darauf geachtet werden, dass die richtige Darstellung gewählt wird. Dies lässt sich z.B. dadurch bewerkstelligen, dass mit Hilfe

des Skalarproduktes immer der kleinere Winkel zwischen zwei Quaternionen wählt wird:

$$q_b \mapsto \begin{cases} -q_b & \text{wenn} \quad \langle q_a | q_b \rangle < 0 \\ +q_b & \text{wenn} \quad \langle q_a | q_b \rangle \geq 0 \end{cases}$$

Ließe man dies außer Acht, würden z.B zwei identische Drehungen, die einmal durch q und einmal durch $-q$ dargestellt sind, zu falschen Ergebnissen bei der Interpolation führen.

1.2.3 Euler-Winkel

Rotationen im dreidimensionalen Raum lassen sich des Weiteren mit Hilfe von sogenannten Euler-Winkeln darstellen. Deren Prinzip wird im Folgenden kurz erläutert. Für weitere Details und Herleitungen siehe z.B. [8].

Euler-Winkel lassen sich als dreifache Hintereinander-Ausführung von Rotationen um verschiedene Achsen interpretieren. Für die Wahl der Rotations-Achsen gibt es 12 Möglichkeiten:

- Die „klassischen“ Euler-Winkel:
  x-y-x, x-z-x, y-x-y, y-z-y, z-x-z, z-y-z
- Tait–Bryan-Winkel (auch Kardan-Winkel):
  x-y-z, x-z-y, y-x-z, y-z-x, z-x-y, z-y-x

Die erste Rotations-Achse ist dabei immer eine im Raum feste Achse, während die beiden weiteren Achsen jeweils mit den voran gegangenen Drehungen mitrotiert werden. Hierbei müssen zwei aufeinander folgende Drehungen jeweils um verschiedene Achsen erfolgen, damit jede Orientierung im dreidimensionalen Raum dargestellt werden kann. Beispielsweise hätte die Wahl der Achsen: y-y-z effektiv nur zwei Rotations-Achsen, da sich die Rotation um die Achse y, gefolgt von einer weiteren Rotation um dieselbe Achse y als eine einzige Rotation um diese Achse darstellen lässt, wodurch die gesamte Drehung nur noch 2 freie Parameter hätte. Die Wahl y-z-y ist wiederum erlaubt, da die Achsen mit jeder Rotation mitrotiert werden und deshalb die erste Achse y nicht mit der letzten Achse y übereinstimmen muss. Um dies zu unterstreichen könnte man für die Achsenwahl auch y-z'-y" schreiben, wobei dies folgendermaßen zu interpretieren ist: Die erste Rotation erfolgt um die Achse y. Dabei wird das Koordinatensystem mitrotiert. Die zweite Drehung erfolgt nun um die Achse z' des mitrotierten Koordinatensystems. Die letzte Drehung erfolgt dann um die Achse y" des zweimal mitrotierten Koordinatensystems (siehe Abb. 1.2). In der Fahrzeugtechnik werden Euler-Winkel als sogenannte Gier-, Nick- und Rollwinkel verwendet.

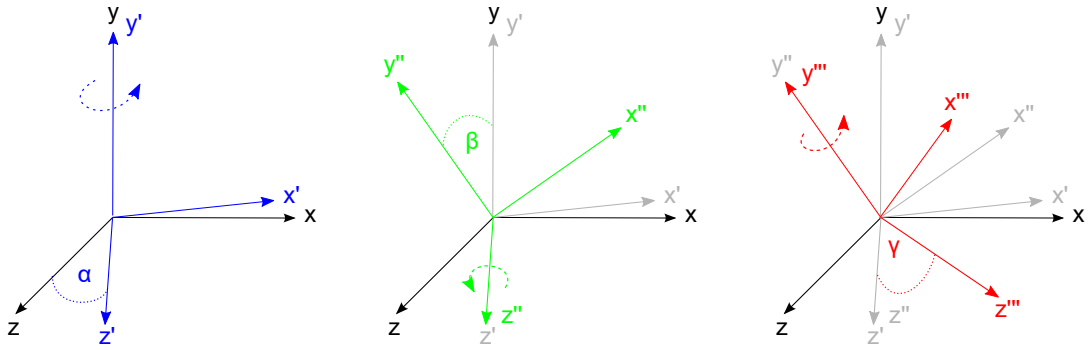


Abbildung 1.2: Beispiel für die Anwendung von Euler-Winkeln: Zuerst wird um die Achse y des ursprünglichen Koordinatensystems rotiert. Die zweite Rotation wird um die Achse z' des mitrotierten Koordinatensystems durchgeführt. Die letzte Rotation findet schließlich um die Achse y'' des zweimal mitrotierten Koordinatensystems statt.

Vorteile bei der Verwendung von Euler-Winkeln

+ Es werden genau 3 Parameter benötigt um eine Rotation zu beschreiben. Dies ist die minimale Anzahl an Parametern, da eine Rotation im dreidimensionalen Raum exakt 3 Freiheitsgrade besitzt. Dadurch ist der Speicheraufwand minimal.

Nachteile bei der Verwendung von Euler-Winkeln

- Es gibt 12 verschiedene Konventionen für die Darstellung von Euler-Winkeln. Man muss also genau wissen, welche Konvention verwendet wird, damit die korrekte Rotation bestimmt werden kann.
- Um eine Rotation auf einen Punkt im dreidimensionalen Raum anzuwenden, muss zuerst die entsprechende Rotationsmatrix (oder eine andere Darstellung, z.B. ein Rotations-Quaternion) konstruiert werden, die dann auf den Punkt wirken kann.
- Die Konkatenation von Rotationen, die als Euler-Winkel repräsentiert sind, muss ebenfalls über den Zwischenschritt von Rotationsmatrizen oder einer anderen Darstellung erfolgen.
- Auch die relative Drehung zwischen zwei gegebenen Rotationen muss über eine alternative Darstellung berechnet werden.
- Die mathematische Abbildung von Euler-Winkeln auf die zugehörige Rotationsmatrix besitzt kritische Punkte. Dort ist die Darstellung nicht mehr lokal umkehrbar. Es gibt also unendlich viele Euler-Winkel, die diese Rotation parametrisieren. Das passiert dann, wenn zwei der Rotationsachsen gleich sind. Beispielsweise wenn bei der y-z'-y'' Konvention der zweite Winkel 0 ist und damit y=y'' ist. Dieses Phänomen wird als **Kardanische Blockade** (englisch: **gimbal lock)** bezeichnet.

1.2.4 Achse-Winkel-Repräsentation und Rodrigues-Vektor

Die Darstellung von Rotationen als Rodrigues-Vektor ist wie folgt definiert (für eine detaillierte Herleitung siehe z.B. [9]):

Gegeben sei eine Rotation in **Achse-Winkel-Repräsentation** $(\varphi, \hat{r})$. Hierbei ist $\hat{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}$ mit $\|\hat{r}\| = 1$ die normierte Rotations-Achse im $\mathbb{R}^3$ und $\varphi \in [0, 2\pi]$ der Rotations-Winkel um diese Achse. Dann ist der zugehörige **Rodrigues-Vektor** das Produkt aus Rotations-Achse und Rotations-Winkel:

$$\vec{R} = \varphi \cdot \hat{r} = \begin{pmatrix} \varphi \cdot r_x \\ \varphi \cdot r_y \\ \varphi \cdot r_z \end{pmatrix}$$

Umgekehrt lässt sich bei gegebenem Rodrigues-Vektor $\vec{R}$ die Achse-Winkel-Repräsentation $(\varphi, \hat{r})$ der zugehörigen Drehung wie folgt berechnen:

$$\varphi = \|\vec{R}\|, \quad \hat{r} = \frac{\vec{R}}{\|\vec{R}\|}$$

Daraus lässt sich wiederum direkt das entsprechende Rotations-Quaternion konstruieren:

$$q = \begin{pmatrix} \cos(\frac{\varphi}{2}) \\ \sin(\frac{\varphi}{2}) \cdot \hat{r} \end{pmatrix}.$$

Man kann eine Rotation, die in Achse-Winkel-Repräsentation gegeben ist auch mit der folgend definierten **Rodrigues-Formel** berechnen:

Sei $(\varphi, \hat{r})$ die Achse-Winkel-Repräsentation der Rotation und sei $\vec{p} \in \mathbb{R}^3$ ein beliebiger Punkt im dreidimensionalen Raum. Dann ist der rotierte Punkt $\vec{p}'$ gegeben durch:

$$\vec{p}' = \vec{p} \cdot \cos\varphi + (\hat{r} \times \vec{p}) \cdot \sin\varphi + \hat{r} \cdot \langle \hat{r} | \vec{p} \rangle \cdot (1 - \cos\varphi) \tag{1.5}$$

Hierbei ist $(\cdot \times \cdot)$ das Kreuzprodukt im $\mathbb{R}^3$ und $\langle \cdot | \cdot \rangle$ das Skalarprodukt ebenfalls im $\mathbb{R}^3$.
Die zugehörige Rotations-Matrix $\Theta \in SO(3)$ lässt sich wie folgt berechnen:

$$\Theta = \mathbb{1} + \sin\varphi \cdot [\hat{r}]_\times + (1 - \cos\varphi) \cdot [\hat{r}]_\times^2 \tag{1.6}$$

mit der schiefsymmetrischen „Kreuzprodukt-Matrix“:

$$[\hat{r}]_\times := \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix},$$

die aus der Rotations-Achse $\hat{r}$ konstruiert wird. Die Multiplikation eines Vektors $\vec{v} \in \mathbb{R}^3$ von links mit dieser Matrix erzeugt das Kreuzprodukt $\hat{r} \times \vec{v}$:

$$[\hat{r}]_\times \cdot \vec{v} = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} r_y v_z - r_z v_y \\ r_z v_x - r_x v_z \\ r_x v_y - r_y v_x \end{pmatrix} = \hat{r} \times \vec{v}$$

Da $(\varphi, \hat{r})$ und $(2\pi - \varphi, -\hat{r})$ dieselbe Rotation beschreiben, tritt auch bei Rodrigues-Vektoren – ähnlich wie bei Rotations-Quaternionen – Antipodalität auf. Dies lässt sich mit Hilfe der Rodrigues-Formel für die Konstruktion der zugehörigen Rotations-Matrix (Gleichung (1.6)) beweisen:

$$\begin{aligned}
\Theta_{(\varphi,\hat{r})} &= \mathbb{1} + \sin\varphi \cdot [\hat{r}]_\times + (1 - \cos\varphi) \cdot [\hat{r}]_\times^2 \\
\Theta_{(2\pi-\varphi,-\hat{r})} &= \mathbb{1} + \sin(2\pi - \varphi) \cdot ([-\hat{r}]_\times) + (1 - \cos(2\pi - \varphi)) \cdot ([-\hat{r}]_\times)^2 \\
&= \mathbb{1} + \sin(2\pi - \varphi) \cdot (-[\hat{r}]_\times) + (1 - \cos(2\pi - \varphi)) \cdot (-[\hat{r}]_\times)^2 \\
&= \mathbb{1} + (-\sin\varphi) \cdot (-[\hat{r}]_\times) + (1 - \cos\varphi)(-1)^2 \cdot [\hat{r}]_\times^2 \\
&= \mathbb{1} + \sin\varphi \cdot [\hat{r}]_\times + (1 - \cos\varphi) \cdot [\hat{r}]_\times^2 \\
&= \Theta_{(\varphi,\hat{r})}
\end{aligned}$$

Hierbei wurden folgende Identitäten verwendet:

$$\sin(2\pi - \alpha) = -\sin\alpha, \quad \cos(2\pi - \alpha) = \cos\alpha, \quad [-\vec{x}]_\times = -[\vec{x}]_\times.$$

Vorteile bei der Verwendung von Rodrigues-Vektoren

+ Wie auch bei den Euler-Winkeln, kommen Rodrigues-Vektoren mit 3 Elementen aus. Damit ist auch hier die Anzahl der Parameter gleich der inhärenten Anzahl der Parameter einer Rotation im dreidimensionalen Raum und damit minimal. Hierdurch ist der Speicherbedarf auch bei der Verwendung von Rodrigues-Vektoren minimal.

+ Der Rechenaufwand für die Inversen-Bildung ist ebenfalls minimal: es müssen lediglich alle drei Vorzeichen von $\vec{R}$ gewechselt werden, da $-\vec{R}$ genau die umgekehrte Rotation zu $\vec{R}$ darstellt

+ Mit Hilfe der Rodrigues-Formel (Gleichung (1.5)) lässt sich eine Rotation, die als Rodrigues-Vektor gegeben ist, direkt auf einen Vektor im dreidimensionalen Raum anwenden.

Nachteile bei der Verwendung von Rodrigues-Vektoren

- Auch bei Rodrigues-Vektoren ist man für die Konkatenation von mehreren Rotationen auf alternative Darstellungen (z.B. Matrizen oder Quaternionen) angewiesen.

- Der relative Winkel zwischen zwei Rotationen kann auch bei Rodrigues-Vektoren wieder nur über den Zwischenschritt in eine andere Darstellung berechnet werden.
- Analog zu Rotations-Quaternionen kann eine Rotation immer jeweils durch zwei antipodale Rodrigues-Vektoren beschrieben werden, weshalb diese Darstellung ebenfalls nicht eindeutig ist.

1.2.5 Fazit

Da sich diese Arbeit vor allem mit Abstandsmaßen zwischen Rotationen beschäftigt, ist eine Repräsentation von Rotationen sinnvoll, mit deren Hilfe sich solche Abstandsmaße definieren lassen. Wie oben beschrieben, müssen sowohl Euler-Winkel als auch Rodrigues-Vektoren zuerst in eine andere Darstellung konvertiert werden, um eine relative Drehung von einer Ausgangslage in eine Ziellage beschreiben zu können. Deshalb sind diese Darstellungen für die Verwendung in dieser Arbeit nicht gut geeignet. Bei Rotations-Matrizen und ebenso bei Rotations-Quaternionen lässt sich der „Abstand“ zwischen zwei Rotationen direkt berechnen, weshalb diese Darstellungen besser geeignet sind. Bei Rotations-Quaternionen ist zum einen – wie bereits weiter oben dargestellt – der Speicherbedarf geringer als bei Rotationsmatrizen. Es wird weniger als die Hälfte an Speicherplatz benötigt, um Quaternionen zu speichern, als bei Rotations-Matrizen erforderlich wäre. Außerdem ist der Rechenzeitbedarf z.B. für die Inversen-Bildung und die Renormierung aufgrund akkumulierter numerischer Fehler nach wiederholten Rechenoperationen bei Quaternionen deutlich geringer, als dies bei Rotations-Matrizen der Fall ist. Deshalb wird im weiteren Verlauf dieser Arbeit hauptsächlich die **Darstellung von Rotationen mit Hilfe von Quaternionen** verwendet.

1.3 Verwendete Datensätze

In dieser Arbeit werden – neben selbst aufgenommenen Daten – die im Folgenden vorgestellten Datensätze verwendet.

1.3.1 Der Opportunity-Datensatz

Der als „OPPORTUNITY Activity Recognition Dataset“ [10, 11] im „UC Irvine Machine Learning Repository“[(2)] online verfügbare Datensatz[(3)] besteht aus einer Vielzahl verschiedener Sensordaten. Hierbei wurden insgesamt 4 Testpersonen mit verschiedenen Sensoren bei der Ausübung typischer alltäglicher Tätigkeiten in einer nachgebauten Wohnungsumgebung vermessen. In dieser Testumgebung gab es eine Küche mit verschiedenen Geräten wie Spülmaschine, Kaffeemaschine und Kühlschrank, einen Liegestuhl, einen Tisch mit Stuhl, sowie Türen, die nach draußen führen. Dabei kamen unter anderem folgende Sensoren zum Einsatz: 7 Inertial Measurement Units (IMUs), 12 3D-Accelerometer und 4 Marker für ein Messsystem, welches die Absolutposition eines Objekts im Raum bestimmen kann. Diese Sensoren wurden an verschiedenen Stellen des Körpers angebracht (Oberarme, Unterarme, Rücken, Hüfte, Füße). Weitere Sensoren waren an verschiedenen Objekten und Geräten in der Umgebung angebracht. Während die 3D-Accelerometer ausschließlich (lineare) Beschleunigungsdaten liefern, kann mit IMUs die Orientierung im dreidimensionalen Raum gemessen werden. Die Daten sind manuell gelabelt auf verschiedenen Ebenen. Auf der untersten Ebene wird jeweils zwischen „stehen“, „laufen“, „sitzen“ und „liegen“ unterschieden. Die Bewegungen selbst werden dann hierarchisch in „low-level“-, „mid-level“- und „high-level“-Aktivitäten eingeordnet. So sind einfache („low-level“) Grundbewegungen wie: „greifen“, „öffnen“ oder „schließen“ für beide Hände einzeln und zusammen gelabelt. Dann gibt es abstrakte („high-level“) Aktivitäten wie z.B. „Ausruhen“, „Kaffee-Pause“ und „Aufräumen“. In dieser Arbeit werden die folgenden 17 „mid-level“-Klassen betrachtet:

- Open Door 1 (öffne Tür 1)
- Close Door 1 (schließe Tür 1)
- Open Door 2 (öffne Tür 2)
- Close Door 2 (schließe Tür 2)
- Open Fridge (öffne Kühlschrank)
- Close Fridge (schließe Kühlschrank)
- Open Dishwasher (öffne Spülmaschine)
- Close Dishwasher (schließe Spülmaschine)
- Open Drawer 1 (öffne Schublade 1)
- Close Drawer 1 (schließe Schublade 1)
- Open Drawer 2 (öffne Schublade 2)
- Close Drawer 2 (schließe Schublade 2)
- Open Drawer 3 (öffne Schublade 3)
- Close Drawer 3 (schließe Schublade 3)

(2) https://archive.ics.uci.edu/ml/index.html
(3) https://archive.ics.uci.edu/ml/datasets/OPPORTUNITY+Activity+Recognition

- Clean Table (reinige Tisch)
- Drink From Cup (trinke aus Tasse)
- Toggle Switch (betätige Lichtschalter)

Die reinen Accelerometer können lediglich lineare Beschleunigungen messen und liefern keine Rotationsdaten. Da später hauptsächlich die Rotationsinformationen von Interesse sein werden, kommen für die Verwendung in dieser Arbeit nur die IMUs in Betracht. Die beiden IMUs an den Füßen spielen für die ausgewählten Klassen keine Rolle, da alle betrachteten Bewegungen mit dem Oberkörper durchgeführt wurden. Aus diesem Grund werden ausschließlich die Daten der folgenden 5 Sensoren verwendet (siehe Abb. 1.3):

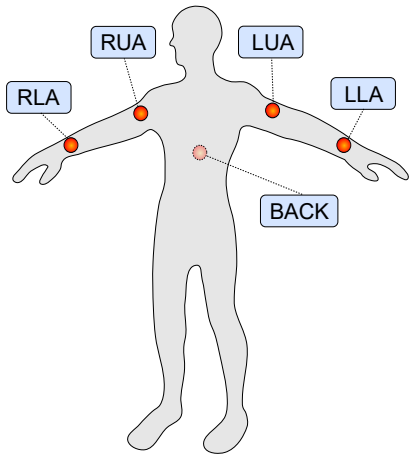
- rechter Oberarm (right upper arm – RUA)
- rechter Unterarm (right lower arm – RLA)
- linker Oberarm (left upper arm – LUA)
- linker Unterarm (left lower arm – LLA)
- Rücken (back - BACK)

Die 4 Test-Personen haben in jeweils 6 Durchläufen die oben angegebenen 17 verschiedenen Bewegungen durchgeführt. In 5 dieser Durchläufe wurde jeweils ein natürlicher Tagesablauf simuliert. Dann wurde ein zusätzlicher „Drill"-Durchlauf durchgeführt, in dem eine fest vorgegebene Bewegungsfolge abgearbeitet wurde. Auf diese Weise wurde sichergestellt, dass von jeder Klasse hinreichend viele Instanzen vorliegen. Die verwendeten Orientierungen sind jeweils als Quaternionen relativ zu einem (festen) Weltkoordinatensystem gegeben. In einem Preprocessing-Schritt wurden aus den gegebenen Daten-Streams die einzelnen Bewegungen ausgeschnitten. Fehlerhafte Messungen (z.B. durch Sensorausfälle) wurden dabei direkt aussortiert. Insgesamt hat der aus diesem Datensatz in dieser Arbeit verwendete Teil pro Sensor 17 Klassen mit je zwischen 100 und 270 Instanzen mit einer mittleren Länge von 95 Datenpunkten. Insgesamt besteht der Datensatz pro Sensor aus 2542 Shapes.

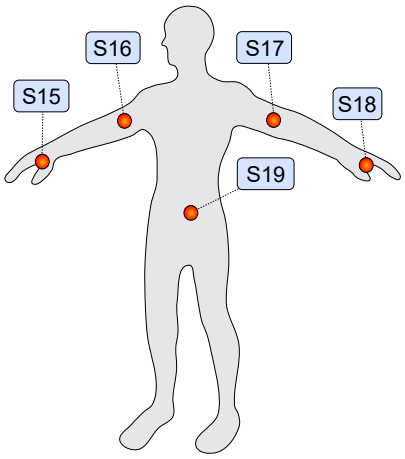
1.3.2 Der QBGR-Datensatz

Alavi et. al haben in ihrer Arbeit „Quaternion-Based-Gesture-Recognition" [12] eine Multi-Sensor Umgebung entwickelt, mit der insgesamt 11 Probanden mit 5 verschiedenen IMU-Sensoren bei verschiedenen Sport-Übungen vermessen wurden. Die Daten sind online öffentlich zugänglich[(4)]. Folgende Bewegungen wurden dabei aufgezeichnet:

[(4)] www.mdpi.com/1424-8220/16/5/605/s1



(a) Sensorkonfiguration: Opportunity-Datensatz



(b) Sensorkonfiguration: QBGR-Datensatz

Abbildung 1.3: (a): Konfiguration der IMU-Sensoren des Opportunity-Datensatzes: Zwei IMU-Sensoren wurden jeweils an den Armen angebracht, einer am Oberarm und einer am Unterarm nahe der Hand. Ein zusätzlicher IMU-Sensor wurde am Rücken befestigt. Die IMUs an den Füßen sowie sämtliche reinen Accelerometer werden in dieser Arbeit nicht betrachtet und sind deshalb nicht eingezeichnet. (b): Die Konfiguration der Sensoren des QBGR-Datensatzes ist sehr ähnlich zu der Konfiguration des Opportunity-Datensatzes. Die Sensoren sitzen auf den Handflächen und an den Armen. Ein weiterer Sensor wurde am unteren Bauch angebracht.

- Jab (horizontaler Schlag)
- Uppercut (Aufwärts-Haken)
- Throw (Wurf)
- Lift (Hantel anheben)
- Block (Blocken)
- Sway (horizont. Bewegung mit Hantel)

Die Sensor-Konfiguration ist sehr ähnlich wie beim Opportunity-Datensatz. Auch hier sind jeweils an den Unter- und Oberarmen IMU-Sensoren angebracht. Ein fünfter IMU-Sensor befand sich vorne etwa auf Höhe des Bauchnabels. Insgesamt wurden folgende Sensoren verwendet (siehe Abb. 1.3):

- rechte Handfläche (S15)
- rechter Arm (S16)
- linker Arm (S17)
- linke Handfläche (S18)
- unterer Bauch (S19)

Die Personen haben die Bewegungen jeweils in 4 bis 6 Wiederholungen durchgeführt. Ein Shape ist also die 4- bzw. 6-malige Wiederholung einer Grundbewegung. Jede Grundbewegung (außer „Sway“) wurde dabei jeweils getrennt mit der rechten Hand und mit der linken

Hand ausgeführt. Die andere Hand wurde dabei jeweils still gehalten. Deshalb wurde im Rahmen dieser Arbeit bei den Sensoren S15, S16 (rechter Arm) und S17, S18 (linker Arm) jeweils unterschieden zwischen aktiver und passiver Messung. Nur bei der aktiven Messung wird die Bewegung mit dem entsprechenden Arm aktiv ausgeführt, während er bei der passiven Messung in Ruheposition ist. Für den Sensor am unteren Bauch (S19) wurde diese Unterscheidung nicht getroffen. Die Unterscheidbarkeit der hauptsächlich mit den Armen durchgeführten Bewegungen ist bei diesem Sensor jedoch generell geringer als bei den Sensoren, die an den Armen befestigt wurden. In einen Preprocessing-Schritt wurden fehlerhafte Messungen (z.B. durch Sensorausfälle) entfernt. Des Weiteren gab es unnötige Redundanzen in den Messungen, sodass der selbe Messpunkt jeweils mehrfach hintereinander vorhanden war. Diese Redundanzen wurden durch ein einfaches Schwellwert-Verfahren entfernt.

Insgesamt hat der QBGR-Datensatz pro Sensor 6 Klassen mit je ca. 100 Instanzen (130 bei „Sway“) mit einer mittleren Länge von 99 Datenpunkten. Der Sensor S19 hat durch die fehlende Unterscheidung zwischen aktiver und passiver Messung ca. 200 Instanzen pro Klasse (auch hier: 130 Instanzen bei „Sway“). Insgesamt besteht der Datensatz pro Sensor aus 633 bis 659 Shapes (S15 - S18). Für Sensor S19 existieren 1186 Shapes.

1.3.3 Der 6DMG-Datensatz

In ihrer Arbeit „6DMG: A New 6D Motion Gesture Database“[13] haben Chen et al. mit Hilfe eines optischen Tracking-Systems und einer Nintendo Wii-Fernbedienung (Wii Remote) verschiedene elementare Gesten aufgezeichnet. Die Wii Remote wird dabei von den Probanden in der Hand gehalten, mit der die Gesten durchgeführt werden. Die Position und Orientierung der Wii Remote wird dabei sowohl über die eingebauten Inertial-Sensoren, als auch über ein externes optisches System getrackt. Auch dieser Datensatz ist öffentlich zugänglich[(5)].

Die Wii Remote verfügt – wie eine IMU – ebenfalls über ein 3-Achsen-Accelerometer und (dank der Motion-Plus-Erweiterung) über ein Gyroskop. Es werden bei diesem Datensatz sowohl die Orientierung der Wii Remote aus den Gyroskop-Daten, als auch deren Absolut-Position im Raum durch zweifache Zeit-Integration der Accelerometer-Daten bestimmt und diese werden mit dem optischen System fusioniert. Zu diesem Zweck wurde ein Infrarot-LED-Marker an der Wii Remote angebracht, die von einem stationären Kamerasystem getrackt wird. Auf diese Weise werden sowohl Absolut-Positionsdaten, als auch Orientierungs-Daten, sowie Winkel-Geschwindigkeiten und lineare Beschleunigungen aufgezeichnet, wobei wir uns in dieser Arbeit lediglich auf die Orientierungs-Daten konzentrieren wollen. Insgesamt wurden 28 Probanden (davon 21 Rechtshänder und 7 Linkshänder) auf-

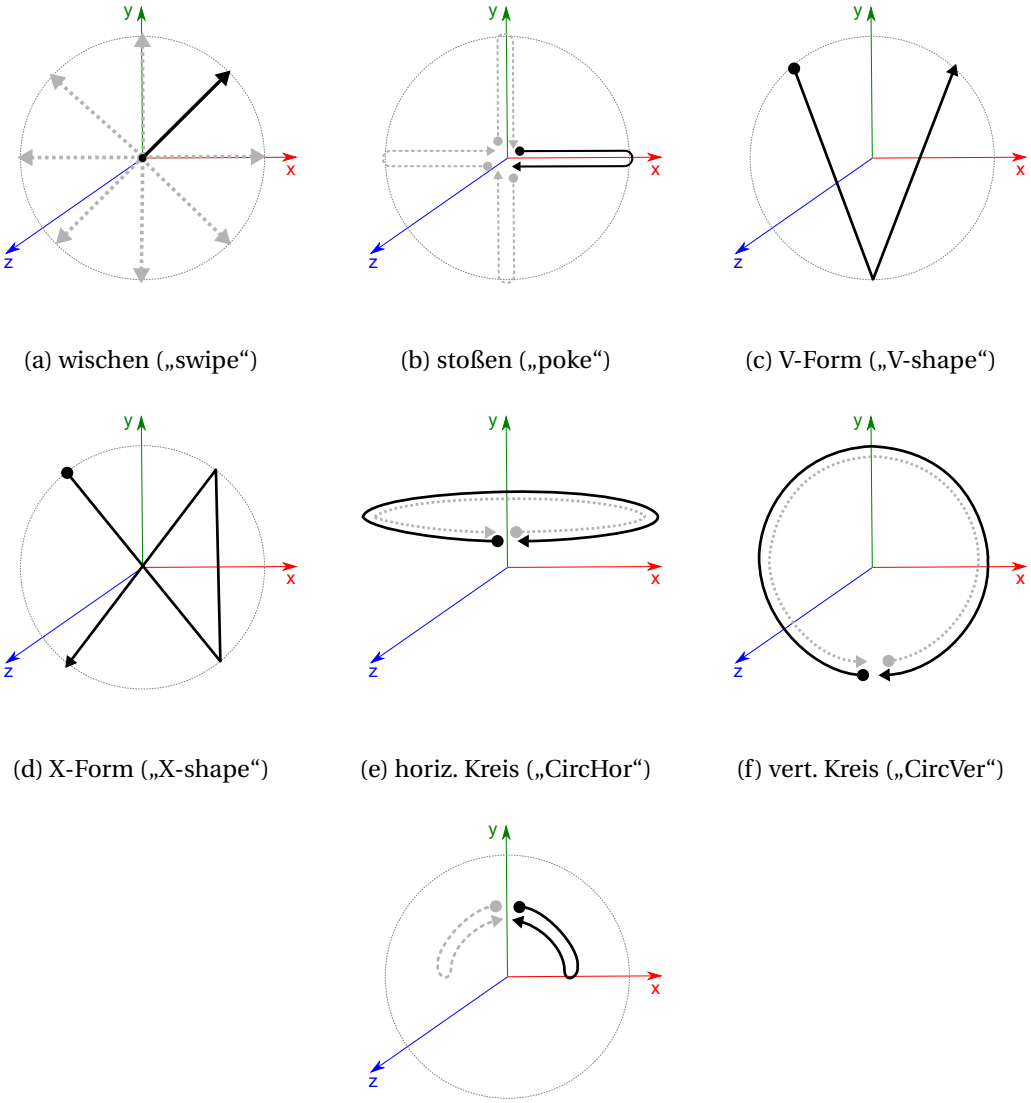
(5) http://www.ece.gatech.edu/6DMG

gefordert, je (mindestens) 10 mal jede der folgenden Gesten mit der Wii-Fernbedienung auszuführen:

- 8× wischen – in je eine von 8 Richtungen („swipe“)
- 4× stoßen – in je eine von 4 Richtungen („poke“)
- 1× V-Form („V-shape“)
- 1× X-Form („X-shape“)
- 2× Kreisbewegung horizontal – im UZS[6] und gegen den UZS („circle horizontal“)
- 2× Kreisbewegung vertikal – im UZS und gegen den UZS („circle vertical“)
- 2× drehen des Handgelenks – nach links und nach rechts („twist“)

In Abb. 1.4 sind die Gesten schematisch dargestellt. Es existieren damit 20 Klassen und insgesamt 5615 Shapes. Die Gesten wurden mit einer Frequenz von 60 Hz aufgezeichnet. Es existiert bei diesem Datensatz nur ein einziger Sensor. Die Daten sind von den Autoren noch einmal in linkshändige Probanden (L) und rechtshändige Probanden (R) unterteilt. In dieser Arbeit werden jedoch alle Daten gemeinsam betrachtet (LR), d.h. es wird nicht zwischen Links- und Rechtshändern unterschieden.

[6] „UZS“ steht hier für „Uhrzeigersinn“.



(a) wischen („swipe“)

(b) stoßen („poke“)

(c) V-Form („V-shape“)

(d) X-Form („X-shape“)

(e) horiz. Kreis („CircHor“)

(f) vert. Kreis („CircVer“)

(g) drehen des Handgelenks („Twist“)

Abbildung 1.4: Die verschiedenen Gesten-Klassen des 6DMG-Datensatzes

1.3.4 Der JGU_Numbers-Datensatz

Im Rahmen dieser Arbeit wurde der folgende – „JGU_Numbers“[(7)] genannte – Datensatz erzeugt. Es wurden insgesamt 20 Probanden aufgefordert, die Ziffern von ’0’ bis ’9’ jeweils 10 mal auf ein DIN-A4-Blatt zu schreiben. Um diese als „Gesten“ besser unterscheiden zu können, sollten die Ziffern jeweils recht groß geschrieben werden. An der rechten Hand trugen die Probanden einen Handschuh, an dem ein aktiver optischer IR-LED-Marker befestigt war. Die Handbewegungen wurden während des Schreibens mit einer IR-Kamera gefilmt und mit Hilfe der von der Hochschule Rhein-Main in Wiesbaden entwickelten Tracking-Technologie (HSRM-Tracking: [1]) die Posen (6DoF: 3 Rotations-Freiheitsgrade und 3 Translations-Freiheitsgrade) aufgezeichnet. Damit für alle Probanden der gleiche Messaufbau verwendet werden konnte, wurden alle „Gesten“ – unabhängig davon, ob die Person Rechts- oder Linkshänder ist – mit der rechten Hand durchgeführt. Der Messaufbau und der verwendete IR-LED-Marker sind in Abb. 1.5 dargestellt.

Die Posen wurden mit einer Rate von 60 Hz aufgenommen, da dies die maximale Frame-Rate der verwendeten Kamera bei der verwendeten Auflösung von 1280×1024 Pixeln ist. Aus den aufgenommenen Daten wurden alle fehlerhaften Shapes entfernt. Fehlerhafte Messungen kamen beispielsweise dadurch zu Stande, dass die Hand zu sehr in Richtung Tisch gedreht wurde, und die Kamera deshalb nicht alle LEDs des Markers sehen konnte. Damit das optische Tracking die Pose des Markers korrekt erkennen kann, ist es jedoch erforderlich, dass alle 7 IR-LEDs von der Kamera erfasst werden – siehe auch [1]. Eine weitere Fehlerquelle war ein falsches (manuelles) Starten bzw. Stoppen der Aufnahme, sodass eine Bewegung gar nicht oder nicht komplett erfasst wurde.

Insgesamt hat dieser Datensatz 10 Klassen (die 10 Ziffern von ’0’ bis ’9’) und 2003 Shapes. In Abb. 1.6 sind exemplarisch jeweils 10 Instanzen jeder Klasse dargestellt. Für die Klassifikation (siehe Kapitel 2) wurden ausschließlich die Rotationsanteile der Posen verwendet. Die Shapes haben eine Länge im Bereich von ca. 60 bis 180 Datenpunkten, was bei der verwendeten Aufnahmefrequenz von 60 Hz einer Aufnahmedauer von ca. 1 bis 3 Sekunden pro Shape entspricht. Der Datensatz enthält einen einzigen Sensor an der rechten Hand (RH).

[(7)] „JGU“ steht hierbei für „Johannes Gutenberg-Universität“

Abbildung 1.5: JGU_Numbers: [oben] Messaufbau mit IR-Kamera und IR-LED-Marker, [unten] Detailansicht: Handschuh mit IR-LED-Marker

Abbildung 1.6: JGU_Numbers: Beispiele für die aufgenommenen Zeitreihen der Ziffern von ’0’ bis ’9’. In jedem Bild sind exemplarisch 10 Instanzen der entsprechenden Ziffer als 3D-Kurve Dargestellt. Die kleinen Koordinatenkreuze repräsentieren die Orientierung des Markers zu jedem Zeitpunkt. (Weitere Informationen zur verwendeten 3D-Darstellung der 6DoF-Posen finden sich im nächsten Kapitel.)

1.3.5 Der qCBF-Datensatz

Zusätzlich zu den bisher vorgestellten realen Datensätzen wird noch ein weiterer, synthetisch erzeugter Datensatz verwendet. Der qCBF-Datensatz (quaternion Cylinder-Bell-Funnel) ist eine modifizierte Variante des CBF-Datensatzes [3], welche den ursprünglich nur für eindimensionale Zeitreihen vorgesehenen CBF-Datensatz auf den Raum der Quaternionen erweitert. Hierbei wird eine Rotation um eine willkürlich festgelegte Achse $\vec{l}$ simuliert. Der Rotations-Winkel α_i wird dabei gemäß der Definition von CBF erzeugt, sodass eine Rotation um die Achse $\vec{l}$ entsteht, deren Winkel im zeitlichen Verlauf den CBF-Kurven entspricht. Jede auf diese Weise erzeugte Bewegungs-Zeitreihe wird anschließend in ein zufällig gewähltes Koordinaten-System wie folgt transformiert: Jedes Rotations-Quaternion wird um eine zufällig gewählte (aber für die jeweilige Zeitreihe feste) Achse um einen zufällig uniform im Intervall $\varphi \in [0, 2\pi]$ gewählten (ebenfalls für die jeweilige Zeitreihe festen) Winkel rotiert:

Definition 5: qCBF

Sei φ ein zufällig aus dem Intervall $[0, 2\pi]$ gewählter Winkel und $\vec{r} = (r_x, r_y, r_z)^\top$ mit $\|\vec{r}\| = 1$ eine zufällig gewählte normierte Rotations-Achse im $\mathbb{R}^3$, dann ist qCBF definiert als:

$$\text{qCBF}(\alpha_i, \vec{l}, \varphi, \vec{r}) := q_{\text{CBF}}(\alpha_i, \vec{l}) * q_{\text{trans}}(\varphi, \vec{r}) = \begin{pmatrix} \cos(\frac{\alpha_i}{2}) \\ \sin(\frac{\alpha_i}{2}) \cdot l_x \\ \sin(\frac{\alpha_i}{2}) \cdot l_y \\ \sin(\frac{\alpha_i}{2}) \cdot l_z \end{pmatrix} * \begin{pmatrix} \cos(\frac{\varphi}{2}) \\ \sin(\frac{\varphi}{2}) \cdot r_x \\ \sin(\frac{\varphi}{2}) \cdot r_y \\ \sin(\frac{\varphi}{2}) \cdot r_z \end{pmatrix}$$

wobei $\vec{l} = (l_x, l_y, l_z)^\top$ mit $\|\vec{l}\| = 1$ eine beliebig zu wählende normierte Rotations-Achse ist, die für alle zu generierenden Shapes gleich ist. Die Winkel α_i werden dabei gemäß CBF generiert (siehe Gleichungen (1.1), (1.2) und (1.3)).

Für den vorliegenden Datensatz wurde um die z-Achse rotiert: $\vec{l} = (0, 0, 1)^\top$. Die CBF-Parameter (siehe Gleichungen (1.1), (1.2) und (1.3)) für die Winkel α_i wurden festgelegt zu:

Parameter	Zeichen[8]	Wert	Anmerkung
Basis-Amplitude	A	π	
Amplituden-Modulations-Parameter	η	$\eta_0 \cdot g_{[\mu=0,\sigma=1]}$[9]	mit $\eta_0 = \frac{\pi}{5}$
Amplitude des Gauß-Rauschens	ε_i	$\varepsilon_0 \cdot g_{[\mu=0,\sigma=1],i}$[9]	mit $\varepsilon_0 = \frac{\pi}{50}$
Zeitreihen-Länge	L	128	

Tabelle 1.1: Parameter für die Generierung des qCBF-Datensatzes

[8] vgl. Gleichungen (1.1), (1.2) und (1.3)

[9] Die Funktion $g_{[\mu,\sigma]}$ ist hierbei die Gaußsche Normalverteilungs-Funktion mit Erwartungswert μ und Standardabweichung σ)

Insgesamt wurden für jede der drei Klassen „Cylinder“, „Bell“ und „Funnel“ je 500 Shapes generiert. Damit besteht dieser Datensatz aus insgesamt 1500 Shapes. Auch hier gibt es nur einen einzigen Teildatensatz („Sensor A“). In Abb. 1.7 sind von jeder Klasse beispielhaft je fünf Instanzen dargestellt.

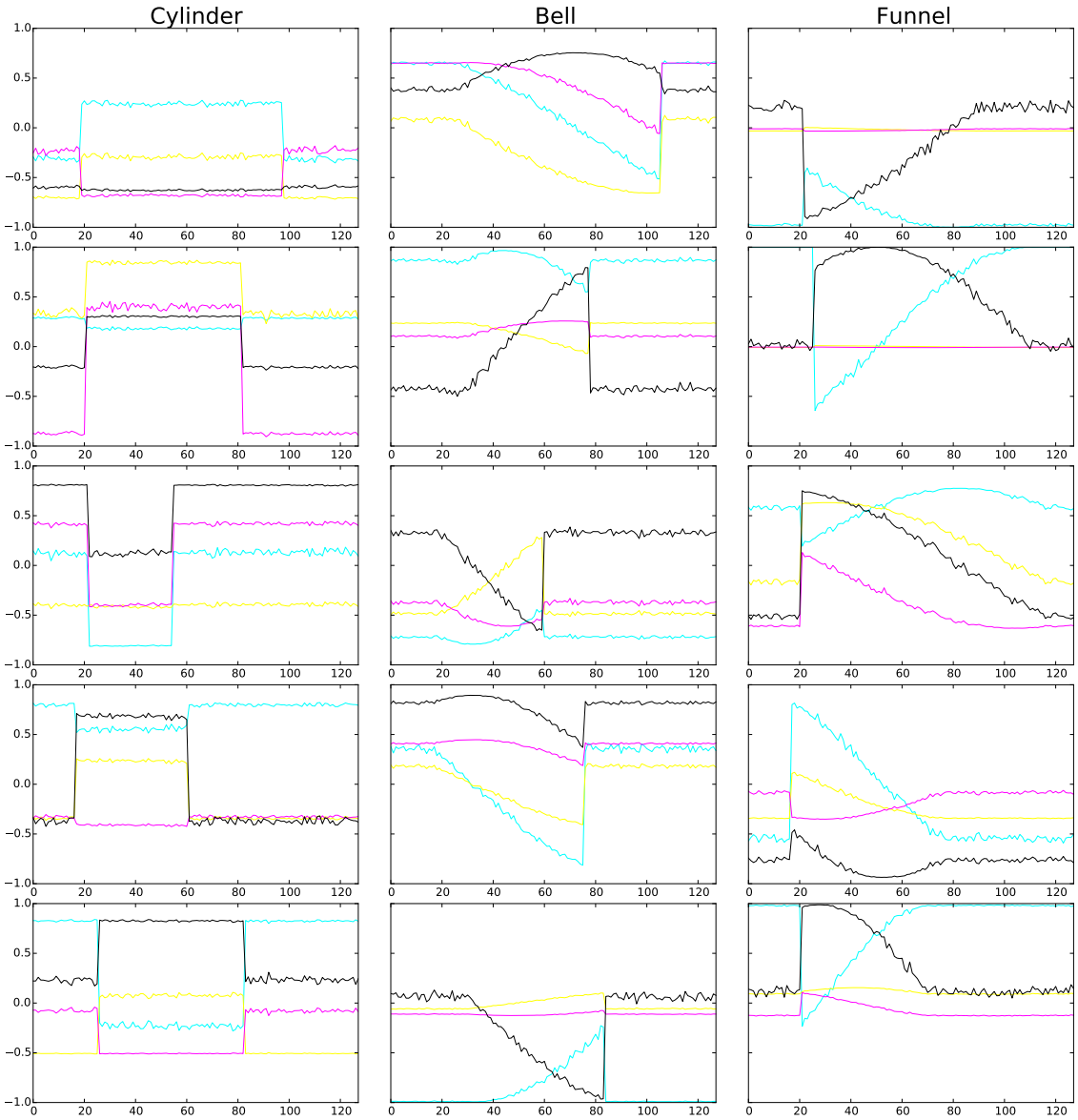


Abbildung 1.7: Beispiel-Shapes für qCBF (quaternion Cylinder-Bell-Funnel). Zu jeder Klasse sind hier exemplarisch fünf Beispiel-Shapes dargestellt. Die Länge der Shapes beträgt jeweils $L = 128$ Zeitschritte. Es wurden jeweils alle vier Quaternionen-Komponenten in ein Diagramm gezeichnet. Hierbei wurde folgende Farbcodierung für die einzelnen Komponenten gewählt: q_w: cyan, q_x: gelb, q_y: magenta, q_z: schwarz.

1.3.6 Übersicht: Verwendete Datensätze

Tabelle 1.2 zeigt eine Übersicht der in dieser Arbeit verwendeten Datensätze.

Datensatz	# Sensoren	# Klassen	# Instanzen / Klasse	# Shapes (gesamt)	Art[10]
Opportunity	5	17	100 – 270	12710	R
QBGR	5 (9)[11]	6	≈100 (S19: ≈ 200)[12]	5875	R
6DMG	1	20	≈280	5615	R+T
JGU_Numbers	1	10	≈200	2003	R+T
qCBF	1	3	500	1500	R

Tabelle 1.2: Übersicht der in dieser Arbeit verwendeten Datensätze

[10] R = nur Rotation (3 DoF), R+T = Rotation + Translation (6 DoF)

[11] Insgesamt gibt es bei diesem Datensatz 5 physisch vorhandene Sensoren. Die 4 Sensoren an Händen und Armen werden jedoch jeweils noch einmal logisch unterteilt in Sensoren am aktiven Arm und Sensoren am inaktiven Arm (siehe Unterabschnitt 1.3.2).

[12] Die Klasse „Sway“ hat für jeden Sensor (inkl. S19) immer 130 Instanzen / Klasse

KAPITEL 2

BEWEGUNGSERKENNUNG UND STREAM-SEGMENTIERUNG

2.1 Visualisierung

Eine Posen-Zeitreihe hat – je nach Darstellung – mindestens 6 Dimensionen: 3 räumliche Positions-Koordinaten und 3 Rotations-Freiheitsgrade. Bei der Darstellung der Rotation mit Hilfe von Quaternionen werden für die Rotation sogar 4 Komponenten benötigt. Um sich einen ersten visuellen Überblick über eine gemessene Zeitreihe zu verschaffen, kann man beispielsweise jede dieser Komponenten einzeln graphisch darstellen (siehe Abb. 2.1). Auf diese Weise kann man z.B. erkennen, an welchen Stellen eine Segmentierung stattfinden könnte und dies dann später vergleichen mit automatisiert gefundenen Segmentierungen.

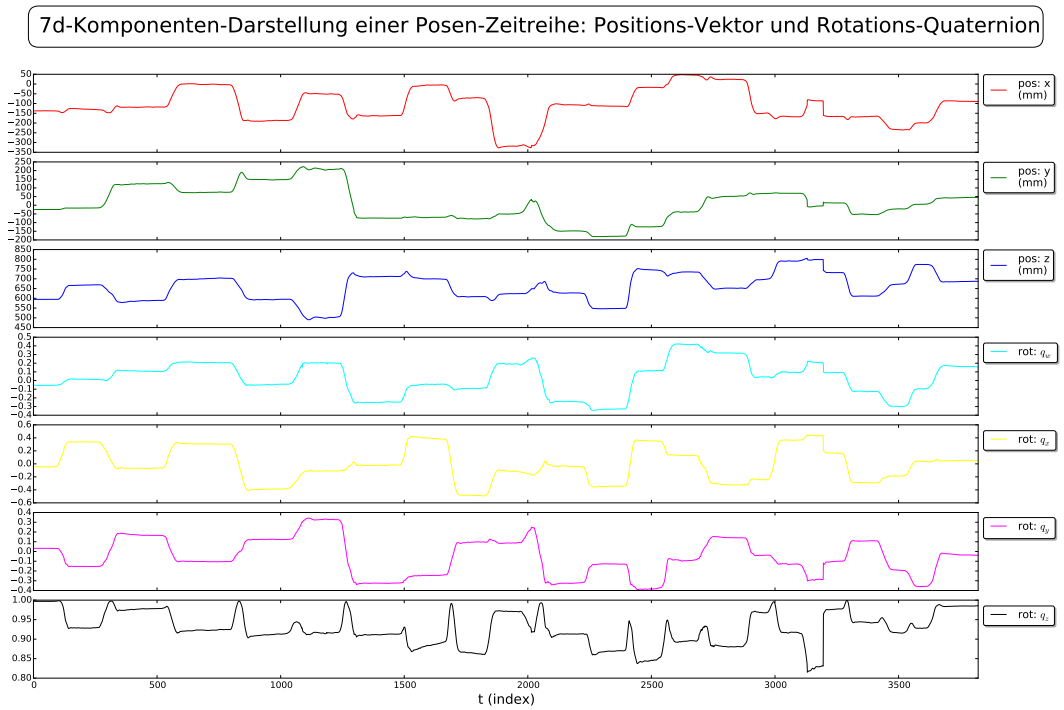


Abbildung 2.1: 7D-Darstellung einer Posen-Zeitreihe. Jede einzelne Komponente wird getrennt visualisiert. Im oberen Bereich werden die 3 Komponenten des Positions-Vektors $\vec{p} = (x, y, z)^\top$, im unteren Bereich die 4 Komponenten des Rotations-Quaternions $q = (q_w, q_x, q_y, q_z)^\top$ jeweils einzeln graphisch dargestellt. Man kann sich auf diese Weise z.B. einen ersten Eindruck über eine mögliche Segmentierung der Zeitreihe verschaffen.

Um einen besseren räumlichen Eindruck von einer Posen-Zeitreihe zu erhalten bzw. eine Segmentierung oder ein Matching visuell bewerten zu können, wurde ein interaktives 3D-Visualisierungstool entwickelt (Abb. 2.2). Mit dessen Hilfe ist es möglich, Posen-Zeitreihen im dreidimensionalen Raum darzustellen. Die Positions-Komponente jeder Pose wird dabei als Punkt im Raum dargestellt. Diese Punkte sind durch Geraden-Segmente miteinan-

der verbunden. Die Orientierungs-Komponente jeder Pose wird mit einem Koordinaten-Dreibein, dessen Ursprung in diesem Punkt liegt, dargestellt. Die Größe dieser Achsen-Kreuze lässt sich anpassen. Dadurch lässt sich die Orientierung auch dann noch visualisieren, wenn man die Zeitreihe als Ganzes von einem weiter entfernten Punkt aus betrachtet. Auf diese Weise lässt sich sowohl die Trajektorie im Raum als auch die Orientierung an jedem Messpunkt in einem einzigen Graphen darstellen. Das Tool ermöglicht es, die Zeitreihen als Ganzes darzustellen und von allen Seiten zu betrachten. Hierzu wurde ein "virtueller Trackball"[14] implementiert, mit dessen Hilfe eine intuitive Drehungs-Einstellung der Szene mit der Maus möglich ist. Als Drehzentrum lässt sich hierbei ein beliebiger Punkt der Zeitreihe auswählen. Jeder Punkt der Zeitreihe lässt sich beliebig genau an die Position des Betrachters heranzoomen, sodass eine umfassende visuelle Bewertung jedes Messpunktes und dessen Nachbarschaft möglich ist. Man kann mehrere Zeitreihen gleichzeitig laden und in einem gemeinsamen Koordinatensystem betrachten. Um auch bei der gleichzeitigen Untersuchung mehrerer Zeitreihen die Übersichtlichkeit zu gewährleisten, lässt sich jede einzelne Zeitreihe „an-“ und „ausschalten“.

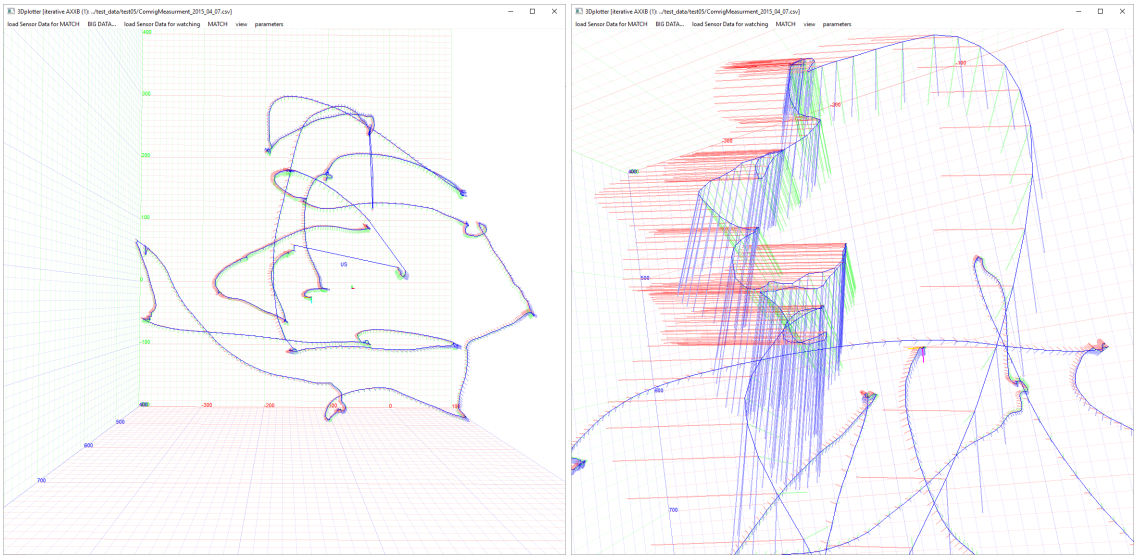


Abbildung 2.2: 3D-Plotter: gesamte Zeitreihe (links) und Detailansicht (rechts). Die Zeitreihen werden als durch Geradensegmente verbundene Punkte im dreidimensionalen Raum perspektivisch dargestellt. Die Orientierungen an jedem Punkt sind durch kleine Achsen-Kreuze dargestellt, deren Ausrichtung der Rotation der jeweiligen Pose entspricht. Man kann die „virtuelle Kamera“, aus deren Sicht man auf die Zeitreihe blickt, beliebig im Raum bewegen und auch die Orientierung frei wählen, sodass man jede Pose der Zeitreihe und auch deren Nachbarschaft visuell genau untersuchen kann.

2.2 Bewegungs-Segmentierung und einfaches Clustering

Hat man eine gemessene Zeitreihe vorliegen, so ist es meistens von Interesse, diese in kleinere Teil-Abschnitte zu segmentieren. Beispielsweise möchte man in einem EKG-Stream einzelne Herzschläge extrahieren oder in einer Sprachnachricht einzelne Worte. Bei einer Bewegungs-Zeitreihe ist man an einer Erkennung von Teilbewegungen interessiert. Man kann, wenn man während einer Bewegungsmessung ein Video mitschneidet, die Bewegung manuell segmentieren und die auf diese Weise gefundenen Bewegungs-Teilabschnitte jeweils manuell einer Grundbewegung bzw. Geste zuordnen („labeln“). Dies ist allerdings recht zeitaufwändig. Außerdem hängt das Ergebnis davon ab, wie derjenige, der die Labels manuell erstellt, die Bewegungen erkennt. Unterschiedliche Menschen beurteilen beispielsweise den genauen Start und das genaue Ende jeder Teilbewegung eventuell anders. Es stellt sich deshalb die Frage, wie eine solche Segmentierung *automatisiert* erfolgen kann.

2.2.1 Extremstellen-Suche

Eine einfache Möglichkeit, eine Bewegungs-Zeitreihe zu segmentieren, ist anhand von Extremstellen. Die intuitivste und einfachste Vorgehensweise Extremstellen zu finden ist es, die Zeitreihe (nach der Zeit) abzuleiten und anschließend die Nullstellen der Ableitung zu bestimmen. Die so gefundenen Extrema kann man dann beispielsweise anhand ihrer räumlichen Lage in sogenannte *Cluster* einteilen. Für dieses *Clustering* gibt es verschiedene Methoden von denen die einfachste und bekannteste wohl der *k-means*-Algorithmus [15] ist:

Beim *k-means*-Algorithmus werden zufällig Punkte aus der Eingabemenge als Cluster-Zentren gewählt und alle Punkte werden entsprechend ihrer (räumlichen) Distanz zu diesen Cluster-Zentren dem entsprechenden Cluster zugeordnet. Anschließend werden in einem iterativen Verfahren die Cluster-Zentren als Mittelwerte der Punkte des jeweiligen Clusters neu berechnet und dann die Punkte erneut zugeordnet. Das Verfahren konvergiert, sobald sich die Zuordnungen der Punkte von einem Iterations-Schritt auf den nächsten nicht mehr ändern. Der k-means Algorithmus muss die gewünschte Anzahl an Clustern a priori kennen und erzwingt diese Anzahl beim Clustering.

Die gefundenen Cluster können anschließend entsprechend sinnvoll mit Labels versehen werden. Im folgenden Beispiel wurde eine einfache Greifbewegung aus verschiedenen Positionen (2, 3, 4) auf einem Tisch zur Arbeitsfläche (1) simuliert (siehe Abb. 2.3). Dabei wurde jede Greifbewegung jeweils 4-mal durchgeführt. Zwischen den einzelnen Bewegungen wurde eine Montage von einzelnen Teilen an Position 1 (Arbeitsfläche) simuliert. Die Hand ruht also nicht an dieser Position, sondern es werden kleinere Handbewegungen durchgeführt, um einen Fertigungsprozess zu simulieren. Die Datenaufzeichnung erfolgte mit dem von der Firma soft2tec in Rüsselsheim entwickelten Nexonar-System [2], welches mit Ultraschall-Emittern als Marker und einer Anordnung von Ultraschall-Mikrofonen als Emp-

fänger arbeitet. Damit wurden die Positions- und Rotationsdaten des Handschuhs mit einer Frequenz von ca. 50 Hz aufgezeichnet. In diesem Beispiel könnte man die gefundenen Cluster dann z.B. mit „Arbeitsfläche / Position Nr. 1“, „Position Nr. 2“, „Position Nr. 3“ und „Position Nr. 4“ identifizieren.

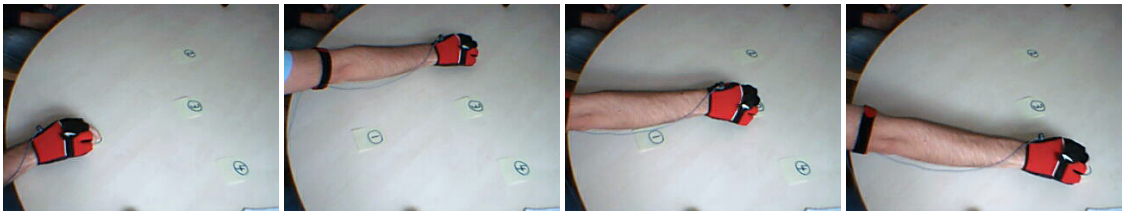


Abbildung 2.3: Test-Messung für eine Greifbewegung an verschiedenen Positionen auf einem Tisch. Es wird simuliert, wie z.B. Teile aus verschiedenen Boxen an den Positionen (2, 3, 4) zur Arbeitsfläche (1) mit einer Greifbewegung geholt werden. Jede Bewegung wurde jeweils 4-mal durchgeführt. Hierbei wurde die Position des Handschuhs und dessen Orientierung im Raum in 6 Dimensionen bei einer Frequenz von ca. 50 Hz erfasst.[(1)]

Gauß-Glättung

Bei der Suche der Extremstellen gibt es jedoch ein offensichtliches Problem: Dadurch, dass reale Messwerte immer mit Sensor-Rauschen behaftet sind, entstehen falsche, unerwünschte Extrema (siehe Abb. 2.4 (a)). Die Zeitreihe muss deshalb vor der Extremstellensuche erst einmal geglättet werden. Für die Glättung gibt es verschiedene Möglichkeiten, von denen die bekannteste Form die sogenannte *Gauß-Glättung* ist. Diese wird z.B. in der Bildverarbeitung verwendet, um Bilddaten zu glätten [16].

Definition 6: kontinuierliche Gauß-Glättung

Sei $f(t) : \mathbb{R}_0^+ \mapsto \mathbb{R}^N$ eine N-dimensionale Zeitreihe und sei $f_i(t) : \mathbb{R}_0^+ \mapsto \mathbb{R}$ die i-te Komponente dieser Zeitreihe, dann ist die **Gauß-geglättete** Zeitreihe $\tilde{f}(t) = \tilde{f}_i(t)$ durch eine komponentenweise **Faltung** mit der Gaußschen Normalverteilungs-Funktion

$$g_{[\mu,\sigma]} : \mathbb{R} \mapsto \mathbb{R}^+, g_{[\mu,\sigma]}(x) := \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

mit Mittelwert $\mu = 0$ definiert:

$$\tilde{f}_i(t) := \left(f_i * g_{[0,\sigma]}\right)(t) = \int_{-\infty}^{+\infty} f_i(t-\tau) \cdot g_{[0,\sigma]}(\tau) d\tau$$

(1) Das Bildmaterial, sowie die Daten wurden freundlicherweise zur Verfügung gestellt von der Firma soft2tec, Rüsselsheim.

Da wir es grundsätzlich mit Messdaten zu tun haben, die in diskreten Zeitschritten gemessen werden, brauchen wir eine diskrete Form der Gauß-Glättung. Für eine aus diskreten Messwerten bestehende Zeitreihe f_t entartet das Integral zu einer Summe und die Gauß-Funktion wird durch einen sogenannten **Gauß-Kernel** substituiert, der die Gauß-Funktion an (endlich vielen) diskreten Stützstellen um den Ursprung $t = 0$ abtastet:

Definition 7: diskrete Gauß-Glättung

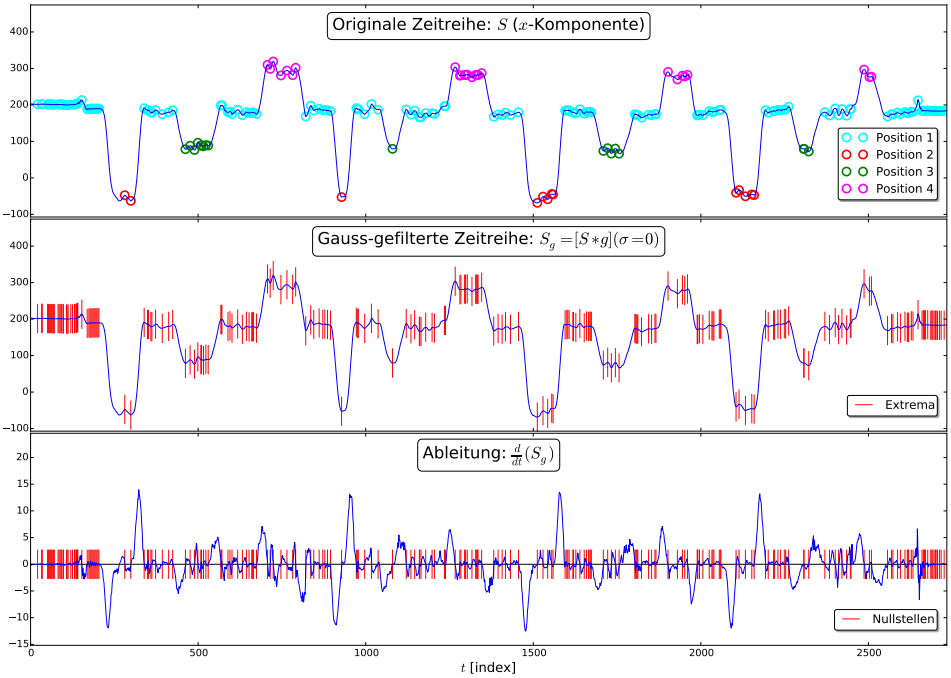
Sei $f_t = \{(t_0, f(t_0)), \ldots, (t_n, f(t_n))\}, \{t_0, \ldots, t_n\} \subset \mathbb{R}_0^+ \mapsto \mathbb{R}^N$ eine aus diskreten Messwerten bestehende, N-dimensionale Zeitreihe und sei $f_{i,t}$, $\{t_0, \ldots, t_n\} \subset \mathbb{R}_0^+ \mapsto \mathbb{R}$ deren i-te Komponente. Sei ferner $g_{[0,\sigma],t} = g_{[0,\sigma]}(t) \mid t \in \{t_{-k}, \ldots, t_k\}$ mit $\sum_{\tau=-k}^{k} g_{[0,\sigma],\tau} = 1$ ein diskreter **Gauß-Kernel**, dann ist die **Gauß-geglättete** Zeitreihe durch eine komponentenweise diskrete Faltung mit dem Gauß-Kernel definiert:

$$\tilde{f}_{i,t} := \left(f_i * g_{[0,\sigma]}\right)_t = \sum_{\tau=-k}^{k} f_{i,t-\tau} \cdot g_{[0,\sigma],\tau}$$

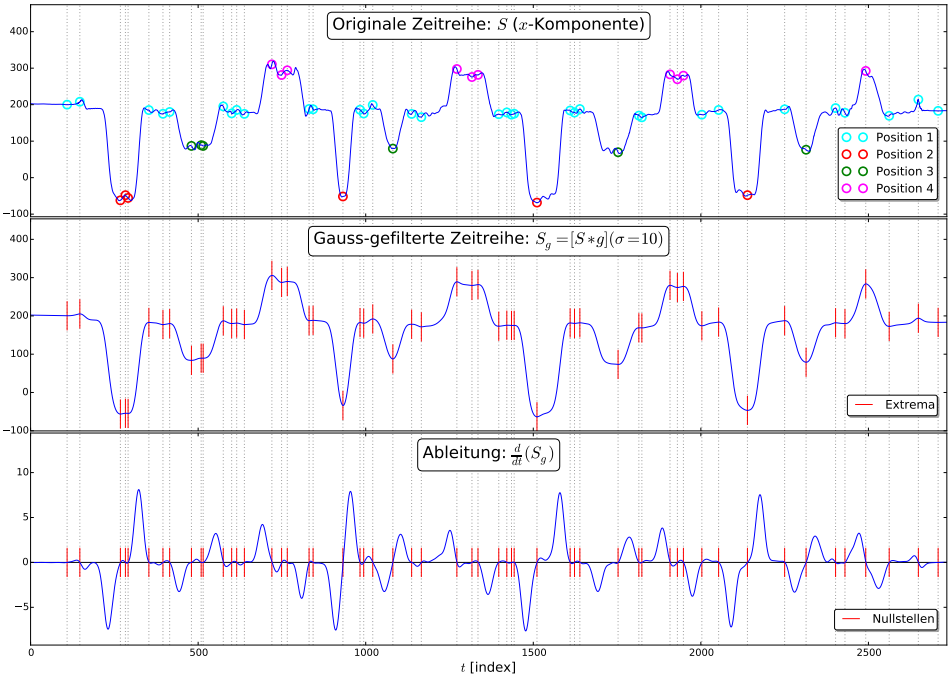
Ein solcher Gauß-Kernel $g_{[\mu,\sigma],t}$ wird durch zwei Parameter beschrieben: Der Mittelwert $\mu \in \mathbb{R}$ und die Standardabweichung $\sigma \in \mathbb{R}^+$. Während der Mittelwert μ des Kernels für eine Glättung stets 0 sein muss, um keine Verschiebung der Messwerte zu bewirken, lässt sich die Standardabweichung σ beliebig variieren. Da die Standardabweichung σ des Gauß-Kernels die Stärke der Glättung angibt, kann man sie auch als *Glättungsparameter* bezeichnen. Die richtige Wahl des Glättungsparameters ist entscheidend für die Qualität des Clustering-Ergebnisses. Um dies anschaulich darzustellen, sind in Abb. 2.4 die Clustering-Ergebnisse für die oben beschriebene simulierte Greifbewegung im eindimensionalen Fall für einige verschiedene Glättungsparameter exemplarisch aufgeführt.

Wir versuchen nun die Stellen der Zeitreihe zu identifizieren, bei denen in eine der drei (gedachten) Boxen gegriffen wird, bzw. die Position der Arbeitsfläche erreicht wird. Die Hand bewegt sich zu dem Zielpunkt hin und ruht dort für kurze Zeit, um sich danach in eine andere Richtung weiter zu bewegen. Da eine solche Bewegung theoretisch gerade in einer Ebene des (willkürlich gewählten) Koordinatensystems laufen kann, muss nicht in jeder Dimension zwingend ein Extremum vorliegen. Deshalb reicht es, wenn ein Extremum gleichzeitig in zwei von drei Koordinaten auftritt, um einen der 4 gesuchten Orte zu identifizieren.

Wenn man nun die Extremstellen in allen drei Raumrichtungen sucht, findet man weniger falsche Extrema durch Rauschen, da bei den gesuchten Stellen wie oben beschrieben, mindestens zwei Extrema gleichzeitig in zwei Raumrichtungen auftreten müssen. Dennoch tritt auch dieser Fall auf, da das Sensorrauschen sehr hochfrequent ist und damit auch Kombinationen von zwei gleichzeitigen „falschen“ Extrema in zwei Koordinaten vorkom-



(a) keine Glättung ($\sigma = 0$)

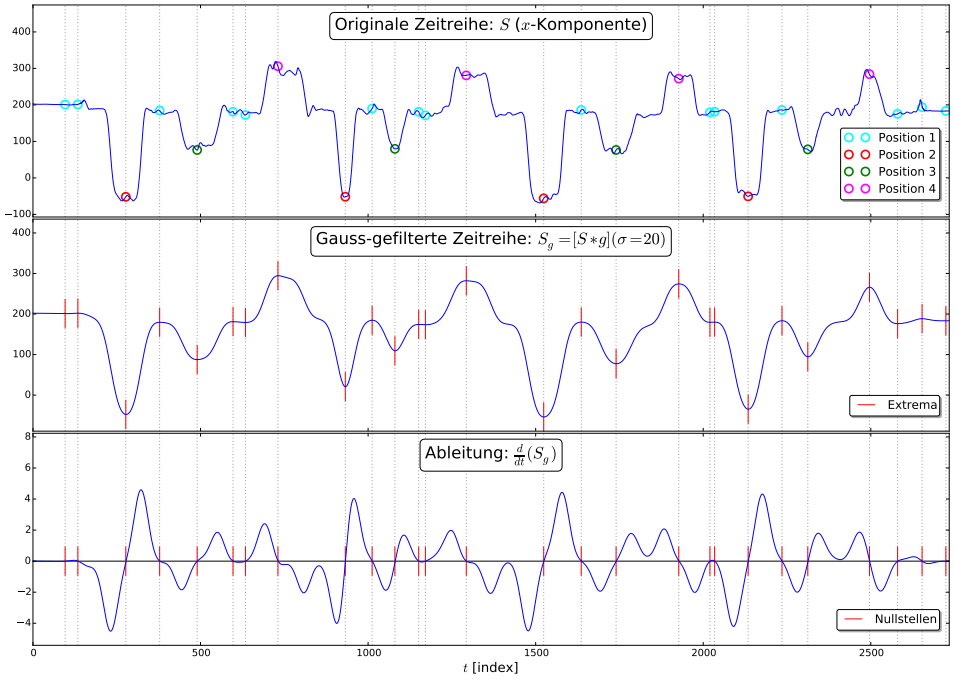


(b) Gauß-Glättung mit $\sigma = 10$

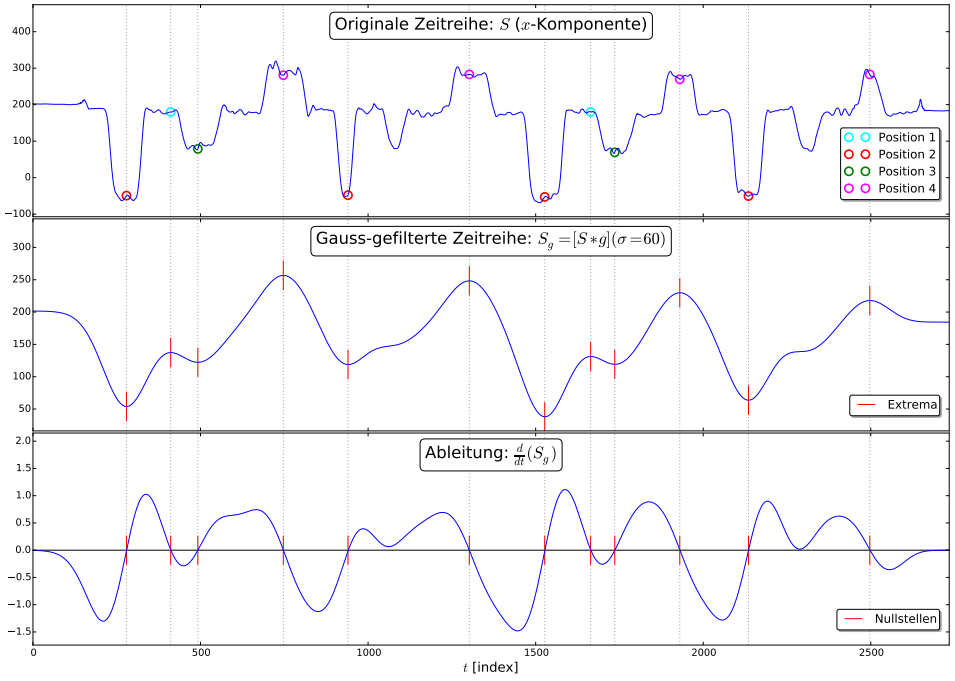
Abbildung 2.4: 1D-Clustering mit Gauß-Glättung: Datensatz der simulierten Greifbewegungen. Verwendet wurde hierbei exemplarisch die x-Komponente. Man erkennt deutlich die Abhängigkeit des Clustering-Ergebnisses vom Glättungsparameter σ.

(a) Ohne Glättung werden, bedingt durch das Sensor-Rauschen, sehr viele falsche Extrema gefunden. (Die Verbindungslinien wurden hier für eine bessere Übersichtlichkeit weggelassen.)

(b) Bei schwacher Glättung sind noch viele falsche Extrema zu sehen.



(c) Gauß-Glättung mit $\sigma = 20$



(d) Gauß-Glättung mit $\sigma = 60$

Abbildung 2.4: (Fortsetzung)

(c) Bei stärkerer Glättung werden bei den Positionen (2,3,4) korrekterweise insgesamt exakt 4 Extrema gefunden. Jedoch werden bei den Wendepunkten mit horizontaler Tangente an Position (1) immer noch falsche Extrema gefunden, während manche korrekte Extrema bereits nicht mehr erkannt werden.

(d) Bei noch stärkerer Glättung werden die meisten Extrema nicht mehr gefunden.

men. Die Problematik, dass die Zeitreihe vor der Extrema-Suche geglättet werden muss, ist damit auch im dreidimensionalen Fall gegeben, auch wenn der Glättungsparameter niedriger als im eindimensionalen Fall gewählt werden kann. Die Clustering-Ergebnisse für die simulierte Greifbewegung für den dreidimensionalen Fall sind in Abb. 2.5 dargestellt.

Grundsätzlich lässt sich mit Hilfe der Gauß-Glättung nur schwer ein optimaler Wert für den Glättungsparameter σ finden. Während bei manchen Clustern bei einem bestimmten Wert von σ noch einige unerwünschte „falsche“ Extrema gefunden werden, werden bei anderen Clustern bereits einige „korrekte“ Extrema nicht mehr erkannt. Ein optimaler Wert von σ, bei dem alle (und nur korrekte) Extrema gefunden werden lässt sich je nach Anwendungsfall eventuell gar nicht bestimmen.

Continuous-Wavelet-Transformation

Eine weitere Möglichkeit, die Extrema zu bestimmen, ist die Verwendung einer **Continuous-Wavelet-Transformation (CWT)** mit anschließender **Non-Maximum-Suppression**. Bei der CWT wird die ursprüngliche Zeitreihe $f(t)$ nicht – wie z.B. bei der einfachen Gauß-Glättung – mit nur einer Funktion $g(t)$ geglättet, sondern mit einer Schar von Funktionen $\psi(t,\sigma)$. Dabei wird eine Funktion, das sogenannte *Mother-Wavelet*, mit einem veränderlichen Parameter immer wieder mit der Ursprungszeitreihe gefaltet. Der Parameter σ wird in einem festen Intervall gesamplet. Um einen möglichst großen Bereich abzutasten ohne zu viele Faltungen berechnen zu müssen, ist es von Vorteil, das Intervall exponentiell zu samplen:

$$\sigma = 2^i \qquad i \in \{0, 1, .., \log_2(N)\}$$

wobei N der maximale Wert für σ ist. Man erhält eine Matrix, deren Zeilen i jeweils die Ursprungszeitreihe gefaltet mit $\psi(t,\sigma_i)$ darstellen. Für ein Beispiel, siehe Abb. 2.6 und 2.7. Formal kann man die CWT wie folgt definieren:

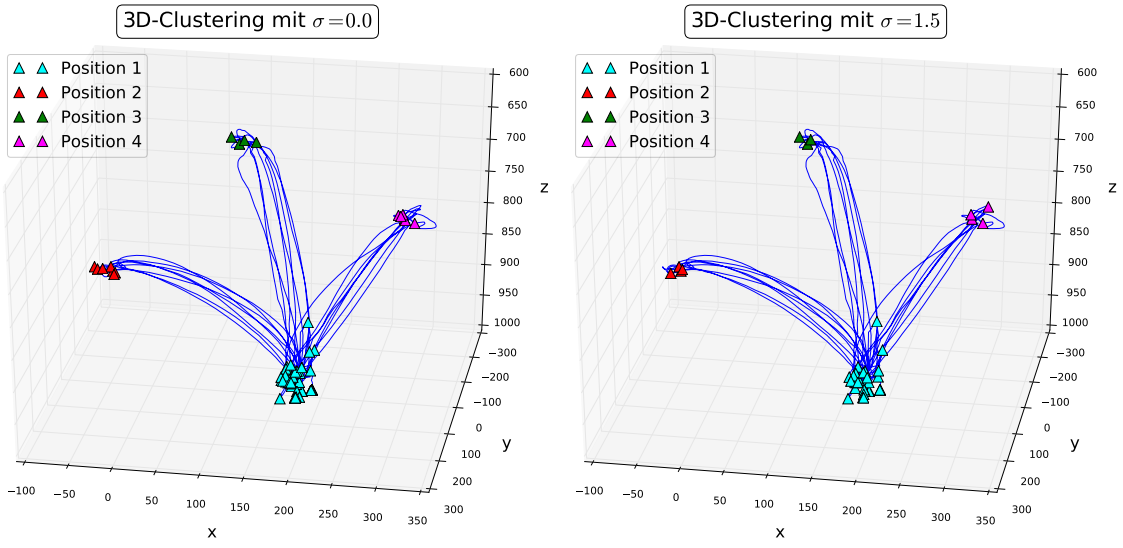
Definition 8: Continuous-Wavelet-Transformation

Sei $f(t) : \mathbb{R} \mapsto \mathbb{R}$ eine stetige Funktion und sei $\psi(\sigma, t) : \mathbb{R}^+ \times \mathbb{R} \mapsto \mathbb{C}$, das sogenannte Mother-Wavelet, ebenfalls eine stetige Funktion, dann ist die CWT definiert über das folgende Integral:

$$\tilde{f}_\psi(\sigma, t) := \frac{1}{\sigma} \int_{-\infty}^{+\infty} \bar{\psi}\left(\frac{\tau - t}{\sigma}\right) \cdot f(\tau)\, d\tau$$

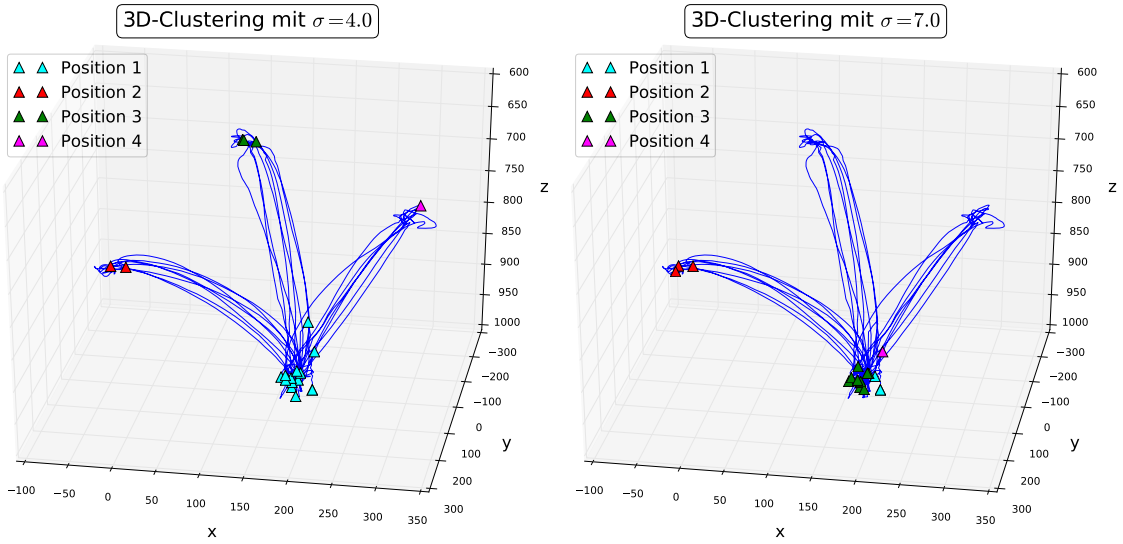
Der Querbalken ($\bar{\cdot}$) bedeutet hierbei die komplexe Konjugation. Für reellwertige Mother-Wavelets gilt diese Definition jedoch entsprechend.

Die **Non-Maximum-Suppression** sucht in dieser Matrix nach Maxima. Je nach gewünschtem Kriterium lässt sich auch nach Minima suchen. Dabei wird in einem lokalen Suchfenster jeweils das „globale“ Extremum (bezogen auf das Suchfenster) gesucht. Das Suchfenster



(a) keine Glättung ($\sigma = 0$)

(b) Gauß-Glättung mit $\sigma = 1.5$



(c) Gauß-Glättung mit $\sigma = 4.0$

(d) Gauß-Glättung mit $\sigma = 7.0$

Abbildung 2.5: 3D-Clustering mit Gauß-Glättung: Datensatz der simulierten Greifbewegungen. Im dreidimensionalen Fall sind bereits bei der ungeglätteten Zeitreihe (a) nur relativ wenige „falsche“ Extrema vorhanden, da solche falschen Extrema durch Sensorrauschen nur dann auftreten, wenn dadurch ein Extremum in mindestens zwei von drei Raumrichtungen gleichzeitig vorliegt. Deshalb führen auch bereits niedrigere Werte des Glättungsparameters zu einem besseren Ergebnis als im eindimensionalen Fall (in diesem Beispiel: etwa eine Größenordnung). (b) zeigt das für dieses Beispiel beste Clustering-Ergebnis bei schon sehr schwacher Glättung. Bei etwas stärkerer Glättung (c) fehlen bereits einige Extrema und für noch stärkere Glättung (d) werden bereits die meisten Extrema nicht mehr gefunden und das Clustering schlägt fehl.

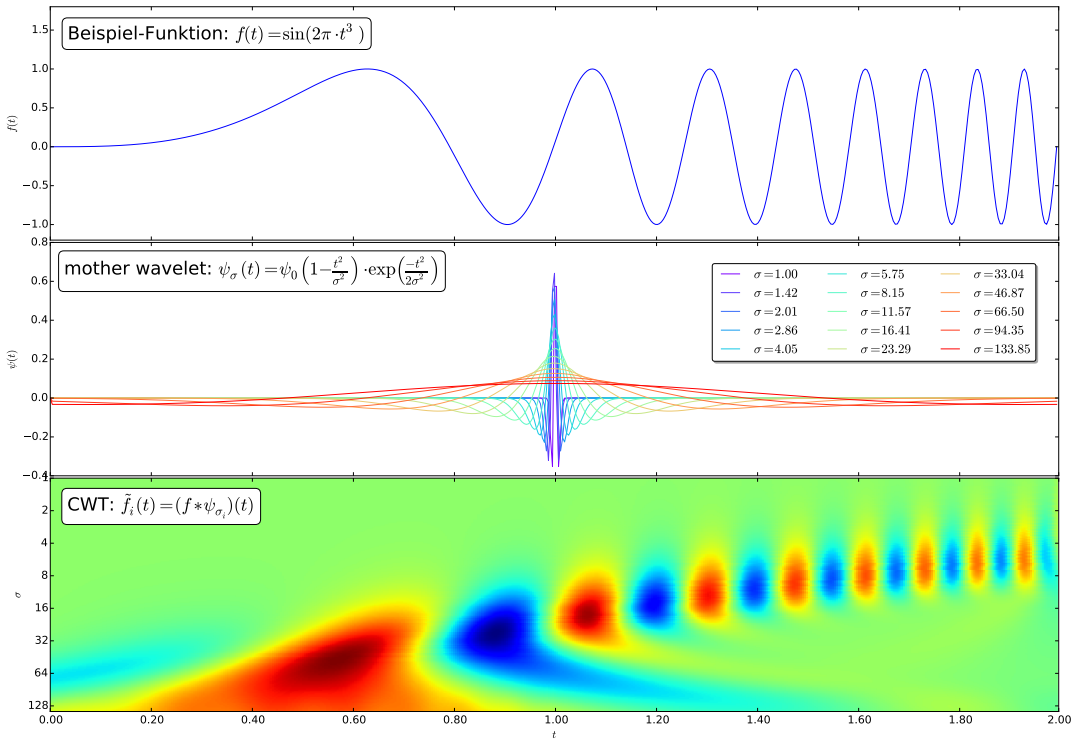


Abbildung 2.6: Beispiel einer Continuous-Wavelet-Transformation (CWT):

(oben) Eine einfache Beispiel-Funktion, die mittels CWT gefaltet werden soll.

(Mitte) *„Mother-Wavelet“*-Funktion für verschiedene Glättungsparameter σ. Als Mother-Wavelet wurde hier das *Ricker-Wavelet* (wegen seines Aussehens auch bekannt unter dem Namen *„Mexican Hat“-Wavelet*) verwendet.

(unten) CWT der Beispiel-Funktion mit dem „Mother Wavelet“. In x-Richtung ist die Zeit t aufgetragen, in y-Richtung der Glättungsparameter σ. Die Farbe eines Bildpunktes steht hierbei für den Wert der gefalteten Funktion $\tilde{f}_i(t) = (f * \psi_{\sigma_i})(t)$. Es wurde das Farbschema „jet“ verwendet: blau bedeutet kleine (negative) und rot bedeutet große (positive) Werte.

wird dann sukzessive zeilen- und spaltenweise über die komplette Matrix geschoben. Auf diese Weise lassen sich Extremstellen in der Matrix bestimmen. Um nun die vielen unerwünschten Extrema, die durch Rauschen in der Zeitreihe vorhanden sind, herauszufiltern betrachtet man nur die Extrema ab einem gewissen Schwellwert $\sigma \geq \sigma_{\text{threshold}}$. Dadurch wird eine hinreichende Glättung der Zeitreihe erreicht.

Die Ergebnisse der Segmentierung und des Clusterings mittels CWT mit anschließender Non-Maximum-Suppression auf dem Datensatz der simulierten Greifbewegungen sind in Abb. 2.8 für den eindimensionalen bzw. in Abb. 2.9 für den dreidimensionalen Fall dargestellt. Als *„Mother-Wavelet“* wurde hier das *Ricker-Wavelet* [17] (wegen seines Aussehens auch bekannt unter dem Namen *„Mexican Hat“-Wavelet*) verwendet. Dieses lässt sich aus der zweiten Zeitableitung der Gaußschen Normalverteilungs-Funktion konstruieren und ist besonders gut geeignet, lokale Extrema hervorzuheben, während hochfrequentes Rauschen unterdrückt wird. Auch bei diesem Verfahren gibt es wieder einen Parameter: den Schwellwert $\sigma_{\text{threshold}}$. Dieser muss entsprechend günstig gewählt sein, damit die Segmentierung korrekt arbeitet. Eine komplett automatisierte Segmentierung, die ohne jegliche menschliche Interaktion auskommt, ist also auch mit diesem Verfahren nicht möglich. Allerdings sind beide vorgestellten Verfahren (Gauß-Glättung und CWT) geeignet, um eine automatisierte Vor-Segmentierung zu bestimmen, die einen anschließenden manuellen Segmentierungs-Vorgang bereits deutlich erleichtert.

Die beiden Verfahren wurden hier lediglich als Konzepte vorgestellt, wie eine Segmentierung zumindest teilweise automatisiert werden kann. Die im folgenden Kapitel verwendeten Daten wurden alle entweder manuell segmentiert (Opportunity-Datensatz) oder direkt als einzelne Bewegungsmuster bzw. Gesten einzeln aufgenommen (alle anderen Datensätze), sodass eine Segmentierung in diesen Fällen prinzipbedingt nicht erforderlich war.

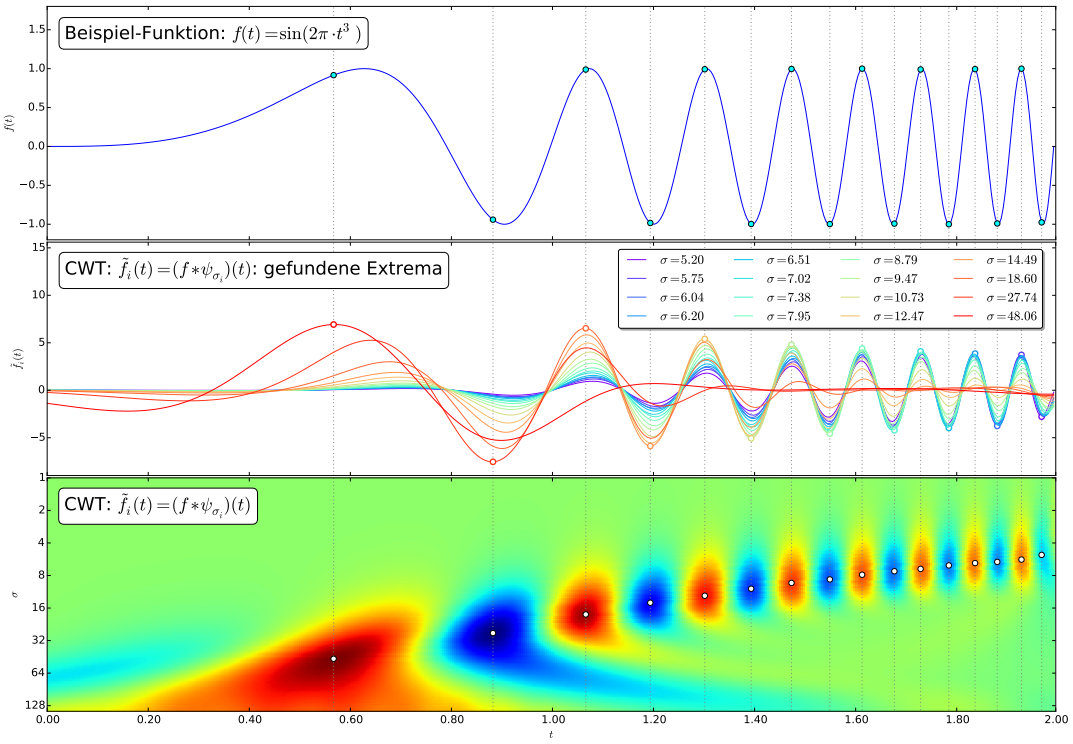


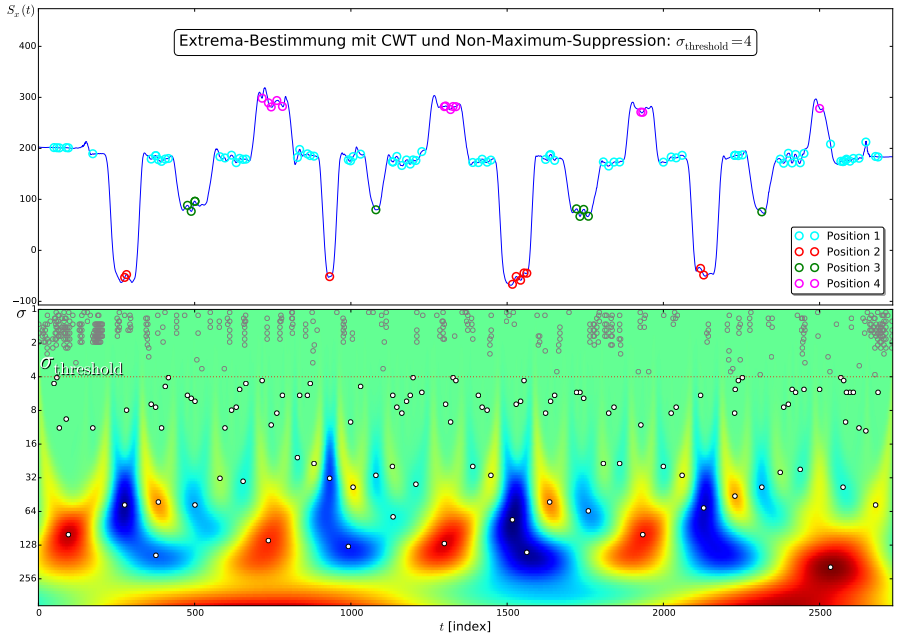
Abbildung 2.7: Beispiel einer Continuous-Wavelet-Transformation (CWT) mit anschließender Non-Maximum-Suppression zur Bestimmung der Extrema:

(oben) Eine einfache Beispiel-Funktion.

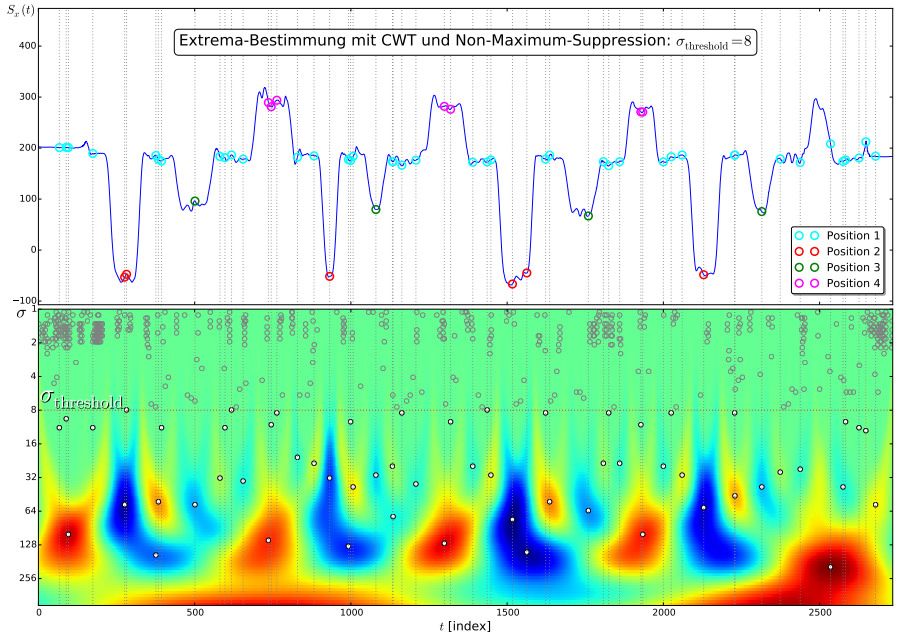
(Mitte) Die Beispiel-Funktion, gefaltet mit der „*Mother-Wavelet*“-Funktion ψ_{σ_i} (in diesem Beispiel das *Ricker-Wavelet*) für verschiedene Glättungsparameter σ_i. Dargestellt sind die Glättungsparameter, bei denen mittels *„Non-Maximum-Suppression“* die Extrema gefunden wurden.

(unten) CWT der Beispiel-Funktion.

Die mittels *„Non-Maximum-Suppression“* gefundenen Extrema sind in den Grafiken als Kreise dargestellt. Man erkennt, dass durch die Glättung die ersten Extrema auf der linken Seite leicht nach links zu kleineren Werten von t hin verschoben sind. Bei den höherfrequenten Schwingungen auf der rechten Seite wird diese Verschiebung immer kleiner.



(a) CWT mit $\sigma_{\text{threshold}} = 4$

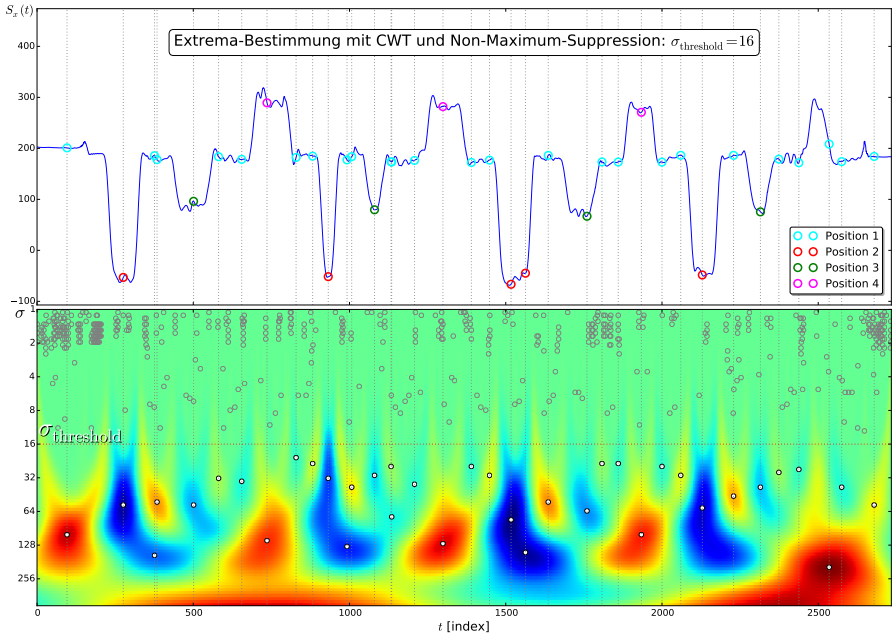


(b) CWT mit $\sigma_{\text{threshold}} = 8$

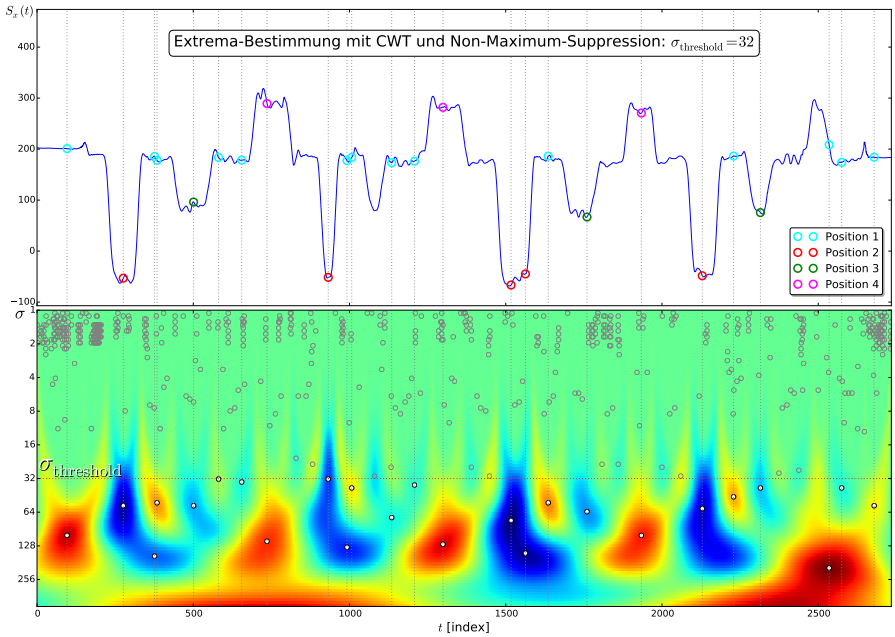
Abbildung 2.8: 1D-Clustering mit CWT und anschließender Non-Maximum-Suppression: Datensatz der simulierten Greifbewegungen. Verwendet wurde hierbei exemplarisch die x-Komponente. Man erkennt auch hier deutlich die Abhängigkeit des Clustering-Ergebnisses vom Schwellwert $\sigma_{\text{threshold}}$.

(a) Berücksichtigt man die meisten gefundenen Extrema (auch solche bei sehr geringer Glättung) so werden – bedingt durch das Sensor-Rauschen – sehr viele falsche Extrema gefunden. (Die Verbindungslinien wurden hier für eine bessere Übersichtlichkeit weggelassen.)

(b) Bei etwas höherem Schwellwert sind noch viele falsche Extrema zu sehen.



(c) CWT mit $\sigma_{\text{threshold}} = 16$

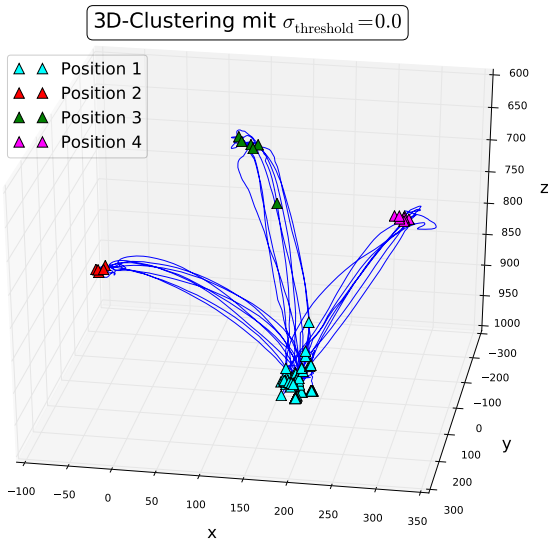


(d) CWT mit $\sigma_{\text{threshold}} = 32$

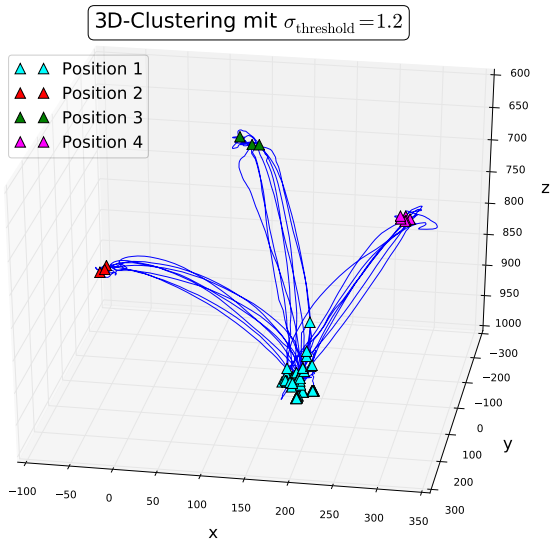
Abbildung 2.8: (Fortsetzung)

(c) Bei noch höherem Schwellwert werden bei Position (3) korrekterweise insgesamt exakt 4 Extrema gefunden. Bei Position (4) wird jedoch eine Position bereits nicht mehr erkannt, während bei Position (2) noch fälschlicherweise ein doppeltes Extremum erkannt wird. Bei den Wendepunkten mit horizontaler Tangente an Position (1) werden immer noch einige falsche Extrema gefunden.

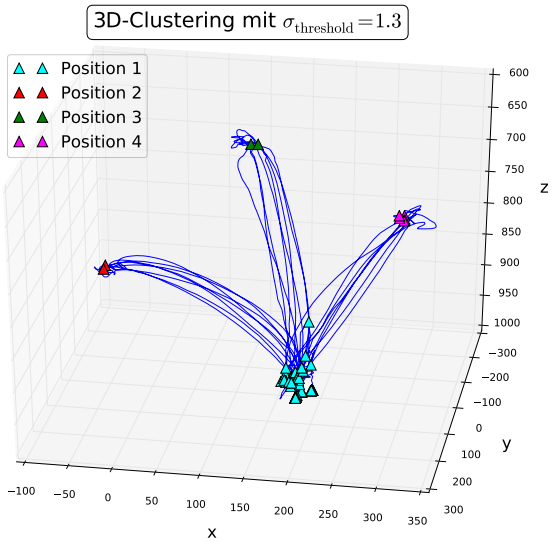
(d) Wählt man den Schwellwert nun noch höher, werden die meisten Extrema nicht mehr gefunden.



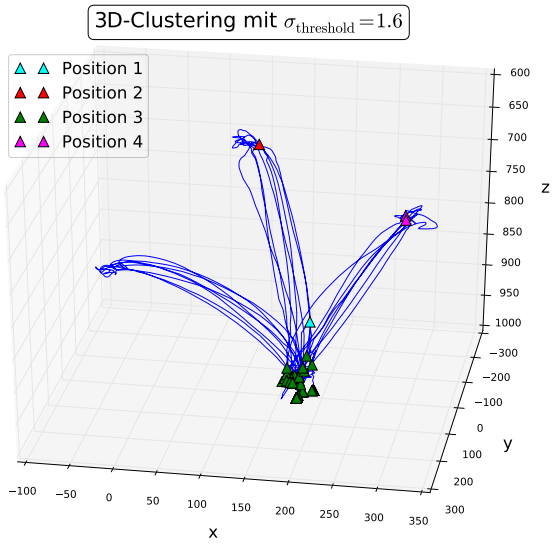
(a) Schwellwert $\sigma_{\text{threshold}} = 0.0$



(b) Schwellwert $\sigma_{\text{threshold}} = 1.2$



(c) Schwellwert $\sigma_{\text{threshold}} = 1.3$



(d) Schwellwert $\sigma_{\text{threshold}} = 1.6$

Abbildung 2.9: 3D-Clustering mit CWT und anschließender Non-Maximum-Suppression: Datensatz der simulierten Greifbewegungen. Im dreidimensionalen Fall sind bereits bei Betrachtung aller gefundenen Extrema (a) nur relativ wenige „falsche“ Extrema vorhanden, da solche falschen Extrema durch Sensorrauschen nur dann auftreten, wenn dadurch ein Extremum in mindestens zwei von drei Raumrichtungen gleichzeitig vorliegt. Deshalb führen auch bereits niedrigere Schwellwerte zu einem besseren Ergebnis als im eindimensionalen Fall (in diesem Beispiel: etwa eine Größenordnung). (b) zeigt das für dieses Beispiel beste Clustering-Ergebnis bei schon sehr niedrigem Schwellwert. Bei etwas höherem Schwellwert (c) fehlen bereits einige Extrema und für noch höheren Schwellwert (d) werden bereits die meisten Extrema nicht mehr gefunden und das Clustering schlägt fehl. Es verhält sich also ganz analog zum Ergebnis unter Verwendung der Gauß-Glättung.

2.3 Vergleich von Zeitreihen: Abstandsmaße

Wie wir bereits gesehen haben, lassen sich Zeitreihen in den unterschiedlichsten Anwendungsgebieten erzeugen. Hat man viele Zeitreihen z.B. in einer Datenbank gesammelt, sind möglicherweise die folgenden Fragestellungen von Interesse:

- "Gegeben eine neue Messung, ist diese bereits in der Datenbank vorhanden?" (Wiedererkennung)
- "Welche Zeitreihe passt am wenigsten zu den anderen?"(Anomaliedetektion)
- Eine weitere Aufgabe wäre z.B. eine Unterteilung in Ausreißer und gültige Messreihen zu finden (Ausreißerdetektion)
- Möglicherweise ist es von Interesse, die Datenbank in Gruppen von Zeitreihen aufzuteilen, die ähnliche Eigenschaften besitzen (Klassifikation / Clustering)

All diese Aufgaben setzen voraus, dass man einen Abstand zwischen zwei gegebenen Zeitreihen definiert. Hierzu ist es wiederum erforderlich, dass für jedes Wertepaar von Punkten der Zeitreihen $(a,b) \in M \times M$ ein Abstandsmaß existiert. Diese sogenannte Metrik muss die folgenden Eigenschaften erfüllen:

Definition 9: Metrik

Sei M eine (nicht leere) Menge und $d : M \times M \to \mathbb{R}$ eine reellwertige Funktion, so ist d eine **Metrik** genau dann, wenn sie die folgenden Eigenschaften erfüllt:

$$d(a,b) = 0 \quad \Leftrightarrow a = b \qquad \forall a,b \in M \qquad \text{(Koinzidenz)} \tag{2.1}$$

$$d(a,b) = d(b,a) \qquad \forall a,b \in M \qquad \text{(Symmetrie)} \tag{2.2}$$

$$d(a,b) + d(b,c) \geq d(a,c) \qquad \forall a,b,c \in M \qquad \text{(Dreiecksungleichung)} \tag{2.3}$$

Des Weiteren gilt:

$$d(a,b) \geq 0 \quad \forall a,b \in M \tag{2.4}$$

Beweis 1

Wählt man $a = c$ gilt nach Dreiecksungleichung $d(a,b) + d(b,a) \geq d(a,a)$. Aus der Symmetrieeigenschaft folgt: $2 \cdot d(a,b) \geq d(a,a)$ und mit der Koinzidenzeigenschaft schließlich $d(a,b) \geq 0$. □

Im Falle skalarer (eindimensionaler) Messwerte $a,b \in M \subseteq \mathbb{R}$ (z.B. Wegstrecke in m, Zeitdauer in s, Masse in kg, etc.) kann man dazu zum Beispiel einfach den Betrag der Differenz der Messwerte verwenden.

$$d_{\text{skalar}}(a,b) := |a-b|$$

Sind die Messpunkte N-dimensionale Vektoren $a, b \in M \subseteq \mathbb{R}^N$ ist z.B. die L_p-Norm

$$d_{L_p}(a,b) := \|a-b\|_p = \left(\sum_{i=0}^{N-1} |a_i - b_i|^p \right)^{\frac{1}{p}} \qquad p > 0$$

eine sinnvolles Abstandmaß zwischen zwei Punkten. Der Parameter p kann dabei z.B. auf $p = 1$ (Manhatten Metrik)

$$d_{L_1}(a,b) := \|a-b\|_1 = \sum_{i=0}^{N-1} |a_i - b_i|$$

oder $p = 2$ (Euklidische Metrik)

$$d_{L_2}(a,b) := \|a-b\|_2 = \sqrt{\sum_{i=0}^{N-1} (a_i - b_i)^2}$$

festgelegt werden. Da das Wurzelziehen eine rechentechnisch „teure“ Operation ist, wird bei der Euklidischen Metrik die Wurzel meist weggelassen. Dies ist zulässig, da die Wurzelfunktion monoton ist und deshalb durch das Weglassen der Wurzel keine Reihenfolge bei der Bewertung der Abstände geändert wird.

Handelt es sich bei den Messpunkten um Rotationen, die als normierte Quaternionen dargestellt werden ($a, b \in \mathbb{H}_1$), ist es sinnvoll den Winkel zwischen beiden Drehungen als Maß zu verwenden (siehe Gleichung (1.4)):

$$d_{\mathbb{H}_1}(a,b) := 2 \cdot \arccos(|\langle a|b\rangle|)$$

Der Operator $\langle\cdot|\cdot\rangle$ steht hier für das Skalar-Produkt. Der Faktor 2 kann hierbei ebenfalls auf Grund der Monotonie weggelassen werden, um Rechenzeit einzusparen. Durch den Betrag ($|\cdot|$) wird jeweils der kleinere Winkel verwendet.

2.3.1 Lockstep-Abstandsmaß

Ist eine Metrik auf dem Raum der Messwerte (Punkte) der Zeitreihe definiert, lässt sich ein Abstandsmaß zwischen zwei gegebenen Zeitreihen definieren. Eine einfache Möglichkeit ist es, bei zwei gleich langen Zeitreihen einfach die paarweisen Abstände zweier Messwerte aufzuaddieren:

Definition 10: Lockstep-Distanz

Gegeben zwei (gleich lange) Zeitreihen $f_i \in M$ und $g_i \in M$ der Länge N mit gleicher Wertemenge M. Sei $d : M \times M \mapsto \mathbb{R}$ eine Metrik auf M so ist die **Lockstep-Distanz** zwischen diesen Zeitreihen gegeben als: $\text{dist}_{\text{lockstep}}(f,g) := \sum_{i=0}^{N-1} d(f_i, g_i)$.

Damit Zeitreihen unterschiedlicher Amplituden sinnvoll aufeinander ausgerichtet („aligniert“) werden können, ist es üblich, eine sogenannte **z-Normierung** durchzuführen. Hierbei wird die Zeitreihe folgendermaßen abgebildet:

Definition 11: z-Normierung

Sei $C \in M^n$ eine Zeitreihe der Länge n, dann nennt man die Abbildung

$$\begin{aligned} &z : M^n \mapsto M^n, C \mapsto z(C) \\ &C = \{c_0, c_1, \ldots, c_{n-1}\} \mapsto C' = \{c'_0, c'_1, \ldots, c'_{n-1}\} \\ &c'_i := \frac{c_i - \mu}{\sigma} \quad \forall i \end{aligned}$$

z-Normierung. Hierbei ist μ der Mittelwert und σ die Standardabweichung der originalen Zeitreihe.

Während μ und σ für eindimensionale, reellwertige Zeitreihen klar definiert sind, muss man für z.B. Quaternionen-wertige Zeitreihen sinnvolle Definitionen für diese statistischen Größen finden. Allerdings spielt die z-Normierung bei Rotations-Quaternionen $q \in \mathbb{H}_1$ keine große Rolle, da diese per definitionem normiert sind und deshalb die Werte aller Komponenten auf das Intervall $q_\mu \in [-1,1] \, \forall \mu \in \{0,1,2,3\}$ beschränkt sind. Lediglich beim translatorischen Anteil einer 6DoF-Pose ist eine z-Normierung sinnvoll. Da wir uns im Folgenden jedoch ausschließlich auf den rotatorischen Anteil konzentrieren werden, kommt die z-Normierung dort nicht zum Einsatz.

2.3.2 Dynamic Time Warping

Beim Lockstep-Abstandsmaß werden paarweise immer zwei Punkte mit gleichem Index aufeinander abgebildet. Dieses Vorgehen ist nur dann sinnvoll, wenn die Zeitreihen auf der Zeitachse synchronisiert sind. Gibt es lokale oder globale Stauchungen oder Streckungen auf der Zeitachse, liefert Lockstep keine optimalen Ergebnisse, da diese Deformationen nicht berücksichtigt werden können. Ein Verfahren, welches solche Deformationen der Zeitachse berücksichtigt, ist Dynamic Time Warping (DTW) [18, 19]. Im Gegensatz zum Lockstep-Verfahren müssen die Zeitreihen bei DTW nicht zwingend gleich lang sein. Ein bekanntes Beispiel ist der Vergleich zweier Wörter bei der automatisierten Spracherkennung [18]. Ein und dasselbe Wort wird von verschiedenen Personen unterschiedlich ausgesprochen. Selbst wenn die gesamte Länge in etwa gleich ist, können einzelne Silben unterschiedlich gedehnt oder gestaucht werden. Abb. 2.10 zeigt anschaulich den Unterschied zwischen Lockstep und DTW.

Für eine formale Beschreibung von Dynamic Time Warping benötigt man die Definition eines Warping-Pfades (vgl. [20]).

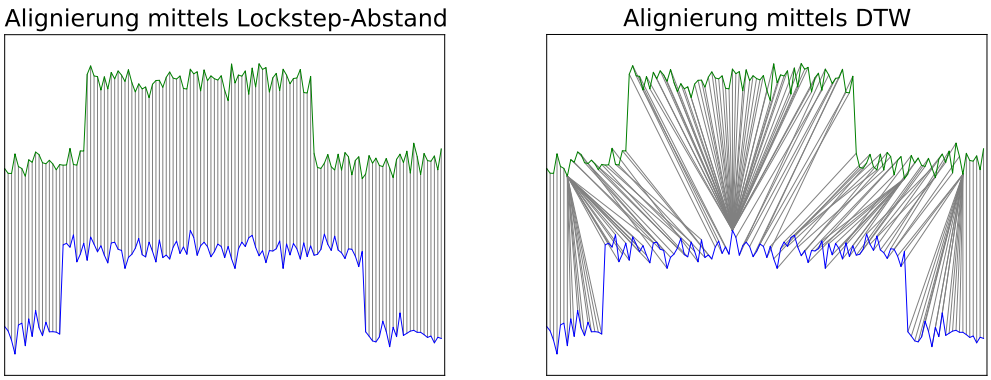


Abbildung 2.10: Veranschaulichung der unterschiedlichen Alignierung von Lockstep und DTW am Beispiel zweier Cylinder-Shapes aus dem CBF-Datensatz. **[links]**: Alignierung mittels Lockstep-Abstandsmaß. **[rechts]**: Alignierung mittels elastischem Abstandsmaß (Dynamic Time Warping). Die x-Achse stellt hierbei jeweils die Zeit-Achse dar, während die y-Achse die Werte-Achse repräsentiert.

Definition 12: Warping-Pfad

Seien C und C' zwei Zeitreihen mit Indexmengen $\mathcal{I}$ und $\mathcal{J}$. Die Folge von Tupeln $\gamma := ((i_l, j_l) \in \mathcal{I} \times \mathcal{J})_l$ nennt man einen **monotonen, kontinuierlichen** Warping-Pfad **mit der Randbedingung einer globalen Alignierung** dann und nur dann wenn

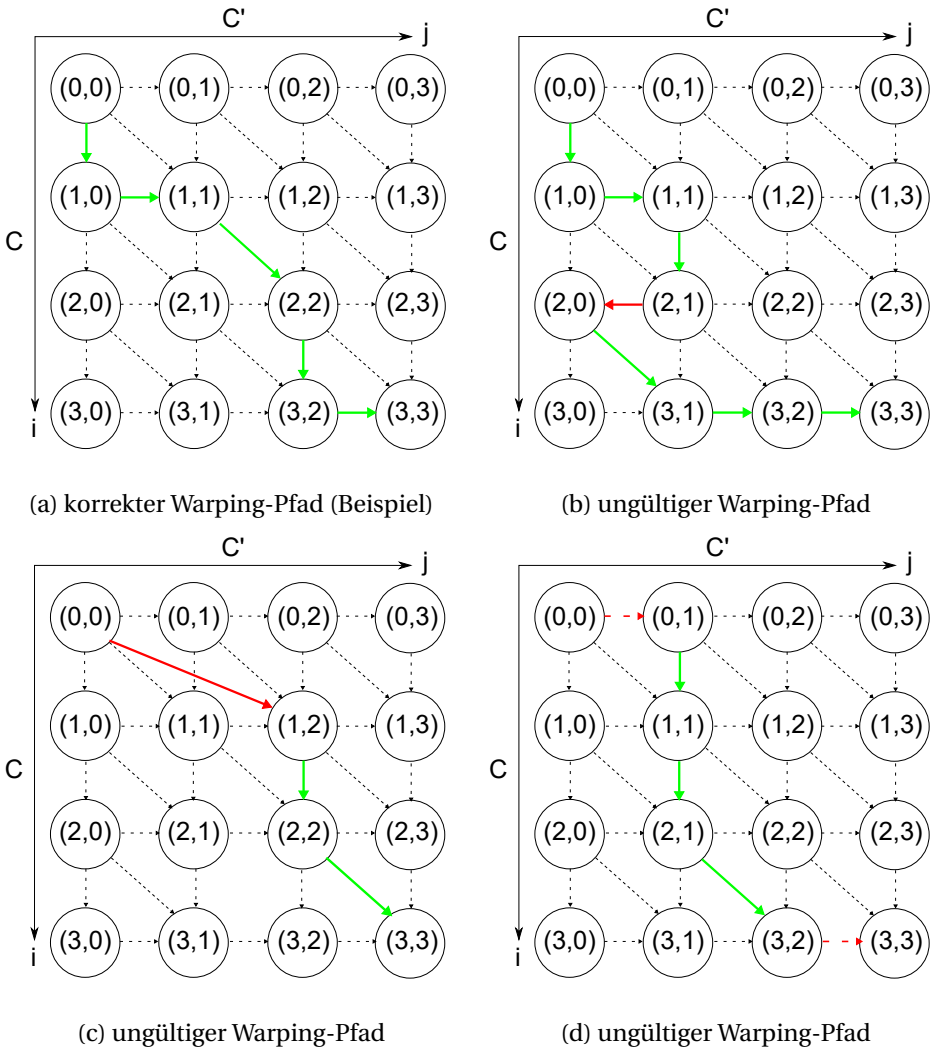
$$\min(i_{l+1} - i_l, j_{l+1} - j_l) \geq 0 \quad \wedge \quad \max(i_{l+1} - i_l, j_{l+1} - j_l) = 1 \quad \forall l \in \{0, ..., |\gamma| - 2\}$$

mit der zusätzlichen Bedingung: $(i_0, j_0) = (0,0)$ und $(i_{|\gamma|-1}, j_{|\gamma|-1}) = (|C|-1, |C'|-1)$.

Aus dieser Definition ergeben sich drei Haupteigenschaften für Dynamic Time Warping:

- **Monotonie:** Da $\max(i_{l+1} - i_l, j_{l+1} - j_l) = 1$ wird mit jeder Kante mindestens ein Index (um genau 1) inkrementiert. Aus der Bedingung $\min(i_{l+1} - i_l, j_{l+1} - j_l) \geq 0$ folgt zusätzlich, dass keine Rückschritte zulässig sind. Dadurch kann jedes Indexpaar (i, j) jeweils nur ein einziges Mal im Pfad vorkommen.
- **Kontinuität:** Aus der Bedingung $\max(i_{l+1} - i_l, j_{l+1} - j_l) = 1$ folgt, dass zwei aufeinanderfolgende Elemente des Pfades nur durch eine horizontale, vertikale oder diagonale Kante mit maximaler Länge 1 in jeder Richtung verbunden sein können. Dies impliziert, dass kein Index aus C oder C' ausgelassen werden kann.
- **Randbedingung einer globalen Alignierung:** Die letzte Bedingung $(i_0, j_0) = (0,0)$ und $(i_{|\gamma|-1}, j_{|\gamma|-1}) = (|C|-1, |C'|-1)$ erzwingt, dass die ersten und die letzten beiden Punkte aus C und C' jeweils aufeinander abgebildet werden. Es handelt sich bei Dynamic Time Warping also um eine *globale Alignierung*.

In Abb. 2.11 sind Beispiele für gültige und ungültige Warping-Pfade bildlich dargestellt.



(a) korrekter Warping-Pfad (Beispiel)

(b) ungültiger Warping-Pfad

(c) ungültiger Warping-Pfad

(d) ungültiger Warping-Pfad

Abbildung 2.11: Beispiele für DTW Warping-Pfade: (a) korrekter Pfad (Beispiel). (b) verletzt Monotonie-Bedingung. (c) verletzt Kontinuitäts-Bedingung. (d) verletzt Randbedingung. Die ungültigen Kanten sind in rot dargestellt. Im Falle (d) fehlen die rot gestrichelt dargestellten Kanten, wodurch der Pfad die ersten und letzten Elemente der Zeitreihen nicht verbindet, was die Randbedingung einer globalen Alignierung verletzt.

Um einen Warping-Pfad bewerten zu können und damit einen Vergleich zwischen verschiedenen Warping-Pfaden zu ermöglichen, benötigt man eine Gewichtungsfunktion $d : M \times M \mapsto \mathbb{R}_0^+, (C_{i_l}, C'_{j_l}) \mapsto d(C_{i_l}, C'_{j_l})$, welche den lokalen Abstand zwischen zwei Punkten $C_{i_l} \in M$ und $C'_{j_l} \in M$ der Zeitreihen C und C' beschreibt. Damit lässt sich der **optimale Warping-Pfad** wie folgt definieren (vgl. [20]):

Definition 13: optimaler Warping-Pfad

Sei Γ die Menge aller monotonen, kontinuierlichen Warping-Pfade mit der Randbedingung einer globalen Alignierung. Der **optimale Warping-Pfad** $\hat{\gamma}$ und das damit verbundene **Abstandsmaß** $\hat{d}$ bezüglich einer gegebenen Gewichtungsfunktion $d : M \times M \mapsto \mathbb{R}_0^+$ sind definiert als:

$$\hat{\gamma} := \underset{\gamma \in \Gamma}{\arg\min} \sum_{(i_l, j_l) \in \gamma} d(C_{i_l}, C'_{j_l}) \quad \text{und} \quad \hat{d} := \min_{\gamma \in \Gamma} \sum_{(i_l, j_l) \in \gamma} d(C_{i_l}, C'_{j_l})$$

In der Praxis lässt sich Dynamic Time Warping mit Hilfe der dynamischen Programmierung berechnen. Dabei wird eine Bewertungs-Matrix (auch **Penalty-Matrix**) P der Größe $(|C|+1) \times (|C'|+1)$ nach folgendem Relaxierungsschema zeilenweise berechnet[2]:

$$P[i,j] = d(C[i-1], C'[j-1]) + \min \begin{cases} P[i-1,j] & \text{(oben)} \\ P[i-1,j-1] & \text{(diagonal)} \\ P[i,j-1] & \text{(links)} \end{cases}$$

mit den Startbedingungen

$$P[0,0] = 0, \quad P[0,j] = \infty \quad \forall\, j \geq 1, \quad P[i,0] = \infty \quad \forall\, i \geq 1.$$

Da jedes Element dieser Matrix ausgerechnet werden muss, ist die dafür benötigte **Laufzeit** in $\mathcal{O}(|C| \cdot |C'|)$. Um den optimalen Warping-Pfad selbst zu erhalten, muss die Vorgänger-Information in einer separaten Matrix mitgespeichert werden. Damit lässt sich mit Hilfe von Backtracing der Warping-Pfad rekonstruieren. Bei der Bewertungs-Matrix P werden jeweils nur die letzten beiden Zeilen für die weitere Relaxierung benötigt. Damit lässt sich der **Speicherbedarf** von $\mathcal{O}(|C| \cdot |C'|)$ auf $\mathcal{O}(2 \cdot |C'|) = \mathcal{O}(|C'|)$ reduzieren. Dies gilt nicht für die Matrix mit den Vorgängerinformationen, da diese für das Backtracing des Warping-Pfades komplett benötigt wird. Deshalb ist der Speicherbedarf für den Fall, dass der Warping-Pfad benötigt wird für die naive Rekonstruktion[3] in $\mathcal{O}(|C| \cdot |C'|)$.

[2] Man beachte hierbei den Index-Shift $i \mapsto i+1$ bzw. $j \mapsto j+1$ für die Adressierung der Matrix-Zellen, da die Penalty-Matrix um eine Zeile und eine Spalte verbreitert wird. Dadurch lässt sich die Startbedingung für die Matrix-Relaxierung einfach implementieren, indem die erste Zeile P[0,j] und die erste Spalte P[i,0] der Penalty-Matrix mit ∞ initialisiert wird und der „Startknoten" P[0,0] mit 0. Die Relaxierung erfolgt dann Zeilenweise, beginnend mit der Zelle P[1,1].

[3] Hirschberg hat in [21] gezeigt, dass das Backtracing bei einer mittels dynamischer Programmierung relaxierten Matrix auch mit nur linearem Speicherbedarf $\mathcal{O}(|C|)$ möglich ist.

Nehmen wir an, die beiden zu vergleichenden Zeitreihen haben ähnliche Längen. Also gilt $|C| \approx |C'|$. Damit ist die Laufzeit von DTW in $\mathcal{O}(|C|^2)$, also quadratisch in der Länge der Eingabe-Zeitreihen. Diese Laufzeit lässt sich reduzieren, indem man nicht die komplette Bewertungs-Matrix relaxiert, sondern nur einen Bereich um die Haupt-Diagonale [22]. Diese durch zusätzliche Randbedingungen eingeschränkte Variante des Dynamic Time Warpings ist auch als **constrained DTW (cDTW)** bekannt. Hierfür gibt es verschiedene Ansätze. Bekannte Varianten sind das **Sakoe-Chiba-Band** und das seltener verwendete **Itakura-Parallelogramm** [23]. In beiden Fällen werden extrem deformierte Pfade ausgeschlossen, die weit von der Haupt-Diagonalen entfernt verlaufen, was die Klassifikationsqualität von cDTW gegenüber der normalen (unconstrained) Variante von DTW verbessert. Das Sakoe-Chiba-Band schreibt folgendes Relaxierungsschema vor:

- Wähle ein Fenster $W \in \mathbb{N}_0$.
- In jeder Zeile i mit $0 \leq i \leq |C| - 1$:
- relaxiere nur die Spalten j mit $\max(i - W, 0) \leq j \leq \min(i + W, |C'| - 1) \quad \forall j$.

Dieses Relaxierungsschema reduziert die asymptotische Laufzeit auf $\mathcal{O}(W \cdot |C'|)$. Da jedoch die Fenstergröße $W \propto |C|$ (beispielsweise $W = 0.1 \cdot |C|$) gewählt werden sollte, um die möglichen Warping-Pfade nicht zu sehr einzuschränken [22], ist die asymptotische Laufzeit dieselbe wie bei der vollen Relaxierung, nämlich $\mathcal{O}(|C|^2)$. Eine Reduktion der Laufzeit um einen konstanten Faktor kann damit allerdings schon erreicht werden. Wählt man $W = 0$, so entartet cDTW zum Lockstep-Abstandsmaß, da dann $j = i \quad \forall j$ ist und der Warping-Pfad damit exakt entlang der Haupt-Diagonalen der Penalty-Matrix verläuft.

Der Pseudo-Code für unconstrained DTW und Sakoe-Chiba constrained DTW ist in Abb. 2.12 und Abb. 2.13 dargestellt. Eine beispielhafte Darstellung der Penalty-Matrix als gerichteter Graph für verschiedene DTW-Varianten ist in Abb. 2.14 gegeben.

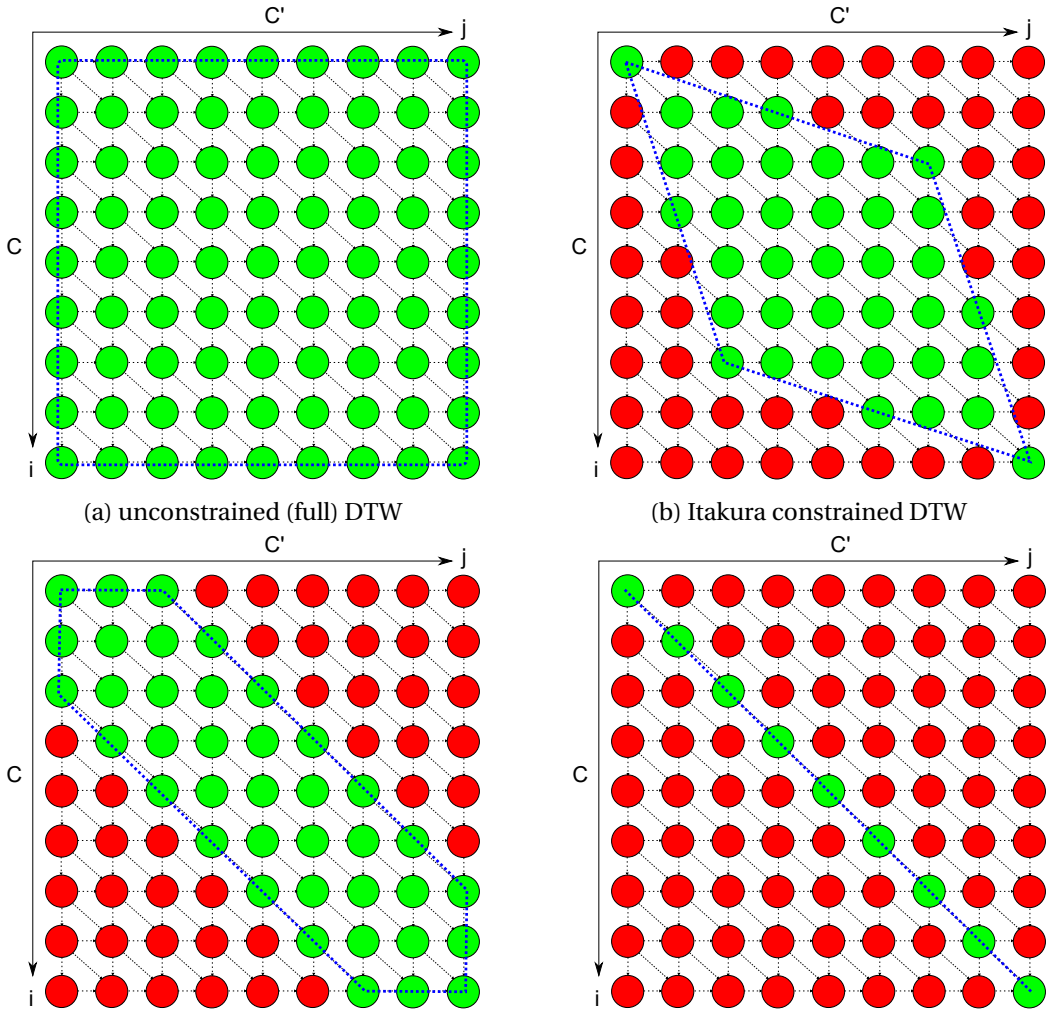
Auf die gleiche Weise lässt sich auch **subsequence DTW** implementieren. Dabei wird die erste (oberste) Zeile der Penalty-Matrix P mit 0 (statt mit ∞ wie bei DTW und cDTW) initialisiert. Dies bedeutet, dass jeder Punkt aus $|C'|$ als Startknoten gewählt werden darf. Zusätzlich wird dann nicht die letzte Zelle $P[|C|, |C'|]$ als Ergebnis zurückgeliefert, sondern das Minimum der kompletten letzten (untersten) Zeile. Dadurch verändert sich die Alignierungs-Eigenschaft von der *globalen* Alignierung bei DTW und cDTW zur *lokalen* Alignierung bei ssDTW.

```
1:  function DTW(C, C')
2:      P ← array[|C|+1, |C'|+1]                          ▷ Penalty-Matrix
3:      for i = 0 → |C| do                                ▷ initialisiere Penalty-Matrix
4:          for j = 0 → |C'| do
5:              P[i, j] ← ∞
6:          end for
7:      end for
8:      P[0,0] ← 0                                        ▷ Startknoten
9:
10:     for i = 1 → |C| do                                ▷ relaxiere alle Knoten
11:         for j = 1 → |C'| do
12:             best ← min(P[i−1, j−1], P[i−1, j], P[i, j−1])
13:             P[i, j] ← best + d(C_{i−1}, C'_{j−1})     ▷ punktweises Abstandsmaß
14:         end for
15:     end for
16:
17:     return P[|C|, |C'|]          ▷ DTW-Wert ist der rechte untere Knoten der Penalty-Matrix
18: end function
```

Abbildung 2.12: Pseudo-Code für die Berechnung des DTW-Abstandsmaßes zwischen zwei Zeitreihen C und C' mittels dynamischer Programmierung. $d(\cdot,\cdot)$ ist hierbei das punktweise Abstandsmaß zwischen zwei Punkten der Zeitreihen.

```
1:  function CDTW(C, C', W)
2:      P ← array[|C|+1, |C'|+1]                          ▷ Penalty-Matrix
3:      for i = 0 → |C| do                                ▷ initialisiere Penalty-Matrix
4:          for j = max(i − W − 1, 0) → min(i + W, |C'|) do
5:              P[i, j] ← ∞
6:          end for
7:      end for
8:      P[0,0] ← 0                                        ▷ Startknoten
9:
10:     for i = 1 → |C| do                                ▷ Relaxiere alle Knoten innerhalb des Bandes
11:         for j = max(i − W, 1) → min(i + W, |C'|) do
12:             best ← min(P[i−1, j−1], P[i−1, j], P[i, j−1])
13:             P[i, j] ← best + d(C_{i−1}, C'_{j−1})     ▷ punktweises Abstandsmaß
14:         end for
15:     end for
16:
17:     return P[|C|, |C'|]          ▷ DTW-Wert ist der rechte untere Knoten der Penalty-Matrix
18: end function
```

Abbildung 2.13: Pseudo-Code für die Berechnung des Sakoe-Chiba-constrained-DTW-Abstandsmaßes zwischen zwei Zeitreihen C und C' für eine Warping-Fenster-Breite W mittels dynamischer Programmierung. $d(\cdot,\cdot)$ ist hierbei das punktweise Abstandsmaß zwischen zwei Punkten der Zeitreihen. Der Code unterscheidet sich von dem Code der unconstrained DTW-Variante lediglich durch die rot markierten Stellen (vgl. Abb. 2.12). Durch die Beschränkung des Warping-Pfades auf ein Band der Breite $(2 \cdot W + 1)$ um die Haupt-Diagonale der Penalty-Matrix werden stark entartete Pfade ausgeschlossen und die benötigte Rechenzeit deutlich verkürzt.



(a) unconstrained (full) DTW

(b) Itakura constrained DTW

(c) Sakoe Chiba constrained DTW ($W = 2$)

(d) Sakoe Chiba constrained DTW ($W = 0$)

Abbildung 2.14: Darstellung der Penalty-Matrix als gerichteter Graph. Die grün dargestellten Knoten können auf den möglichen Pfaden erreicht werden. Die roten Knoten sind aufgrund der Randbedingungen nicht erreichbar. (a): Bei unconstrained DTW wird die komplette Penalty-Matrix relaxiert und jeder Knoten kann durch den Warping-Pfad erreicht werden. Bei den constrained Varianten – (b): Itakura-Parallelogramm, (c), (d): Sakoe-Chiba-Band – können durch die Randbedingungen nicht alle Knoten erreicht werden. (d): Für $W = 0$ kann der Warping-Pfad nur entlang der Haupt-Diagonalen verlaufen. In diesem Fall entartet DTW zum Lockstep-Abstandsmaß.

2.4 Lower Bounds für DTW

Wenn man versucht, ein gegebenes Shape in einem Datenstrom wieder zu finden, kann man dazu folgendermaßen vorgehen: Zu einem gegebenen Shape Q wird ein gleich großes Fenster der Länge $W = |Q|$ über den Datenstrom S von Anfang bis Ende durchgeschoben. Dabei wird zu jedem „Kandidaten“ $C_i = \{S_i, \ldots, S_{i+W-1}\}$ der Abstand zu Q mittels DTW bestimmt. Diese auch *sliding window subsequence DTW* genannte Vorgehensweise benötigt jedoch eine sehr lange Laufzeit: Die bei einer einzelnen Ausführung von DTW zwischen zwei Shapes Q und C benötigte Laufzeit ist proportional zum Produkt der Längen der beiden Zeitreihen: $T_{DTW} \in \mathcal{O}(|Q| \cdot |C|)$. Bei gleicher Länge $|C_i| = |Q|$ hängt die Laufzeit damit quadratisch von der Länge der Zeitreihen ab: $T_{DTW} \in \mathcal{O}(|Q|^2)$. Da das Fenster über den gesamten Datenstrom geschoben wird, beträgt die asymptotische Gesamt-Laufzeit von sliding window subsequence DTW damit $T_{ssDTW} \in \mathcal{O}(|S| \cdot |Q|^2)$.

Mit Hilfe sogenannter **Lower Bounds** lässt sich die effektive Laufzeit mitunter drastisch verringern. Die Idee ist hierbei folgende: Man erzeugt für den Abstand zweier Zeitreihen eine schnell zu berechnende *untere Schranke* (englisch: lower bound). Im Gegensatz zu DTW, welches wie oben beschrieben quadratische Laufzeit hat, kann eine solche Lower Bound in linearer Zeit $T_{LB} \in \mathcal{O}(|Q|)$ oder sogar in konstanter Zeit $T_{LB} \in \mathcal{O}(1)$ berechnet werden. Bei dem oben beschriebenen sliding window subsequence DTW sucht man den „besten“ Kandidaten, also den Kandidaten, der den geringsten Abstand (im Sinne von DTW) von dem gegebenen Shape Q hat:

$$C_{\text{best}} = \arg\min_{C_i} \text{DTW}(Q, C_i)$$

Es wird also sukzessive nach dem Testkandidaten C_i gesucht, dessen DTW-Abstand zu Q am geringsten ist. Man kann hierzu folgendermaßen vorgehen: Für den ersten Wert (wenn sich das Fenster ganz am Anfang von S befindet) muss der richtige Abstand per $\text{BSF} = \text{DTW}(Q, C_0)$ bestimmt werden. BSF steht hierbei für „best so far“ und ist der beste (also geringste) Abstandswert, der bisher gefunden wurde. Jetzt wird das Fenster entlang von S weiter geschoben und ein Wert gesucht, der kleiner ist als BSF. Dazu müsste $\text{DTW}(Q, C_1)$ ausgerechnet werden. Nun kann man aber stattdessen zunächst die Lower Bound $\text{LB}(Q, C_1)$ bestimmen. Ist dieser Wert bereits größer oder gleich dem BSF-Wert so braucht der (bezogen auf die Laufzeit „teure“) DTW-Abstand zwischen Q und C_1 gar nicht erst ausgerechnet werden, da bereits die untere Schranke $\text{LB}(Q, C_1)$ „schlechter“ (größer) ist als der BSF-Wert und per definitionem $\text{DTW}(Q, C_1) \geq \text{LB}(Q, C_1)$ sein muss. In diesem Falle kann das Fenster direkt weitergeschoben werden ohne DTW ausrechnen zu müssen. Wie oft eine Lower Bound greift, wird durch die sogenannte **Pruning-Rate** beschrieben. Dies ist das Verhältnis aus der Anzahl der Fälle, in denen die Lower Bound bereits schlechter ist als der BSF Wert (und damit die Berechnung von DTW nicht durchgeführt werden muss) zu der Gesamtzahl der Vergleiche. Die Pruning-Rate hängt natürlich vom Datensatz ab und sogar von der Reihenfolge der Vergleiche. Im besten Fall würde man direkt beim ersten Vergleich

das Minimum erwischen und für alle späteren Vergleiche würde die Lower Bound greifen, sodass insgesamt nur ein einziges Mal DTW ausgewertet werden muss. Im schlechtesten Fall könnte es aber auch vorkommen, dass die Lower Bound gar nicht greift und DTW immer ausgewertet werden muss.

Es ist deshalb ein sinnvolles Vorgehen, dass man zuerst die Lower Bounds für alle vorgesehenen Vergleiche berechnet. Man verwendet nun das Zeitreihenpaar $(Q, C_{\min})$ mit dem niedrigsten Wert der Lower Bound und berechnet deren exakten Abstand per $\mathrm{DTW}(Q, C_{\min})$ als besten bisher gefundenen Wert (BSF). Nun vergleicht man die Zeitreihen sortiert nach dem vorher bestimmten Lower Bound-Wert in aufsteigender Reihenfolge. Dabei fallen bereits alle Paare weg, deren Lower Bound-Wert größer oder gleich dem BSF Wert sind. Auf diese Weise maximiert man die Pruning-Rate der gewählten Lower Bound für den betrachteten Datensatz.

Lower Bounds können jedoch nicht nur im oben beschriebenen Fall für sliding window subsequence DTW verwendet werden. Ein weiteres klassisches Szenario ist die sogenannte (Shape-basierte) **Klassifizierung**: Gegeben einen Test-Datensatz $\mathscr{D}_{\text{test}} = \{Q_0, \ldots, Q_{k-1}\}$ der Größe $|\mathscr{D}_{\text{test}}| = k$, der also aus k Shapes (Zeitreihen) besteht, und einen Trainings-Datensatz $\mathscr{D}_{\text{train}} = \{C_0, \ldots, C_{l-1}\}$ der Größe $|\mathscr{D}_{\text{train}}| = l$, bestehend aus l Shapes. Für jedes Shape existiert ein Label, welches das Shape einer Klasse zuordnet. Solche Klassen können im Fall von Bewegungs-Zeitreihen z.B. einfache Bewegungsmuster sein wie das Öffnen oder Schließen eines Kühlschranks oder das Betätigen eines Lichtschalters. Die Aufgabe bei der Klassifizierung besteht nun darin, jedes Shape aus dem Test-Datensatz unter Zuhilfenahme des Trainings-Datensatzes seiner korrekten Klasse zuzuordnen. Das einfachste Schema hierzu ist die „One-Nearest-Neighbor“ (1NN) Klassifikation. Hierbei wird ein einzelnes Shape aus dem Test-Datensatz mit jedem Shape aus dem Trainings-Datensatz verglichen. Es wird dann der Klasse desjenigen Trainings-Shapes zugeordnet, zu welchem es den geringsten Abstand hat. Führt man dies für alle Test-Shapes durch, so lässt sich jedes Test-Shape damit einer Klasse zuordnen. Auch hier ist wiederum eine sehr hohe Rechenzeit vonnöten: Wenn jedes Test-Shape aus $\mathscr{D}_{\text{test}}$ mit jedem Trainings-Shape aus $\mathscr{D}_{\text{train}}$ per DTW verglichen werden muss, dann beträgt die asymptotische Gesamt-Laufzeit dieses Verfahrens: $T \in \mathscr{O}(k \cdot l \cdot |Q|^2)$ (hierbei wird wieder vereinfachend angenommen, dass alle Shapes die gleiche Länge $|Q|$ haben). Auch hier kann man die Laufzeit mit Hilfe von Lower Bounds wieder deutlich verbessern: Statt direkt den Abstand zweier Shapes per DTW zu bestimmen, kann man zunächst eine – im Verhältnis zu DTW – schnell auswertbare untere Schranke berechnen. Ist diese bereits größer als der beste bisher gefundene Abstandswert, braucht man DTW für dieses Zeitreihenpaar nicht mehr explizit auszurechnen.

Im Folgenden sollen zwei im Zusammenhang mit DTW häufig verwendete Lower Bounds – LB_{Kim} und $\mathrm{LB}_{\text{Keogh}}$ – näher beschrieben werden.

2.4.1 Lower Bounds im eindimensionalen Fall

Zunächst sollen hier die ursprünglich für eindimensionale Zeitreihen entwickelten Lower Bounds vorgestellt werden.

LB_{Kim}

Kim et al. definieren in [24] eine untere Schranke zur Abschätzung des DTW-Abstandes. Die im Folgenden vorgestellte Variante ist eine von Rakthanmanon et al. in [25] beschriebene modifizierte Version, die in konstanter Zeit ausgewertet werden kann.

Die Idee hierbei ist es, lediglich kleine Untermatrizen am Anfang und am Ende der Penalty-Matrix zu relaxieren, da der Warping-Pfad auf jeden Fall durch diese beiden Bereiche laufen muss (siehe auch Abb. 2.15):

Definition 14: LB_{Kim}

Seien Q und C zwei gleich lange Zeitreihen. Dann ist $LB_{Kim}(Q,C)$ definiert als:

$$LB_{Kim,k}(Q,C) := DTW^*_{(0,\ldots,k-1)}(Q,C) + DTW^*_{(|Q|-k,\ldots,|Q|-1)}(Q,C)$$

Hierbei ist mit $DTW^*_{(i,\ldots,i+k-1)}$ der per DTW bestimmte Abstandswert auf der $k \times k$ Teilmatrix $P_{[i,\ldots,i+k-1][i,\ldots,i+k-1]}$ von der Penalty-Matrix P gemeint, bei dem jedoch nicht die Zelle rechts unten, sondern das Minimum über den unteren und rechten Rand als Wert zurückgegeben wird (siehe Abb. 2.15).

Dies ist eine gültige untere Schranke für das DTW-Abstandsmaß, da die punktweise Metrik zwischen zwei Zeitreihenpunkten nicht negativ sein darf (siehe Gleichung (2.4)) und deshalb keine negativen Beiträge bei der Relaxierung entstehen können. Aus diesem Grund muss das Gesamtergebnis größer oder gleich dieser Abschätzung sein. Da diese Teilbereiche fest gewählt und damit unabhängig von der Länge der Zeitreihen Q und C sind, lässt sich $LB_{Kim,k}$ in konstanter Zeit ($T_{LB\_Kim} \in \mathcal{O}(1)$) (für festes k) ausrechnen.

$LB_{Keogh,1d}$

Eine schärfere untere Schranke, die näher als LB_{Kim} am wirklichen Wert des DTW-Abstandes liegt und damit höhere Pruning-Raten verspricht, wurde in [23] von Keogh et al. vorgeschlagen. Da DTW nur die Zeitachse strecken oder stauchen kann, nicht jedoch die Werteachse, kann auch die gewarpte Zeitreihe nur die Werte der originalen Zeitreihe annehmen. Wenn der Warping-Pfad wie bei Sakoe-Chiba constrained DTW auf eine Fensterbreite $W_{eff} = 2 \cdot W + 1$ beschränkt ist, dann kann der Wert Q'_i der gewarpten Zeitreihe Q' am Punkt i folglich nur Werte aus $\hat{Q}_{W,i} := \{Q_{i-W}, \ldots, Q_{i+W}\} \subseteq Q$ innerhalb des Warping-Fensters annehmen. Auf diese Weise kann für jeden Punkt Q_i ein Intervall $[L_i, U_i]$ gefunden werden mit $L_i \leq Q'_i \leq U_i$. L_i bzw. U_i sind dann gleich dem Minimum bzw. Maximum von Q_i in-

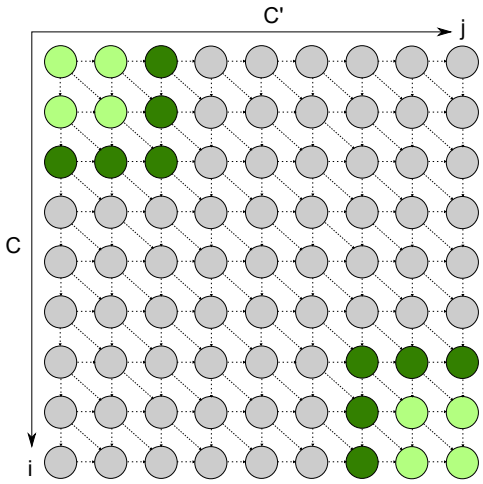


Abbildung 2.15: Penalty-Matrix am Beispiel von $LB_{Kim,3}$: Lediglich die grün dargestellten Knoten werden bei der Relaxierung betrachtet, da der optimale Warping-Pfad in jedem Fall durch diese Bereiche verläuft. Da der Warping-Pfad jeweils durch jeden der dunkelgrün dargestellten Knoten verlaufen kann, muss das Minimum der Penalty-Werte jeweils aller dunkelgrün dargestellten Knoten der Teilmatrizen betrachtet werden. Die untere Matrix lässt sich aus Symmetriegründen exakt wie die obere Matrix relaxieren (von der unteren rechten Ecke ausgehend bis zum dunkelgrünen Rand). Dies lässt sich einfach durch folgende Index-Transformation realisieren: $C_{i-1} \mapsto C_{|C|-i}$ bzw. $C'_{j-1} \mapsto C'_{|C'|-j}$. LB_{Kim} ergibt sich dann aus der Summe der Minima der Penalty-Werte der dunkelgrün dargestellten Knoten der oberen und der unteren Teilmatrix.

nerhalb des Warping-Fensters: $L_i = \min(\hat{Q}_{W,i})$ und $U_i = \max(\hat{Q}_{W,i})$. Die Menge all dieser Intervalle für alle Punkte $i \in \{0,\ldots,|Q|-1\}$ ist dann der sogenannte *Envelope* $[L,U]_Q$ um die Zeitreihe Q. Dieser lässt sich formal wie folgt definieren:

Definition 15: 1D-Envelope

Sei Q eine Zeitreihe mit Punkten $Q_i \in \mathbb{R}$. Sei ferner Q' die (zu diesem Zeitpunkt noch nicht notwendigerweise bekannte) gewarpte Zeitreihe Q, dann ist

$$[L,U]_Q := \left\{[L_0,U_0],\ldots,[L_{|Q|-1},U_{|Q|-1}]\right\}$$

mit $L_i, U_i \in \mathbb{R}$ ein (1-dimensionaler) *(Warping-)Envelope* um Q, wenn gilt:

$$L_i \leq Q'_i \leq U_i \quad \forall\, i \in \{0,\ldots,|Q|-1\}$$

Die Idee für die untere Schranke ist nun folgende: Der Beitrag zum Gesamtwert der unteren Schranke eines Punktes C_i einer Zeitreihe C, der außerhalb des Envelopes liegt, ist mindestens so groß wie der punktweise Abstand des Punktes zum Envelope. Diese Abschätzung ist

korrekt, da sich die gewarpte Zeitreihe Q' nach Definition nur innerhalb des Envelopes befinden kann. Liegt ein Punkt C_i innerhalb des Envelopes, so ist der Beitrag dieses Punktes zum Gesamtwert der unteren Schranke (mindestens) gleich Null. Ein höherer Wert kann für Punkte innerhalb des Envelopes nicht angegeben werden. Für ein Beispiel siehe Abb. 2.16.

Die formale Definition von $\mathrm{LB}_{\mathrm{Keogh,1d}}$ lässt sich nun wie folgt formulieren:

Definition 16: $\mathrm{LB}_{\mathrm{Keogh,1d}}$

Seien Q und C zwei gleich lange Zeitreihen mit Punkten $Q_i, C_i \in \mathbb{R}$. Sei ferner $[L,U]_Q$ der *Envelope* um Q, dann ist $\mathrm{LB}_{\mathrm{Keogh,1d}}$ definiert als:

$$\mathrm{LB}_{\mathrm{Keogh,1d}}([L,U]_Q, C) := \sum_{i=0}^{|Q|-1} \begin{cases} d(C_i, U_i) & \text{wenn} \quad C_i > U_i \\ d(C_i, L_i) & \text{wenn} \quad C_i < L_i \\ 0 & \text{sonst} \end{cases}$$

$d(a,b) : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_0^+$ bezeichnet hierbei das punktweise Abstandsmaß zwischen zwei Punkten der Zeitreihen.

Ein naiver Ansatz zur Erzeugung des Envelopes würde für jeden Punkt Q_i jeweils das Minimum und das Maximum aus $\hat{Q}_{W,i}$ berechnen. Die Laufzeit dieses Ansatzes wäre damit $\mathrm{T}_{\mathrm{envelope}} \in \mathcal{O}(W \cdot |Q|)$ und da die Warping-Fensterbreite bei Sakoe-Chiba constrained DTW in der Regel wie bereits erwähnt $W \propto |Q|$ gewählt wird, wäre die asymptotische Gesamtlaufzeit damit wiederum quadratisch in der Länge der Zeitreihe $\mathrm{T}_{\mathrm{envelope}} \in \mathcal{O}(|Q|^2)$ und damit genauso groß wie die Laufzeit von DTW. Somit wäre keine sinnvolle Lower Bound für DTW möglich, die einen Envelope um die Zeitreihe voraussetzt. Bei dieser Überlegung muss man jedoch Folgendes beachten: Hat man zum Beispiel einen Test-Datensatz mit k Zeitreihen und einen Trainings-Datensatz mit l Zeitreihen und möchte nun jede Zeitreihe des Test-Datensatzes mit jeder Zeitreihe des Trainings-Datensatzes vergleichen, so muss natürlich nicht für jede der $k \cdot l$ Kombinationen der Envelope ausgerechnet werden. Es reicht aus, wenn für jede Test-Zeitreihe ein Envelope berechnet wird, der dann mit allen Zeitreihen des Trainings-Datensatzes verglichen wird. Für die Trainings-Zeitreihen muss jeweils kein Envelope berechnet werden. Dennoch wäre alleine der gesamte Rechenzeitaufwand für die Envelope-Bestimmung $\mathrm{T}_{\mathrm{all\ envelopes}} \in \mathcal{O}(k \cdot |Q|^2)$ und damit immer noch proportional zum Quadrat der (mittleren) Zeitreihenlänge.

Allerdings haben Lemire et al. in [26] gezeigt, dass sich durch die Ausnutzung von Redundanzen bei der Extrema-Bestimmung in einem Datenstrom der Länge $|Q|$ die amortisierte Anzahl der Vergleiche pro Element auf nicht mehr als 3 und damit die Gesamt-Laufzeit für die Envelope-Konstruktion auf $\mathrm{T}_{\mathrm{envelope}} \propto 3 \cdot |Q|$ reduzieren lässt. Dadurch lässt sich der Envelope in linearer Zeit $\mathrm{T}_{\mathrm{envelope}} \in \mathcal{O}(|Q|)$ bestimmen. Da $\mathrm{LB}_{\mathrm{Keogh}}$ die Abstände jedes Punktes aus C zum Envelope berechnet, benötigt die Auswertung ebenfalls lineare Rechenzeit

$T_{\text{LB\_Keogh}} \in \mathcal{O}(|Q|)$.

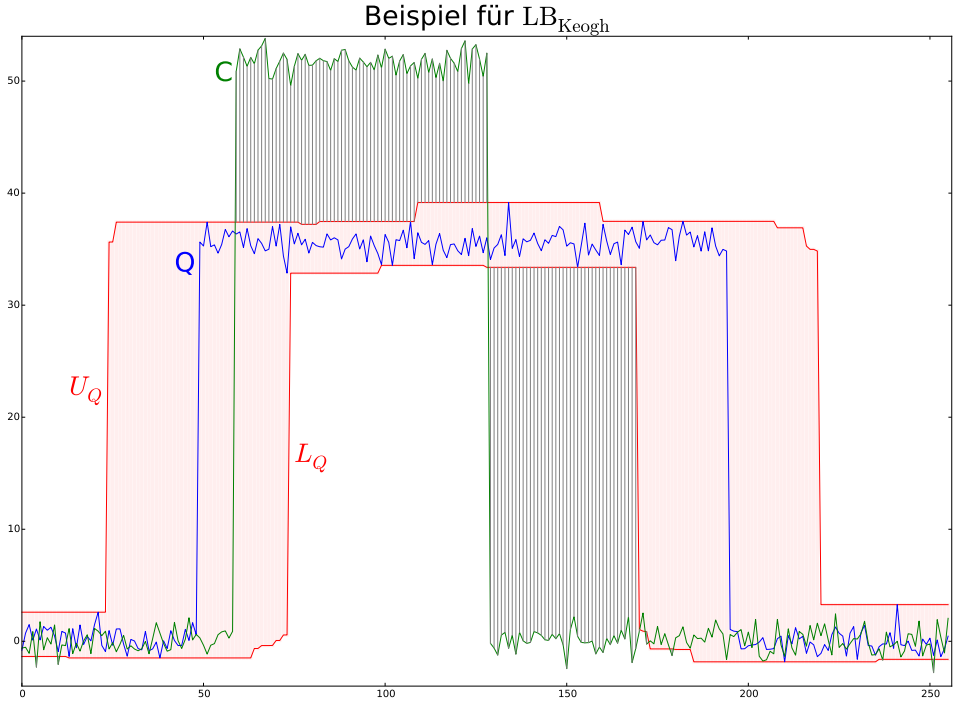


Abbildung 2.16: Beispiel für LB_{Keogh} im eindimensionalen Fall für zwei Cylinder-Shapes aus dem CBF-Datensatz: Nur die Differenzen der Punkte C_i des Shape-Kandidaten C zur jeweiligen Grenze des Envelopes $[L,U]_Q$ von Q, bei denen der Wert des Punktes C_i außerhalb des Warping-Envelopes liegt, tragen zum Gesamtwert der unteren Schranke bei. Diese sind hier durch die dunkelgrauen vertikalen Linien dargestellt. Für die Bereiche, in denen C innerhalb des Envelopes liegt, kann keine Aussage getroffen werden. Der Beitrag dieser Punkte zum Gesamtergebnis ist also (mindestens) gleich Null.

2.4.2 Erweiterung der Lower Bounds auf höher-dimensionale Zeitreihen

Die vorangehend beschriebenen Lower Bounds sind ursprünglich für eindimensionale Zeitreihen definiert. Im Folgenden wollen wir diese Definitionen nun erweitern, um den Lower-Bound-Mechanismus auch bei höherdimensionalen Zeitreihen einsetzen zu können.

$\text{LB}_{\text{Kim,Nd}}$ und qLB_{Kim}

Wie bereits in Definition 14 beschrieben wird bei LB_{Kim} ein unconstrained Dynamic Time Warping auf zwei Teilmatrizen am Anfang und am Ende der Penalty-Matrix durchgeführt. Möchte man nun höherdimensionale Zeitreihen $X = \{x_0, \ldots, x_{n-1}\}$, $x_i \in \mathbb{R}^N$ miteinander vergleichen, so verwendet man ein höherdimensionales punktweises Abstandsmaß

$d(x,y) : \mathbb{R}^N \times \mathbb{R}^N \mapsto \mathbb{R}_0^+$ für DTW und $\mathrm{LB}_{\mathrm{Kim}}$ wird dann mit dem gleichen Abstandsmaß ausgewertet. Das Gleiche gilt entsprechend für Quaternionen-wertige Zeitreihen: hier wird ein punktweises Abstandsmaß $d(x,y) : \mathbb{H} \times \mathbb{H} \mapsto \mathbb{R}_0^+$ verwendet, welches sowohl in DTW, als auch in $\mathrm{qLB}_{\mathrm{Kim}}$ ausgewertet wird. Die Definition für $\mathrm{LB}_{\mathrm{Kim,Nd}}$ bzw. $\mathrm{qLB}_{\mathrm{Kim}}$ ist damit gleich der Definition im eindimensionalen Fall (Definition 14). Es wird lediglich das punktweise Abstandsmaß $d(x,y)$ bei der Berechnung von DTW ausgetauscht.

$\mathbf{LB_{Keogh,Nd}}$

Für die Berechnung von $\mathrm{LB}_{\mathrm{Keogh}}$ wird, wie in Definition 16 beschrieben, ein Envelope benötigt. Für jeden Anfragepunkt C muss dann geprüft werden, ob dieser innerhalb oder außerhalb des Envelopes liegt. Betrachten wir nun den Fall mehrdimensionaler Vektoren im N-dimensionalen Euklidischen Vektorraum $\mathbb{R}^N$ am Beispiel der L_2^2-Norm: $d_{L_2^2}(a,b) := \|a-b\|_2^2 = \sum_{\mu=0}^{N-1}(a_\mu - b_\mu)^2$. Die μ-te Komponente dieser Norm ist einfach $(a_\mu - b_\mu)^2$ und alle Komponenten sind voneinander *entkoppelt*, d.h. *unabhängig voneinander*. Da für jede Komponente einzeln gilt: $\mathrm{LB}(a_\mu, b_\mu) \leq d(a_\mu, b_\mu)$ gilt dies folglich auch für die Summe:

$$\mathrm{LB}(a,b) = \sum_{\mu=0}^{N-1} \mathrm{LB}(a_\mu, b_\mu) \leq \sum_{\mu=0}^{N-1} d(a_\mu, b_\mu) = d(a,b)$$

Deshalb lässt sich sowohl der Envelope, als auch die untere Schranke $\mathrm{LB}_{\mathrm{Keogh,Nd}}$ *komponentenweise* berechnen. Dies gilt nicht nur für die L_2^2-Norm als punktweises Abstandsmaß. Die notwendige Voraussetzung für obige Argumentation ist lediglich folgende:

$$d(a,b) = \psi\left(\sum_{\mu=0}^{N-1} \varphi(a_\mu, b_\mu)\right)$$

mit $\varphi(a_\mu, b_\mu) : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_0^+$, also stets ≥ 0 und $\psi(x) : \mathbb{R} \mapsto \mathbb{R}$ monoton steigend, also $\psi(x_1) < \psi(x_2) \Leftrightarrow x_1 < x_2$. Im Falle z.B. der L_2^2-Norm wäre dies einfach die Identität $\psi(x) = x$ und bei der L_2-Norm wäre $\psi(x) = \sqrt{x}$. Damit lässt sich Definition 15 wie folgt erweitern:

Definition 17: Nd-Envelope

Sei Q eine Zeitreihe mit Punkten $Q_i \in \mathbb{R}^N$. Sei ferner Q' die (zu diesem Zeitpunkt noch nicht notwendigerweise bekannte) gewarpte Zeitreihe Q, dann ist

$$[L,U]_Q := \left\{[L_0, U_0], \ldots, [L_{|Q|-1}, U_{|Q|-1}]\right\}$$

mit $L_i, U_i \in \mathbb{R}^N$ ein (N-dimensionaler) *(Warping-)Envelope* um Q, wenn gilt:

$$L_{i,\mu} \leq Q'_{i,\mu} \leq U_{i,\mu} \; \forall\, i \in \{0, \ldots, |Q|-1\}, \mu \in \{0, \ldots, N-1\}$$

Damit lässt sich Definition 16 nun wie folgt auf N Dimensionen erweitern:

Definition 18: $\mathrm{LB}_{\mathrm{Keogh,Nd}}$

Seien Q und C zwei gleich lange Zeitreihen mit Punkten $Q_i, C_i \in \mathbb{R}^N$. Sei ferner $[L,U]_Q$ der *Envelope* um Q, dann ist $\mathrm{LB}_{\mathrm{Keogh,Nd}}$ definiert als:

$$\mathrm{LB}_{\mathrm{Keogh,Nd}}([L,U]_Q, C) := \sum_{i=0}^{|Q|-1} \mathrm{LB}_{\mathrm{Keogh,i}}([L,U]_{Q,i}, C_i)$$

mit

$$\mathrm{LB}_{\mathrm{Keogh,i}}([L,U]_{Q,i}, C_i) := \sum_{\mu=0}^{N-1} \begin{cases} d(C_{i,\mu}, U_{i,\mu}) & \text{wenn} \quad C_{i,\mu} > U_{i,\mu} \\ d(C_{i,\mu}, L_{i,\mu}) & \text{wenn} \quad C_{i,\mu} < L_{i,\mu} \\ 0 & \text{sonst} \end{cases}$$

$d(a_\mu, b_\mu) : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_0^+$ bezeichnet hierbei das punktweise Abstandsmaß der μ-ten Komponente zwischen zwei Punkten der Zeitreihen.

$\mathbf{qLB_{Keogh}}$

Sind die Datenpunkte der Zeitreihen Rotationen, die als Quaternionen repräsentiert werden, ist die Fragestellung, ob ein Anfragepunkt C_l innerhalb eines Envelopes liegt bzw. welchen Abstand ein Punkt zum Envelope hat, komplizierter. Auch im Falle von Quaternionen kann ein Envelope – analog zu dem Fall im Euklidischen Vektorraum $\mathbb{R}^N$ – als $[L,U]_{Q,\mu}$: $L_{i,\mu} \leq Q'_{i,\mu} \leq U_{i,\mu}\, \forall\, i, \mu$ komponentenweise definiert werden. Der Envelope $[L_i, U_i]_Q$ an einer Stelle i ist dann ein Hyper-Quader im Vektorraum $\mathbb{R}^4$. Da er komponentenweise aus den lokalen Minima und Maxima der Quaternionen-Komponenten der Zeitreihe Q zusammengesetzt ist, sind die Größen L_i und U_i im Allgemeinen nicht normiert. Es handelt sich bei dem Envelope selbst also nicht um Rotations-Quaternionen, sondern nur um eine Zeitreihe aus Bounding-Boxen im Raum $\mathbb{R}^4$, die die gewarpten Punkte Q'_i enthalten muss.

Als nächstes betrachten wir das punktweise Abstandsmaß für Quaternionen-wertige Zeitreihen. Das geodätische Abstandsmaß, also das Maß für den kürzesten Abstand zweier Punkte auf deren Mannigfaltigkeit, zwischen zwei Quaternionen ist definiert als:

$$d_{\mathbb{H}}(q_1, q_2) := \arccos |\langle q_1 | q_2 \rangle|$$

Dieses liefert den (halben) relativen Winkel zwischen den beiden durch die Quaternionen q_1 und q_2 repräsentierten Rotationen. Durch den Betrag ($|\cdot|$) wird jeweils immer der kleinere Winkel verwendet. Wir möchten nun eine untere Schranke für dieses Abstandsmaß finden. Als erstes können wir festhalten, dass für den Fall $C_l \in [L_l, U_l]_Q$, also dass der Anfrage-Punkt C_l in allen Dimensionen innerhalb des Envelopes liegt, der Beitrag dieses Punktes zum Gesamtergebnis – analog zum eindimensionalen Fall – ebenfalls verschwindet. Also nehmen

wir von jetzt ab an, dass mindestens eine Komponente $C_{l,\mu}$ von C_l außerhalb des Envelopes liegt:

$$\exists C_{l,\mu} \,|\, C_{l,\mu} \notin [L_l, U_l]_{Q,\mu} \Leftrightarrow C_l \notin [L_l, U_l]_Q$$

Um nun eine untere Schranke für den Winkelabstand zwischen dem Anfrage-Quaternion C_l und dem entsprechenden Quaternion der gewarpten Zeitreihe Q'_l angeben zu können, müssen wir das Quaternion q innerhalb des Envelopes bestimmen, welches den geringsten Winkelabstand zu C_l hat. Die grundlegende Idee unseres Ansatzes besteht nun darin, dass das Quaternion q innerhalb des Envelopes, welches am nächsten (bezogen auf den Winkelabstand) zum Anfragepunkt C_l liegt, **auf dem Rand** des Envelopes liegen muss. Mit diesem Argument kann man zuerst eine Dimension μ betrachten und die entsprechende Komponente der gesuchten Lösung q_μ in dieser Dimension auf den Rand des Hyper-Quaders legen. Die restlichen Komponenten von q werden durch die Nebenbedingung (siehe unten) bestimmt. Liegt die Lösung q damit bereits innerhalb des Envelopes, so haben wir eine gültige Lösung gefunden. Ist dies nicht der Fall, kann man rekursiv für die weiteren Dimensionen analog vorgehen bis entweder alle Dimensionen der Lösung festgelegt sind oder bereits vorher eine gültige Lösung[4] gefunden wurde. Wird keine gültige Lösung gefunden, so wird die triviale Lower Bound 0 zurückgegeben. Für alle Quaternionen $q \in \mathbb{H}$, die eine Drehung im dreidimensionalen Raum beschreiben, gibt es noch eine zweite Bedingung. Sie müssen normiert sein: $q \in \mathbb{H}_1$. Formal müssen wir also das folgende Problem lösen:

Definition 19: $\mathrm{qLB}_{\mathrm{Keogh}}$

Seien Q und C zwei gleich lange Zeitreihen. Die Punkte der Zeitreihen seien normierte Rotations-Quaternionen: $Q_i, C_i \in \mathbb{H}_1$. Sei ferner $[L,U]_Q$ der *Envelope* um Q, dann ist $\mathrm{qLB}_{\mathrm{Keogh}}$ definiert als:

$$\mathrm{qLB}_{\mathrm{Keogh}}([L,U]_Q, C) := \sum_{i=0}^{|Q|-1} \min_{q_i \in \mathbb{H}} \arccos |\langle q_i | C_i \rangle|$$
$$s.t. \quad \|q_i\| = 1, \quad q_i \in [L_i, U_i]_Q \backslash (L_i, U_i)_Q$$

Da wir bei dem relativen Rotationswinkel zwischen zwei Quaternionen immer nur den kleineren Winkel $\theta \in [0, \frac{\pi}{2}]$ betrachten wollen, verwenden wir den Betrag des Skalarproduktes. Da sowohl das Anfrage-Quaternion C, als auch das zu bestimmende Quaternion q normiert ist, ist das Argument der inversen Cosinus-Funktion $|\langle q_i | C_i \rangle|$ beschränkt auf das Intervall $[0,1]$. Auf diesem Intervall ist die inverse Cosinus-Funktion $\arccos(x)$ monoton

[4] Genau genommen müssen jeweils zwei Seiten des Hyper-Quaders (L_μ und U_μ) pro Dimension betrachtet werden und dann jeweils noch zusätzlich zwei mögliche Lösungen für den Lagrange-Multiplikator ($+\lambda$ und $-\lambda$). Schließlich wird dann von allen möglichen Lösungen das Minimum als untere Schranke gewählt. In Abb. 2.17 ist das Vorgehen detailliert in Pseudo-Code dargestellt.

fallend. Deshalb bedeutet die Minimierung von $\arccos|\langle A|B\rangle|$ die Maximierung des Arguments $|\langle A|B\rangle|$. Unter Verwendung dieser Tatsache kann unser Problem wie folgt umformuliert werden:

$$\mathrm{qLB}_{\mathrm{Keogh}}([L,U]_Q,C) := \sum_{i=0}^{|Q|-1} \arccos \max_{q_i\in\mathbb{H}} |\langle q_i|C_i\rangle|$$
$$s.t. \quad \|q_i\| = 1, \quad q_i \in [L_i,U_i]_Q \backslash (L_i,U_i)_Q$$

Auf diese Weise benötigen wir die arccos-Funktion nicht während des Optimierungsprozesses und können sie am Schluss einmal auf das gefundene Maximum anwenden. Dadurch wird der Prozess, eine analytische Lösung für das Problem zu finden, drastisch vereinfacht.

Für eine bessere Lesbarkeit werden wir in den folgenden Herleitungen den Punktindex i weglassen. Man beachte, dass die folgende Prozedur natürlich für *jeden* Punkt $C_i, i \in \{0,\ldots,|C|-1\}$ der Anfrage-Zeitreihe durchgeführt werden muss und der Envelope $[L_i,U_i]_Q$ an jedem Punkt i einen anderen Wert hat.

Das Optimierungsproblem mit der Nebenbedingung $\|q\| = 1$ kann mit Hilfe von Lagrange-Multiplikatoren ausgedrückt werden [46]. Die hierzu benötigte Lagrange-Funktion sieht folgendermaßen aus:

$$\Lambda(q_0,q_1,q_2,q_3,\lambda) = \sum_{\mu=0}^{3} q_\mu \cdot C_\mu + \tfrac{1}{2}\lambda[\sum_{\mu=0}^{3} q_\mu^2 - 1]$$

Hierbei ist λ ein sogenannter Lagrange-Multiplikator. Gemäß dem Lagrange-Formalismus, liefert die Lösung des Gleichungssystems

$$\text{(I), (II), (III), (IV)} \quad 0 = \frac{\partial\Lambda}{\partial q_\mu} = C_\mu + \lambda q_\mu \quad \forall\mu\in\{0,1,2,3\}$$
$$\text{(V)} \qquad\qquad 0 = \frac{\partial\Lambda}{\partial\lambda} = \sum_{\mu=0}^{3} q_\mu^2 - 1$$

alle gültigen Optimums-Lösungen für das gegebene Problem unter Berücksichtigung der Randbedingung, dass das gefundene Quaternion q normiert sein muss:

$$\sum_{\mu=0}^{3} q_\mu^2 = 1$$

Optimums-Lösungen bedeutet, dass sowohl Minima, als auch Maxima gefunden werden. Da das Gleichungssystem alle Optima liefert, kann der Betrag $|\cdot|$ in der Rechnung weggelassen werden, da wir alle Lösungen betrachten und am Ende das Maximum der Absolutbeträge wählen:

$$\varphi_{\min} = \arccos\left(\max_i |\mathrm{solution}_i|\right)$$

Nun setzen wir eine oder mehrere Koordinaten fest auf den Rand des Hyper-Quaders $[L,U]_Q$ und lösen dann das Gleichungssystem analytisch. Dies liefert:

$$q_\mu = \frac{-C_\mu}{\lambda} \quad \text{mit} \quad \lambda = \pm\sqrt{\frac{1-\sum\limits_{v\in\Omega} C_v^2}{1-\sum\limits_{v\in\Omega} q_v^2}}$$

mit $q_v \in \{L_v, U_v\} \quad \forall v \in \Omega$. Mit Ω ist hierbei also die Menge der Dimensions-Indizes bezeichnet, bei denen q_v fest auf den Rand des Hyper-Quaders gesetzt wurde. Betrachten wir nun die Lösung für den Lagrange-Multiplikator λ etwas genauer. Der Zähler $(1-\sum_{v\in\Omega} C_v^2)$ in der Wurzel von λ ist stets ≥ 0, da das Anfrage-Quaternion C als normiert angenommen wird und damit der Term $(\sum_{v\in\Omega} C_v^2) \leq 1$ ist. Der Nenner $(1-\sum_{v\in\Omega} q_v^2)$ in der Wurzel von λ kann jedoch unter Umständen ≤ 0 werden, und zwar genau dann wenn $(\sum_{v\in\Omega} q_v^2) \geq 1$ ist. Wenn dies der Fall ist, dann beschreibt die Kombination von festgehaltenen Koordinaten ein Quaternion q, das außerhalb der 4D-Einheitssphäre liegt und damit die Normierungs-Randbedingung verletzt. Da wir jedoch nur an Lösungen auf der Oberfläche der Einheitssphäre interessiert sind, können wir diese Kombination von festgehaltenen Koordinaten in diesem Fall einfach verwerfen[5]. Der Algorithmus ist in Abb. 2.17 dargestellt.

Da die Anzahl der Dimensionen konstant ist (im Fall von Quaternionen ist die Anzahl der Dimensionen immer $D = 4$), arbeitet der beschriebene Algorithmus in konstanter Zeit $T \in \mathcal{O}(1)$. Da er auf jeden Punkt C_l der Anfrage-Zeitreihe C angewendet werden muss, um die Lower Bound $\text{qLB}_{\text{Keogh}}$ auszurechnen, liegt die asymptotische Gesamtlaufzeit für $\text{qLB}_{\text{Keogh}}$ in $\text{T}_{\text{qLB\_Keogh}} \in \mathcal{O}(|C|)$, ist also linear in der Länge der Anfrage-Zeitreihe.

Trotz dieser Tatsache ist der Aufwand *pro Punkt*, die Lower Bound $\text{qLB}_{\text{Keogh}}$ auszurechnen relativ hoch: Im ungünstigen Fall finden *für einen Einzelnen Punkt* folgende Berechnungen statt:

- $4\cdot 3\cdot 2\cdot 1\cdot 2^4 = 384$ Schleifendurchgänge insgesamt
- $(4\cdot 2^1)\cdot 1 + (4\cdot 3\cdot 2^2)\cdot 1 + (4\cdot 3\cdot 2\cdot 2^3)\cdot 1 = 248$ Aufrufe der Wurzelfunktion
- $1+(4\cdot 2^1)\cdot 2+(4\cdot 3\cdot 2^2)\cdot 2+(4\cdot 3\cdot 2\cdot 2^3)\cdot 2+(4\cdot 3\cdot 2\cdot 1\cdot 2^4)\cdot 1 = 881$ Aufrufe der IS_IN_BOX()-Funktion, welche jeweils bis zu 8 Vergleichsoperationen durchführt, was insgesamt bis zu 7048 Vergleichsoperationen bedeutet.

In der Praxis benötigt die Berechnung von $\text{qLB}_{\text{Keogh}}$ bei einem einzelnen Punkt im Mittel etwa 150 mal so lange, wie die Relaxierung eines Knotens bei cDTW. Dies zeigt, dass

[5] Bedingt durch Rundungsfehler aufgrund der endlichen numerischen Präzision kann der Nenner auch dann negativ werden, wenn q eine gültige Lösung innerhalb des Envelopes ist. Diese Fälle müssen in Wirklichkeit ebenfalls berücksichtigt werden. Dadurch werden in seltenen Fällen (bei den von uns untersuchten Daten $\leq 0.5\%$) Lösungen verworfen. Dies wirkt sich allerdings nur marginal auf die Pruning-Raten aus (Unterschied $< 10^{-5}$).

```
1: function qLB_Keogh(L, U, C)
2:     if quat_is_in_box(L,U,C) then                          ▷ Liegt C komplett im Envelope?
3:         return 0                                           ▷ ⇒ Der Beitrag des Punktes C ist (mindestens) 0.
4:     end if
5:
6:     result ← −1.0                                          ▷ initialisiere Maximum
7:     for fix_i ∈ {0,1,2,3} do                               ▷ wähle erste Koordinate, die festgehalten wird
8:         for fix_A ∈ {L[fix_i], U[fix_i]} do                ▷ setze diese Koordinate auf den Rand von [L,U]
9:             λ ← sqrt((1−C[fix_i]^2)/(1−fix_A^2))           ▷ berechne Lagrange-Multiplikator (λ = ±√···)
10:            Q+[fix_i] ← fix_A                              ▷ berechne Lösungen für λ = +√··· und λ = −√···
11:            Q−[fix_i] ← fix_A
12:            for v ∈ {0,1,2,3}\{fix_i} do
13:                Q+[v] ← +C[v]/λ
14:                Q−[v] ← −C[v]/λ
15:            end for
16:            if quat_is_in_box(L,U,Q+) then                 ▷ Lösung Q+ liegt im Envelope?
17:                result ← max(result, |C · Q+|)             ▷ aktualisiere Maximum
18:            end if
19:            if quat_is_in_box(L,U,Q−) then                 ▷ Lösung Q− liegt im Envelope?
20:                result ← max(result, |C · Q−|)             ▷ aktualisiere Maximum
21:            end if
22:
23:                ▷ Wenn eine der gefundenen Lösungen Q+ bzw. Q− bisher noch nicht im Envelope liegt...
24:            if not quat_is_in_box(L,U,Q+) or not quat_is_in_box(L,U,Q−) then
25:                for fix_k ∈ {0,1,2,3}\{fix_i} do           ▷ setze weitere Koordinate fest...
26:                    for fix_B ∈ {L[fix_k], U[fix_k]} do    ▷ ...auf den Rand von [L,U]
27:                        λ ← sqrt((1−C[fix_i]^2−C[fix_k]^2)/(1−fix_A^2−fix_B^2))   ▷ berechne Lagrange-Multiplikator
28:                        ··· ⇒ berechne wieder die Lösungen Q+ und Q−
29:                        ··· ⇒ überprüfe erneut, ob Q+ bzw. Q− nun im Envelope liegen
30:                        ··· ⇒ aktualisiere ggf. das Maximum
31:                        ··· ⇒ setze ggf. weitere Koordinate(n) fest
32:                        ··· ⇒ fahre analog fort bis
33:                        ...  1) beide Lösungen Q+ und Q− im Envelope liegen oder
34:                        ...  2) alle 4 Koordinaten fest gesetzt sind
35:                    end for
36:                end for
37:            end if
38:        end for
39:    end for
40:
41:    if result ≥ 0 then                                     ▷ Es wurde mindestens eine gültige Lösung gefunden.
42:        return arccos(result)
43:    else                                                   ▷ Es wurde KEINE gültige Lösung gefunden.
44:        return 0                                           ▷ Die „triviale Lower Bound“ 0 wird zurückgegeben.
45:    end if
46: end function
```

Abbildung 2.17: Pseudo-Code für die Berechnung von qLB_{Keogh}

```
1:  function QUAT_IS_IN_BOX(L, U, C)
2:      function IS_IN_BOX_4D(L, U, C)
3:          dim ← 4                                   ▷ Quaternionen haben 4 Dimensionen.
4:          for μ = 0 → dim − 1 do
5:              if (C[μ] < L[μ]) or (C[μ] > U[μ]) then   ▷ Liegt C_μ außerhalb des Intervalls [L_μ, U_μ] ?
6:                  return false
7:              end if
8:          end for
9:          return true                               ▷ C_μ liegt innerhalb des Intervalls [L_μ, U_μ] ∀μ ∈ {0,1,2,3}.
10:     end function
11:
12:     return is_in_box_4D(L, U, C) or is_in_box_4D(L, U, −C)   ▷ Antipodalität!
13: end function
```

Abbildung 2.18: Pseudo-Code für die Funktion, die überprüft, ob ein Quaternion C innerhalb einer gegebenen 4-dimensionalen Box $[L,U]$ liegt. Hierbei muss auch der Fall überprüft werden, ob das zu C antipodale Quaternion $-C$ in der Box liegt.

sich der Einsatz dieser Lower Bound erst bei sehr langen Zeitreihen, bzw. großem Warping-Fenster amortisiert, wenn die lediglich lineare Abhängigkeit der Laufzeit von $\text{qLB}_{\text{Keogh}}$ von der Zeitreihenlänge zum Tragen kommt gegenüber der Laufzeit von cDTW, die quadratisch von der Länge der Zeitreihe abhängt. Eine einfache Abschätzung dafür, ab welcher Zeitreihenlänge und Fensterbreite cDTW mehr Zeit benötigt als $\text{qLB}_{\text{Keogh}}$ liefert folgende Überlegung:

Wenn die Relaxierung eines einzelnen Knotens von cDTW um einen Faktor p schneller ist als die Berechnung von $\text{qLB}_{\text{Keogh}}$ bei einem einzelnen Punkt, dann muss für die Anzahl N_{cDTW} der cDTW-Relaxierungen und die Anzahl $N_{\text{qLB\_Keogh}}$ der $\text{qLB}_{\text{Keogh}}$-Berechnungen Folgendes gelten, damit sich die Anwendung der Lower Bound amortisiert:

$$\begin{aligned} N_{\text{cDTW}} &\geq p \cdot N_{\text{qLB\_Keogh}} \\ \sum_{i=0}^{|C|-1} \sum_{j=l(i)}^{h(i)} 1 &\geq p \cdot |C| \end{aligned}$$

Die Grenzen des Spaltenindexes j sind hierbei $l(i) = \max(0, i - W_{\text{eff}})$ und $h(i) = \min(|C| - 1, i + W_{\text{eff}})$. Mit W_{eff} ist hierbei die effektive (halbe) Breite des Warping-Fensters gemeint, die als prozentualer Anteil der Zeitreihenlänge gewählt wird: $W_{\text{eff}} = \frac{W \cdot |C|}{100}$. Es ergeben sich für $p \approx 150$ folgende Abschätzungen: Für $|C| = 300$ muss das Warping-Fenster mindestens 30% betragen, für $|C| = 200$ bereits 50% und für $|C| = 100$ amortisiert sich $\text{qLB}_{\text{Keogh}}$ selbst für $W = 100\%$ – also unconstrained DTW! – nicht.

Aus diesem Grund haben wir eine schnelle Approximation an diese Lower Bound entwickelt, die wir im Folgenden vorstellen werden: $\text{qLB}_{\text{Keogh\_interval}}$. Die hier im Detail beschrie-

bene exakte Lower Bound $\text{qLB}_{\text{Keogh}}$ werden wir im Anschluss verwenden, um die Güte der approximativen Lösung im direkten Vergleich mit der exakten Lower Bound zu beurteilen.

$\mathbf{qLB_{Keogh\_interval}}$

Wie bereits vorangehend beschrieben, müssen wir eine untere Schranke für das geodätische Abstandsmaß

$$d_{\mathbb{H}}(q_1, q_2) := \arccos|\langle q_1|q_2\rangle|$$

zwischen zwei Quaternionen q_1 und q_2 finden. Durch die Bedingung normierter Quaternionen ist das Argument der arccos-Funktion $|\langle q_1|q_2\rangle| \in [0,1]$ und damit ist

$$\min_{q_i \in \mathbb{H}_1} \arccos|\langle q_i|C_i\rangle| = \arccos \max_{q_i \in \mathbb{H}_1} |\langle q_i|C_i\rangle|.$$

Wir suchen in Definition 19 also das Minimum der arccos-Funktion, was für die Einschränkung normierter Quaternionen gleich dem Maximum des Arguments entspricht. Da wir wissen, dass q_μ in jeder Dimension $\mu \in \{0,1,2,3\}$ im Intervall $[L_\mu, U_\mu]$ liegen muss, können wir das Maximum mit Hilfe der Intervall-Arithmetik berechnen.

Auch in der folgenden Herleitung wird für eine bessere Lesbarkeit der Punktindex i weglassen. Mit q, C und $[L,U]$ sind also jeweils q_i, C_i und $[L_i, U_i]$ für $i \in \{0,\ldots,|C|-1\}$ gemeint.

Das Maximum des Skalarproduktes $\max|\langle q|C\rangle|$ mit der Bedingung, dass $q_\mu \in [L_\mu, U_\mu]\ \forall \mu \in \{0,1,2,3\}$ lässt sich wie folgt mit Hilfe von Intervallen darstellen:

$$\begin{aligned}\max_{q\in[L,U]} |\langle q|C\rangle| &= \max_{q\in[L,U]} |\sum_{\mu=0}^{3} q_\mu \cdot C_\mu| \\ &= \max|\sum_{\mu=0}^{3} ([L_\mu, U_\mu]\cdot[C_\mu, C_\mu])|\end{aligned}$$

Hierbei suchen wir also das Quaternion q innerhalb des Envelopes $[L,U]$, welches das größtmögliche Skalarprodukt zu dem Anfrage-Quaternion C bildet (und damit mit diesem den kleinstmöglichen Winkel einschließt). Da wir wissen, dass jede Komponente q_μ im Intervall $[L_\mu, U_\mu]$ liegt, lässt sich q_μ durch dieses Intervall abschätzen. Da C_μ bekannt ist, also keine Intervall-Abschätzung benötigt, wird diese Komponente durch das entartete Intervall $[C_\mu, C_\mu]$ dargestellt. Dieser gesamte Ausdruck lässt sich nun mit Hilfe von Intervall-Arithmetik lösen.

Man beachte, dass hierbei die Normierungsbedingung $\|q\| = 1$ fallen gelassen wird. Aus diesem Grunde muss vor der Anwendung der arccos-Funktion das Ergebnis noch gegen 1.0 nach oben beschränkt werden.

Wir benötigen nun folgende Operationen aus der Intervall-Arithmetik:

Definition 20: Intervall-Arithmetik (Summe; Produkt; Absolutbetrag)

Sei $[x_1, x_2]$ mit $x_1, x_2 \in \mathbb{R}$ und $x_1 \leq x_2$ ein reellwertiges, geschlossenes Intervall:

$$[x_1, x_2] := \{x \in \mathbb{R} \mid x_1 \leq x \leq x_2\} \subseteq \mathbb{R}$$

Sei $[y_1, y_2] \subseteq \mathbb{R}$ ebenfalls ein reellwertiges, geschlossenes Intervall. Nun lassen sich folgende Operationen definieren:

- Summe:
  $[x_1, x_2] + [y_1, y_2] := [x_1 + y_1, x_2 + y_2]$
- Produkt:
  $[x_1, x_2] \cdot [y_1, y_2] := [\min(x_1 \cdot y_1, x_1 \cdot y_2, x_2 \cdot y_1, x_2 \cdot y_2), \max(x_1 \cdot y_1, x_1 \cdot y_2, x_2 \cdot y_1, x_2 \cdot y_2)]$
- Absolutbetrag:
  $$|[x_1, x_2]| := \begin{cases} [x_1, x_2] & \text{wenn} \quad x_1 \geq 0, x_2 \geq 0 \\ [0, \max(|x_1|, |x_2|)] & \text{wenn} \quad x_1 < 0, x_2 \geq 0 \\ [|x_2|, |x_1|] & \text{wenn} \quad x_1 < 0, x_2 < 0 \end{cases}$$

Mit Hilfe dieser Operationen lässt sich das Maximum des Skalarproduktes nun wie folgt berechnen:

$$\begin{aligned}
\max_{q \in [L,U]} |\langle q|C\rangle| &= \max|\sum_{\mu=0}^{3}([L_\mu, U_\mu] \cdot [C_\mu, C_\mu])| \\
&= \max|\sum_{\mu=0}^{3}[\min(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu), \max(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu)]| \\
&= \max|[\sum_{\mu=0}^{3}\min(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu), \sum_{\mu=0}^{3}\max(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu)]| \\
&= \max([\ldots, \max(|\sum_{\mu=0}^{3}\min(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu)|, |\sum_{\mu=0}^{3}\max(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu)|)]) \\
&= \max(|\sum_{\mu=0}^{3}\min(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu)|, |\sum_{\mu=0}^{3}\max(L_\mu \cdot C_\mu, U_\mu \cdot C_\mu)|)
\end{aligned}$$

Bei der Auswertung des Absolutbetrages im vorletzten Schritt wurde lediglich die rechte Grenze des resultierenden Intervalls ausgewertet und im letzten Schritt wurde dann ausgenutzt, dass das Maximum eines Intervalls gleich seiner rechten Grenze ist. Diese beiden Schritte sind aus folgendem Grund zulässig:

Gegeben ein Intervall $[x_1, x_2]$. Wir bilden nun den Absolutbetrag dieses Intervalls gemäß

Definition 20:

$$[L,R] := |[x_1,x_2]| = \begin{cases} [x_1,x_2] & \text{wenn} \quad x_1 \geq 0, x_2 \geq 0 \\ [0,\max(|x_1|,|x_2|)] & \text{wenn} \quad x_1 < 0, x_2 \geq 0 \\ [|x_2|,|x_1|] & \text{wenn} \quad x_1 < 0, x_2 < 0 \end{cases}$$

Während die linke Grenze L des resultierenden Intervalls je nach Lage von x_1 und x_2 unterschiedlich zu bestimmen ist, kann man die rechte Grenze R für alle drei Fälle mit $R = \max(|x_1|,|x_2|)$ bestimmen. Da wir jedoch ausschließlich am Maximum des Intervalls interessiert sind, reicht es, die rechte Grenze auszuwerten, denn es gilt:

$$\max([L,R]) = \max(\{x \in \mathbb{R} \mid L \leq x \leq R\}) = R$$

Deshalb braucht die linke Grenze des resultierenden Intervalls nicht ausgewertet werden. Damit lässt sich $\text{qLB}_{\text{Keogh\_interval}}$ nun wie folgt definieren:

Definition 21: $\text{qLB}_{\text{Keogh\_interval}}$

Seien Q und C zwei gleich lange Zeitreihen. Die Punkte der Zeitreihen seien normierte Rotations-Quaternionen: $Q_i, C_i \in \mathbb{H}_1$. Sei ferner $[L,U]_Q$ der *Envelope* um Q, dann ist $\text{qLB}_{\text{Keogh\_interval}}$ definiert als:

$$\text{qLB}_{\text{Keogh\_interval}}([L,U]_Q, C) := \sum_{i=0}^{|Q|-1} \arccos(\min(1.0, \max(|\tilde{L}_i|, |\tilde{U}_i|)))$$

mit

$$\tilde{L}_i = \sum_{\mu=0}^{3} \min(L_{i,\mu} \cdot C_{i,\mu}, U_{i,\mu} \cdot C_{i,\mu}) \quad \text{und} \quad \tilde{U}_i = \sum_{\mu=0}^{3} \max(L_{i,\mu} \cdot C_{i,\mu}, U_{i,\mu} \cdot C_{i,\mu})$$

Diese Lower Bound ist nach wie vor in linearer Zeit $\mathcal{O}(|Q|)$ auszurechnen. Der Rechenaufwand pro Punkt ist allerdings viel kleiner als bei $\text{qLB}_{\text{Keogh}}$: Es werden lediglich 4 Schleifendurchläufe benötigt mit jeweils nur zwei Vergleichsoperationen. Mit den äußeren beiden min- und max-Aufrufen werden insgesamt pro Punkt nur 10 Vergleichsoperationen benötigt.

2.4.3 Ergebnisse: Pruning-Raten

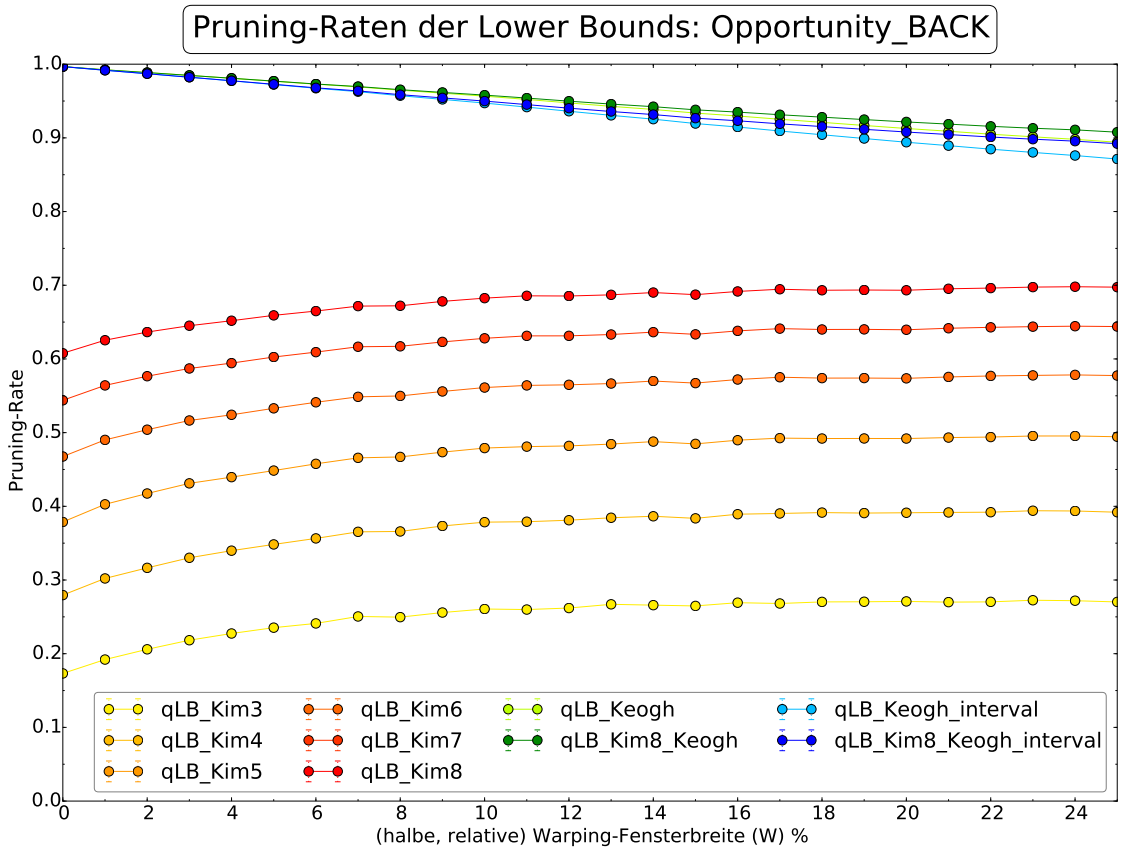


Abbildung 2.19: Pruning-Raten der Lower Bounds am Beispiel des Datensatzes Opportunity, Sensor: BACK. Der Messfehler entspricht dem Standardfehler über (bis zu) 10 Messungen. Dieser ist jedoch so klein (≤ 0.002), dass die Fehlerbalken komplett von den Markern verdeckt werden.

Um die Effizienz einer Lower Bound quantitativ auswerten zu können, betrachtet man die sogenannte **Pruning-Rate**. Diese ist hierbei einfach das Verhältnis aus der Anzahl der Vergleiche, bei denen die jeweilige Lower Bound **greift**, also größer oder gleich dem besten bisher gefundenen Penalty-Wert (BSF) ist, geteilt durch die Gesamt-Anzahl der Vergleiche:

Definition 22: Pruning-Rate

$$r_{\text{pruning}} := \frac{\#(\text{LB} \geq \text{BSF})}{\#(\text{Vergleiche insgesamt})}$$

hierbei ist LB der Wert der Lower Bound und BSF der beste (=kleinste), bisher gefundene Abstandswert.

Die Pruning-Raten der verschiedenen Lower Bounds wurden auf allen Datensätzen „Op-

portunity“, „QBGR“, „6DMG“, „JGU_Numbers“ und „qCBF“ jeweils für alle Sensoren getrennt getestet. Dabei wurde jede einzelne Shape-Instanz per 1NN mit jeder der übrigen Shape-Instanzen verglichen. Insgesamt ergeben sich damit für einen Datensatz, der aus N Shapes besteht $N \cdot (N-1)$ Vergleiche. Da die Pruning-Rate von der Reihenfolge der Vergleiche abhängt, wurde die Reihenfolge der Vergleiche für jede Shape-Instanz zufällig gewählt. Die Pruning-Raten für verschiedene Lower Bounds bei verschiedenen Warping-Fensterbreiten $0\% \leq W \leq 25\%$ sind in Abb. 2.19, 2.20 und 2.21 dargestellt. W ist hierbei genau genommen die halbe, relative Warping-Fensterbreite. Für eine Zeitreihen-Länge |Q| (beide Zeitreihen werden auf die gleiche Größe gestreckt bzw. gestaucht) beträgt die effektive Warping-Fensterbreite $W_{\text{eff}} = 2 \cdot \frac{W}{100} \cdot |Q| + 1$. Es wurden dabei für jeden Wert von W (bis zu) 10 Durchgänge durchlaufen und als Ergebnis der Mittelwert der Pruning-Raten über alle Durchgänge dargestellt. Durch die große Anzahl an Shapes in jedem Datensatz schwanken die Pruning-Raten bei gleichem Parameter W jedoch sehr wenig, sodass der Mittelwert μ bereits nach einem einzigen Durchlauf nahezu konstant bleibt und die Standard-Abweichung σ immer unter 0.005 bleibt, was für N=10 einem Standard-Fehler ($\sigma_{\text{std}} = \frac{\sigma}{\sqrt{N}}$) von unter 0.002 entspricht. Wie man in Abb. 2.19ff. erkennt, sind die Ergebnisse der Pruning-Raten für alle verwendeten Datensätze und Sensoren relativ gleich.

Die Pruning-Raten von $\text{qLB}_{\text{Kim,k}}$ liegen für kleine Werte von k deutlich unter denen von $\text{qLB}_{\text{Keogh}}$. Nur bei einem einzigen Teildatensatz (QBGR_S19, siehe Abb. 2.20 unten rechts) erreicht $\text{qLB}_{\text{Kim,k}}$ für $k \geq 7$ bessere Pruning-Raten als $\text{qLB}_{\text{Keogh}}$. Allerdings muss man dabei beachten, dass der Rechenzeit-Aufwand für die Berechnung von $\text{qLB}_{\text{Kim,k}}$ zwar konstant in der Länge der Zeitreihe $|Q|$ ist, jedoch quadratisch mit dem Parameter k wächst. Sind für $k = 4$ lediglich $4^2 \cdot 2 = 32$ Matrix-Relaxierungen nötig, so sind es für $k = 8$ bereits $8^2 \cdot 2 = 128$. Deshalb sollte der Parameter k nicht zu hoch gewählt werden. Des Weiteren hängt die Pruning-Rate für $\text{qLB}_{\text{Kim,k}}$ empfindlich von der Länge der Zeitreihen ab. Je länger die Zeitreihen sind, desto kleiner ist der relative Anteil der Penalty-Matrix, der von $\text{qLB}_{\text{Kim,k}}$ (für festes k) ausgewertet wird: Da jeweils zwei Teil-Matrizen der Größe k^2 ausgewertet werden, ist der relative Anteil somit $r(\text{qLB}_{\text{Kim,k}}, |Q|) = \frac{2 \cdot k^2}{|Q|^2}$ und skaliert damit reziprok mit dem Quadrat der Zeitreihenlänge. Dementsprechend sinkt die Pruning-Rate bei längeren Zeitreihen stark ab, da der von $\text{qLB}_{\text{Kim,k}}$ ausgewertete Anteil der gesamten Penalty-Matrix immer weniger Gewicht hat. Dafür hat $\text{qLB}_{\text{Kim,k}}$ den Vorteil, dass es – bezogen auf die Länge $|Q|$ der Zeitreihen – in konstanter Zeit $\text{T}_{\text{qLB\_Kim,k}} \in \mathcal{O}(1)$ (bei festem k) ausgewertet werden kann, während für $\text{qLB}_{\text{Keogh}}$ lineare Rechenzeit für die Auswertung benötigt wird: $\text{T}_{\text{qLB\_Keogh}} \in \mathcal{O}(|Q|)$.
In Tabelle 2.1 sind die Pruning-Raten-Differenzen

$$\Delta r_{\text{pruning}} := r_{\text{pruning}}(\text{qLB}_{\text{Keogh}}) - r_{\text{pruning}}(\text{qLB}_{\text{Keogh\_interval}})$$

und die Laufzeit-Verhältnisse

$$r_{\text{runtime}} := \frac{T_{\text{runtime}}(\text{qLB}_{\text{Keogh}})}{T_{\text{runtime}}(\text{qLB}_{\text{Keogh\_interval}})}$$

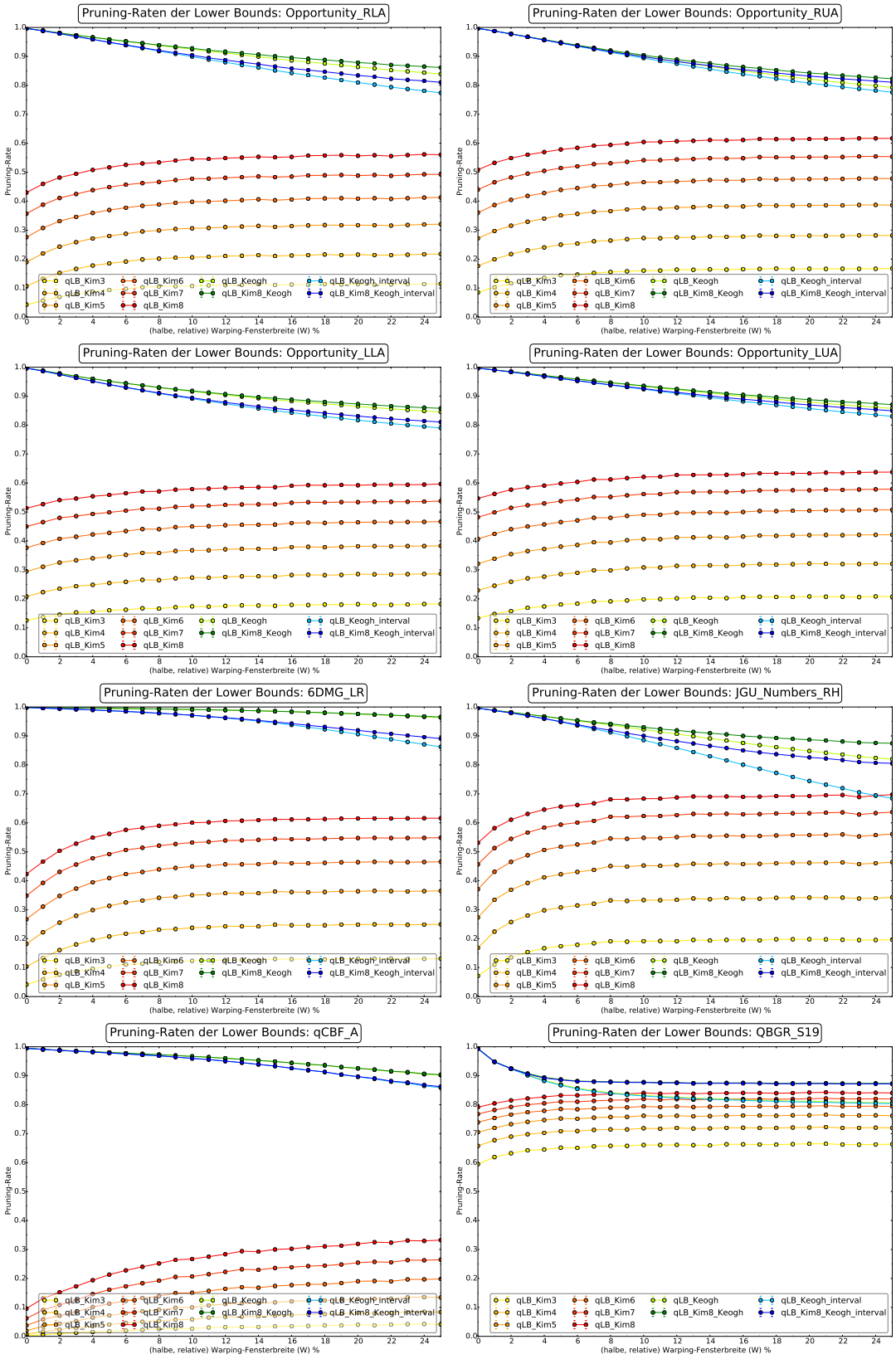


Abbildung 2.20: Pruning-Raten der Lower Bounds für alle weiteren Datensätze. Man sieht, dass die Pruning-Raten bei allen Datensätzen und Sensoren sehr ähnlich sind.

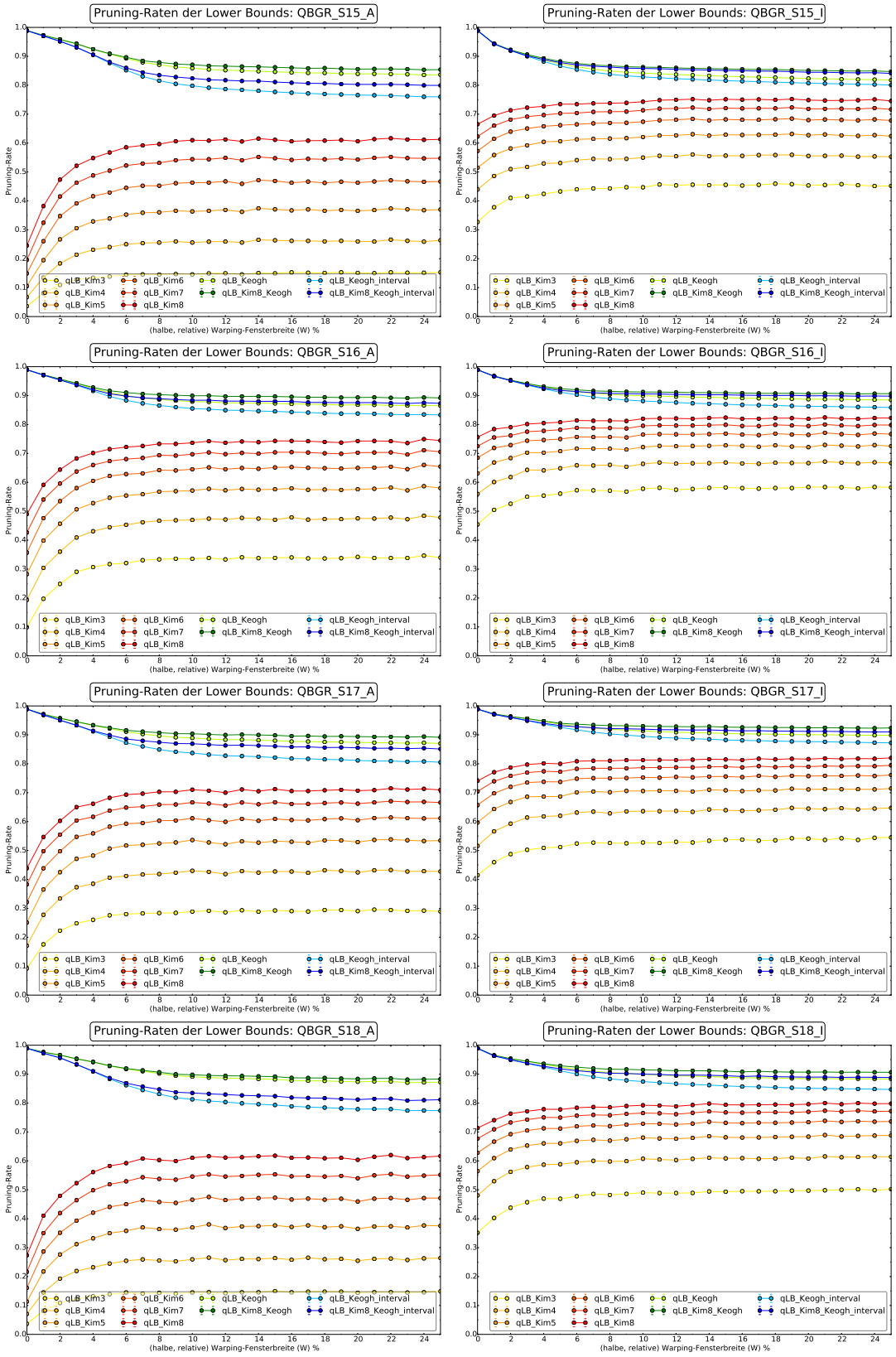


Abbildung 2.21: Pruning-Raten der Lower Bounds für alle weiteren Datensätze (Fortsetzung)

für alle Datensätze und Sensoren getrennt aufgeführt. Die Werte stellen Mittelwert und Standard-Fehler über die relativen Warping-Fensterbreiten $W \in \{0,\ldots,25\}$ dar. Die Laufzeiten wurden jeweils auf einem einzelnen Kern des MOGON HPC-Clusters[6] bestimmt. Hierbei handelt es sich um AMD Opteron 6272 CPUs mit einer Taktfrequenz von 2.1 GHz [27].

Mit $\mathrm{qLB}_{\mathrm{Keogh}}$, sowie der schnellen Intervall-Approximation $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$ wurden bei allen Datensätzen Pruning-Raten von über 80% erreicht. Bei kleinen Warping-Fenster-Breiten $W \leq 5$ betragen die Pruning-Raten sogar über 90%. Dies bedeutet, dass nur bei 10% bis 20% der Vergleiche zwischen zwei Zeitreihen das „teure“ DTW-Abstandsmaß ausgewertet werden muss.

Man erkennt deutlich, dass die exakte Variante von $\mathrm{qLB}_{\mathrm{Keogh}}$ und die Intervall-Approximation $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$ sehr nahe beieinander liegen. Auf den getesteten Daten war die Performanz bezüglich der Pruning-Rate von $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$ gegenüber $\mathrm{qLB}_{\mathrm{Keogh}}$ maximal 7.03% schlechter (QBGR S18_A), im Mittel aller getesteten Datensätze jedoch nur 2.77%.

Die Laufzeit-Verhältnisse zwischen $\mathrm{qLB}_{\mathrm{Keogh}}$ und $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$ sind beachtlich. So braucht $\mathrm{qLB}_{\mathrm{Keogh}}$ im Durchschnitt ca. 70-mal so lange wie die approximative Variante $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$! Dies zeigt, dass $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$ wesentlich effizienter auswertbar ist, als die exakte Lower-Bound $\mathrm{qLB}_{\mathrm{Keogh}}$ und dabei nur unwesentlich schlechtere Pruning-Raten erzeugt.

Kombiniert man nun $\mathrm{qLB}_{\mathrm{Kim,8}}$ und $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$ in einer Lower-Bound-Kaskade, lassen sich die Pruning-Raten noch weiter verbessern. Bei dieser Kaskade wird zunächst $\mathrm{qLB}_{\mathrm{Kim,8}}$ ausgewertet. Wenn diese nicht greift, so wird im Anschluss $\mathrm{qLB}_{\mathrm{Keogh\_interval}}$ ausgewertet. Erst wenn diese ebenfalls nicht greift, muss der Abstand mit Hilfe von DTW berechnet werden. Die Ergebnisse sind in Tabelle 2.2 aufgelistet. Die dargestellten Pruning-Raten-Differenzen sind hier:

$$\Delta r_{\mathrm{pruning}} := r_{\mathrm{pruning}}(\mathrm{qLB}_{\mathrm{Kim,8}} \rightarrow \mathrm{qLB}_{\mathrm{Keogh\_interval}}) - r_{\mathrm{pruning}}(\mathrm{qLB}_{\mathrm{Keogh\_interval}}).$$

Durch die Kaskade lässt sich die Gesamt-Pruning-Rate (im Mittel über alle getesteten Datensätze) um ca. weitere 2% verbessern. In einzelnen Fällen werden sogar über 4% bessere Pruning-Raten erreicht (QBGR S19 und JGU_Numbers RH).

[6] Der Autor bedankt sich hiermit ausdrücklich für die zur Verfügung gestellte Rechenzeit auf dem Supercomputer MOGON der Johannes Gutenberg-Universität Mainz. (hpc.uni-mainz.de)

Datensatz	**Pruning-Raten-Differenz** ($\Delta r_{pruning}$)	**Laufzeit-Verhältnis** ($r_{runtime}$)
Opportunity BACK	$(1.16 \pm 0.14)\%$	71.11 ± 0.36
Opportunity RLA	$(3.24 \pm 0.41)\%$	71.61 ± 0.49
Opportunity RUA	$(0.80 \pm 0.11)\%$	71.20 ± 0.19
Opportunity LLA	$(2.97 \pm 0.34)\%$	66.30 ± 0.49
Opportunity LUA	$(1.33 \pm 0.17)\%$	74.74 ± 1.23
QBGR S15_A	$(5.40 \pm 0.49)\%$	78.93 ± 0.63
QBGR S15_I	$(1.20 \pm 0.10)\%$	68.73 ± 0.20
QBGR S16_A	$(2.19 \pm 0.20)\%$	70.44 ± 0.36
QBGR S16_I	$(1.71 \pm 0.17)\%$	65.88 ± 0.38
QBGR S17_A	$(4.64 \pm 0.40)\%$	64.58 ± 0.44
QBGR S17_I	$(1.76 \pm 0.15)\%$	70.09 ± 0.37
QBGR S18_A	$\mathbf{(7.03 \pm 0.61)\%}$	66.81 ± 0.58
QBGR S18_I	$(2.39 \pm 0.22)\%$	73.56 ± 0.25
QBGR S19	$(0.23 \pm 0.01)\%$	71.47 ± 0.53
6DMG LR	$(3.80 \pm 0.62)\%$	68.94 ± 1.39
JGU_Numbers RH	$(5.74 \pm 0.86)\%$	76.25 ± 0.33
qCBF A	$(1.51 \pm 0.27)\%$	75.81 ± 0.11
Mittelwert	$\mathbf{(2.77 \pm 0.46)\%}$	$\mathbf{70.97 \pm 0.93}$

Tabelle 2.1: Vergleich der Pruning-Raten und Laufzeiten von qLB_{Keogh} und $qLB_{Keogh\_interval}$ für alle untersuchten Datensätze und Sensoren.

Datensatz	**Pruning-Raten-Differenz** ($\Delta r_{pruning}$)
Opportunity BACK	$(0.69 \pm 0.13)\%$
Opportunity RLA	$(1.23 \pm 0.23)\%$
Opportunity RUA	$(1.21 \pm 0.23)\%$
Opportunity LLA	$(0.70 \pm 0.13)\%$
Opportunity LUA	$(0.62 \pm 0.12)\%$
QBGR S15_A	$(2.41 \pm 0.29)\%$
QBGR S15_I	$(2.65 \pm 0.28)\%$
QBGR S16_A	$(2.58 \pm 0.28)\%$
QBGR S16_I	$(2.36 \pm 0.27)\%$
QBGR S17_A	$(2.87 \pm 0.33)\%$
QBGR S17_I	$(2.34 \pm 0.27)\%$
QBGR S18_A	$(2.15 \pm 0.26)\%$
QBGR S18_I	$(2.51 \pm 0.29)\%$
QBGR S19	$(4.32 \pm 0.45)\%$
6DMG LR	$(0.63 \pm 0.17)\%$
JGU_Numbers RH	$(4.01 \pm 0.78)\%$
qCBF A	$(0.06 \pm 0.02)\%$
Mittelwert	$\mathbf{(1.96 \pm 0.29)\%}$

Tabelle 2.2: Vergleich der Pruning-Raten von $qLB_{Keogh\_interval}$ und der Lower-Bound-Kaskade $qLB_{Kim8} \rightarrow qLB_{Keogh\_interval}$ für alle untersuchten Datensätze und Sensoren.

2.5 Klassifikation

Auf den in Kapitel 1 vorgestellten Daten wird nun folgende Klassifizierungs-Aufgabe gelöst: Gegeben eine Menge von gelabelten Zeitreihen („Shapes“) unterschiedlicher Klassen. Gelabelt bedeutet, dass zu jedem Shape die Information bekannt ist, zu welcher Klasse es gehört. Dies ist wichtig, damit später die automatisiert gefundene Zuordnung eines Shapes zu einer Klasse verifiziert werden kann. Eine Klasse ist dabei eine Art der Bewegung (beim Opportunity-Datensatz etwa: „Kühlschrank öffnen“ oder „Lichtschalter betätigen“, beim QBGR-Datensatz z.B. „horizontaler Schlag“ oder „Hantel anheben“, beim 6DMG-Datensatz z.B. „wischen nach oben“ oder „stoßen nach rechts“, beim JGU_Numbers-Datensatz z.B. „Ziffer 0“ oder „Ziffer 1“ und beim qCBF-Datensatz „Cylinder“, „Bell“ oder „Funnel“). Die gegebenen Zeitreihen werden nun aufgeteilt in einen **Test-Datensatz** und einen **Trainings-Datensatz**.

Die Aufgabe besteht nun darin, dass jede Zeitreihe aus dem Test-Datensatz – unter Verwendung der Informationen aus dem Trainings-Datensatz – einer Klasse zugeordnet werden soll. Dazu gibt es verschiedene Ansätze. Aus dem Bereich des maschinellen Lernens gibt es viele bekannte Verfahren, die für eine Klassifikation verwendet werden können. Beispiele hierfür sind Support Vector Maschinen (SVMs) [28], Neuronale Netze (NNs) [29], Entscheidungsbäume (englisch: Decision Trees) [30] oder Random Forests [31]. Diese sind jedoch nicht immer leicht und intuitiv reimplementierbar und benötigen zudem eine hinreichend ausgeprägte Lern-Phase, um korrekt funktionieren zu können. Außerdem haben viele dieser Verfahren Parameter, die zuerst in einem Cross-Validierungs-Schritt bestimmt werden müssen. Des Weiteren benötigen diese Verfahren viele Trainings-Daten in der Lern-Phase um korrekt arbeiten zu können. Eine deutlich einfachere Möglichkeit der Klassifikation, die per se parameterfrei, sowie leicht zu implementieren ist [32] und auch mit kleineren Trainings-Datensätzen zurecht kommt, ist die sogenannte **One-Nearest-Neighbor-Suche (1NN-Suche)** [33]. Ein weiterer Vorteil ist dabei, dass sich bei diesem Verfahren sinnvolle Abstandsmaße zwischen Zeitreihen definieren lassen. Bei der 1NN-Suche wird jede Zeitreihe aus dem Test-Datensatz mit jeder Zeitreihe aus dem Trainings-Datensatz verglichen und jeweils deren „Abstand“ bestimmt. Für den „Abstand“ zwischen zwei Zeitreihen gibt es verschiedene Maße, von denen wir drei unterschiedliche vergleichen werden.

Das Vorgehen ist dabei wie folgt: Zuerst wird ein Shape aus dem Test-Datensatz ausgewählt. Es wird nun der Abstand dieses Test-Shapes zu jedem Shape des Trainings-Datensatzes bestimmt. Die Klasse des Test-Kandidaten wird dann festgelegt als die Klasse desjenigen Trainings-Shapes, welches den geringsten Abstand zu diesem Test-Kandidaten hat (also der *„nächste Nachbar“ (englisch: „nearest neighbor“)* des Test-Kandidaten in Bezug auf das gewählte Abstandsmaß). Da jedes Shape über ein Label verfügt, lässt sich somit einfach überprüfen, ob die gefundene Klasse korrekt ist oder nicht. Dieser Vorgang wird ebenfalls für alle übrigen Test-Shapes durchgeführt. Das Verhältnis aus der Anzahl der Shapes, die der

falschen Klasse zugeordnet wurden zur Gesamt-Anzahl der zu testenden Shapes ist hierbei der sogenannte **Klassifikations-Fehler**:

Definition 23: Klassifikations-Fehler

$$\varepsilon_{\text{classification}} := \frac{\#\text{ falsch zugeordneter Shapes}}{\#\text{ Shapes des Test-Datensatzes insgesamt}}$$

Der Klassifikations-Fehler ist somit auf das Intervall $\varepsilon_{\text{classification}} \in [0,1]$ beschränkt. Ein Klassifikations-Fehler von $\varepsilon_{\text{classification}} = 0$ würde also bedeuten, dass jedes Shape seiner korrekten Klasse zugeordnet wurde, während ein Klassifikations-Fehler von $\varepsilon_{\text{classification}} = 1$ bedeutet, dass jedes Shape einer falschen Klasse zugeordnet wurde.

Bei der Aufteilung der Daten in Trainings- und Test-Datensatz muss man beachten, dass der relative Anteil jeder einzelnen Klasse erhalten bleibt. Dies ist wichtig, damit das Ergebnis nicht durch ungünstig gewählte Aufteilungen verfälscht wird, in denen manche Klassen unter- oder überrepräsentiert sind. Sind beispielsweise in einem zu untersuchenden Datensatz mit zwei Klassen A und B 40% der Shapes aus der Klasse A und 60% aus der Klasse B, dann sollte dieses Verhältnis auch in den daraus erzeugten Teildatensätzen (Test- und Trainings-Datensatz) erhalten bleiben. Also sollte sowohl für den Test-Datensatz als auch für den Trainings-Datensatz gelten, dass jeweils 40% der Shapes der Klasse A und 60% der Klasse B angehören. Eine solche Aufteilung, die dieses Verhältnis erhält, wird auch als **stratifiziert** bezeichnet.

2.5.1 Klassifikation: Vergleich der Zeitreihen-Abstandsmaße

Wir untersuchen und vergleichen nun folgende Abstandsmaße zwischen jeweils zwei gegebenen Zeitreihen:

- Lockstep-Abstand (LS)
- (Sakoe-Chiba) constrained Dynamic Time Warping (cDTW)
- subsequence Dynamic Time Warping (ssDTW)

Während der Lockstep-Abstand und ssDTW parameterfrei sind, gibt es bei Sakoe-Chiba-constrained DTW einen Parameter, nämlich die Breite des Warping-Fensters W. Diesen haben wir mittels **Leave-One-Out-Cross-Validation** (LOOCV) bestimmt. Hierbei wird der Trainings-Datensatz erneut aufgespalten in zwei Teil-Datensätze. Der eine (LOOCV-Test) besteht dabei lediglich aus einem einzigen Shape, der andere (LOOCV-Training) aus den restlichen Shapes des ursprünglichen Trainings-Datensatzes. Die Anzahl der möglichen Aufteilungen eines Trainings-Datensatzes der Größe N in LOOCV-Test und LOOCV-Training ist damit gleich N: Jedes Shape des Trainings-Datensatzes bildet einmal den LOOCV-Test-

Datensatz, die verbleibenden $N-1$ Shapes bilden den zugehörigen LOOCV-Trainings-Datensatz. Für jeden zu untersuchenden Wert des Parameters $W \in \{0,\ldots,25\}$ wird nun für jede mögliche Aufteilung mittels LOOCV eine 1NN-Suche durchgeführt und schließlich der Parameterwert W_{best} zurückgeliefert, für den die 1NN-Suche den geringsten Klassifikations-Fehler auf den Trainings-Daten liefert. Dieser Wert wird dann für die eigentliche Berechnung des Klassifikations-Fehlers auf den Test-Daten verwendet.

Für die beiden Maße Lockstep (LS) und constrained Dynamic Time Warping (cDTW) werden alle Zeitreihen jeweils auf die selbe (mittlere) Länge gedehnt (bzw. gestaucht). Die Quaternionen-wertigen Rotationen werden hierfür mittels „spherical linear interpolation“ (SLERP) [6] interpoliert. Für subsequence Dynamic Time Warping (ssDTW) werden die Zeitreihen unverändert verwendet und jeweils die kleinere als Teilsequenz in der größeren gesucht. Für alle diese drei *Zeitreihen*-Abstandsmaße werden wiederum jeweils drei *punktweise* Abstandsmaße verwendet. Deren Vergleich erfolgt im nächsten Unterabschnitt. Im Folgenden dargestellt sind die sogenannten *Lern-Kurven* (englisch: *learning curves*). Dabei wird der Klassifikations-Fehler $\varepsilon_{\text{classification}}$ gegen die relative Größe des Test-Datensatzes

$$r_{\text{test}} := \frac{|\text{test}|}{|\text{test}| + |\text{training}|}$$

aufgetragen. Es ist trivial klar, dass mit steigender Größe des Test-Datensatzes die Größe des Trainings-Datensatzes abnimmt, da $r_{\text{test}} + r_{\text{train}} = 1$ sein muss. Der Klassifikations-Fehler wird jeweils über 10 stratifizierte Aufteilungen des Datensatzes gemittelt und der zugehörige Standard-Fehler als Fehlerbalken dargestellt. Es wird jeweils zwischen allen Klassen eines Datensatzes unterschieden. Beim Opportunity-Datensatz sind dies 17, beim QBGR-Datensatz 6, beim 6DMG-Datensatz 20, beim JGU_Numbers-Datensatz 10 und beim qCBF-Datensatz sind es 3 Klassen (siehe Kapitel 1).

Man erkennt (siehe Abb. 2.22 – 2.26), dass der Klassifikations-Fehler mit zunehmender Größe des Test-Datensatzes (und damit abnehmender Größe des Trainings-Datensatzes) zuerst leicht zunimmt. Diese Zunahme steigert sich ab $r_{\text{test}} \approx 0.7$ noch weiter. Dies ist auch nicht weiter verwunderlich, da in diesem Bereich zunehmend weniger Trainings-Daten zur Verfügung stehen. Eine korrekt stratifizierte Aufteilung der Daten ist in diesem Bereich zunehmend schwerer zu realisieren. Hat ein Datensatz beispielsweise 10 gleich große Klassen, der Trainings-Datensatz aber eine Größe von nur z.B. 105 Instanzen, dann sind dadurch 5 Klassen einmal mehr vorhanden als die anderen. Dieser Quantisierungs-Effekt wird immer größer, je kleiner der Trainings-Datensatz wird: Hat der Trainings-Datensatz beispielsweise nur noch 25 Instanzen, dann sind 5 Klassen und damit 50% der Klassen im Trainings-Datensatz um einen Faktor 1.5 überrepräsentiert. Hat der Trainings-Datensatz nur noch einen Umfang von 15 Instanzen, dann ist eine Hälfte der Klassen bereits doppelt so oft enthalten wie die andere Hälfte. Dadurch verschlechtern sich die Ergebnisse der Klassifikation natürlich deutlich, da dann ein Test-Kandidat bei der 1NN-Suche mit höherer

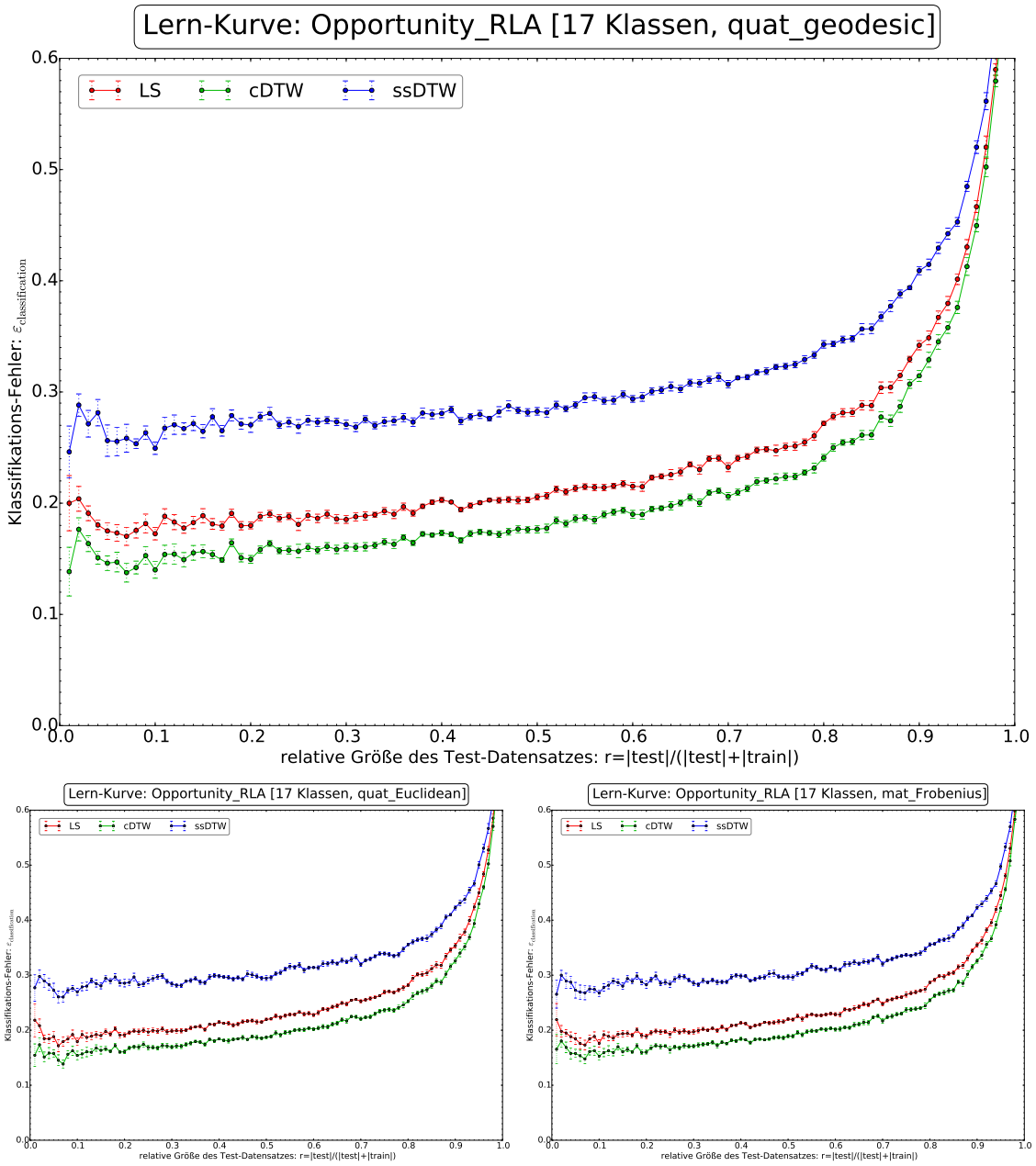


Abbildung 2.22: Lern-Kurve: Opportunity Datensatz – Sensor: RLA (rechter Unterarm).
Vergleich der **Zeitreihen**-Abstandsmaße: LS, cDTW, ssDTW. Dargestellt ist der Mittelwert des Klassifikations-Fehlers über 10 stratifizierte Aufteilungen des Datensatzes (mit dem Standard-Fehler als Fehlerbalken) gegen die relative Größe des Test-Datensatzes. Wie man an diesem Beispiel sieht, sind die Kurven für die verschiedenen *punktweisen* Abstandsmaße (quat_geodesic (**[oben]**), quat_Euclidean (**[unten links]**) und mat_Frobenius (**[unten rechts]**)) sehr ähnlich, weshalb im Weiteren exemplarisch nur die Ergebnisse unter dem *punktweisen* Abstandsmaß „quat_geodesic“ dargestellt wird. Die verschiedenen *punktweisen* Abstandsmaße werden im nächsten Abschnitt näher untersucht.

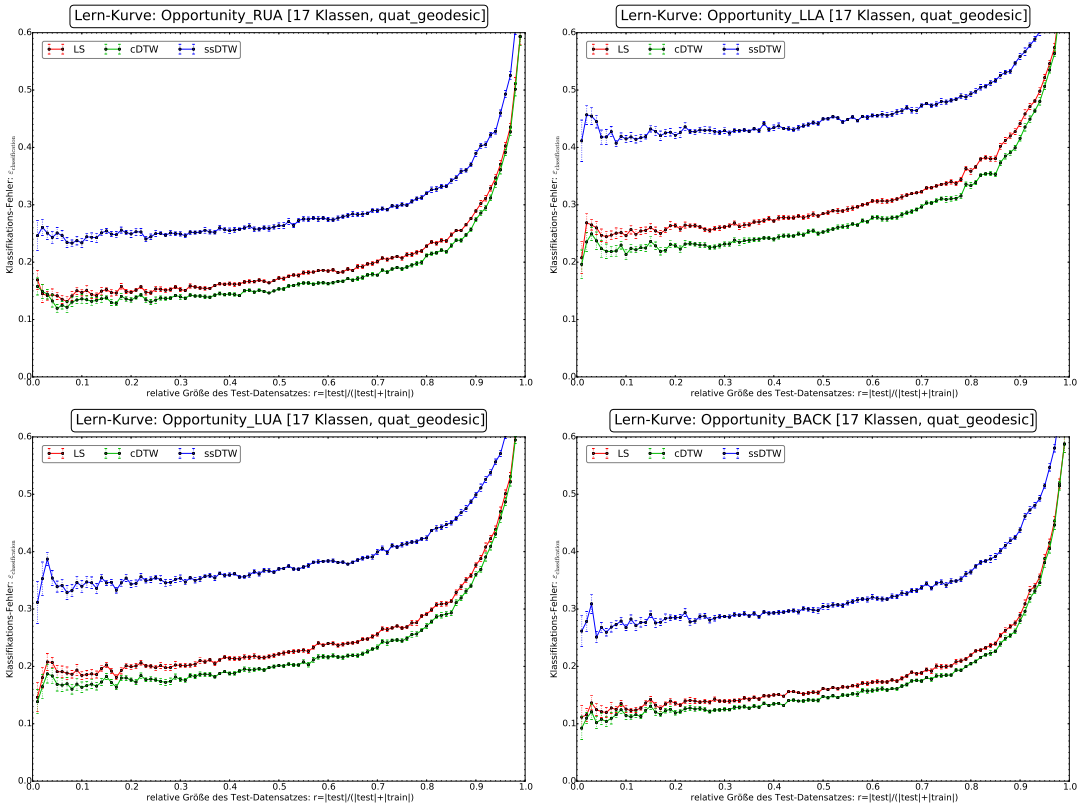


Abbildung 2.23: Lern-Kurve: Opportunity-Datensatz (Fortsetzung) – Sensoren: RUA, LLA, LUA, BACK. Vergleich der **Zeitreihen**-Abstandsmaße: LS, cDTW, ssDTW. (*punktweises* Abstandsmaß: „quat_geodesic“). Für alle Sensoren des Opportunity-Datensatzes schneidet ssDTW mit Abstand am schlechtesten ab. Lockstep und cDTW liefern deutlich bessere Ergebnisse, wobei cDTW immer etwas besser ist als Lockstep.

Wahrscheinlichkeit zu einer der überrepräsentierten Klassen zugeordnet wird. Ist der Trainings-Datensatz schließlich kleiner als die Anzahl der Klassen, sind manche Klassen gar nicht mehr enthalten. Eine stratifizierte Aufteilung ist dann gar nicht mehr möglich. Dieser entartete Fall wird hier allerdings nicht betrachtet. Die Größe des Trainings-Datensatzes wird immer ≥ 1% der Gesamtdaten gewählt und ist bei den untersuchten Datensätzen immer größer als die Anzahl der Klassen.

Es ist deutlich erkennbar, dass das ssDTW-Abstandsmaß in fast allen Fällen deutlich schlechter abschneidet als die beiden anderen Abstandsmaße LS und cDTW. Lediglich beim S15_A-Sensor des QBGR-Datensatzes (siehe Abb. 2.24 [oben]) liegt ssDTW zwischen LS und cDTW. Das elastische cDTW-Abstandsmaß liefert im Vergleich zum Lockstep-Abstandsmaß bei allen Datensätzen geringfügig bessere Klassifikations-Ergebnisse.

Um eine Vorstellung für die Größenordnung der zu erwartenden Klassifikations-Fehler zu bekommen, nehmen wir einmal an, es gäbe insgesamt N *nicht unterscheidbare* Klassen mit jeweils gleich vielen Instanzen (Shapes). Die Shapes des Test-Datensatzes würden dabei jeweils rein zufällig einer beliebigen Klasse zugeordnet werden. Der mittlere Klassifikations-Fehler bei diesem Vorgehen, welches wir als **„bloßes Raten“** bezeichnen wollen – also rein zufällig gewählter Klassenzuordnung – für N (gleich große) Klassen läge dann bei

$$\varepsilon_{\text{classification,random}}(N) = \frac{N-1}{N} \tag{2.5}$$

Dies wollen wir im Folgenden als theoretische Obergrenze für den Klassifikations-Fehler betrachten. Der tatsächliche Klassifikations-Fehler sollte bei einem korrekt arbeitenden Klassifikations-Algorithmus natürlich immer deutlich unter diesem Wert bleiben.

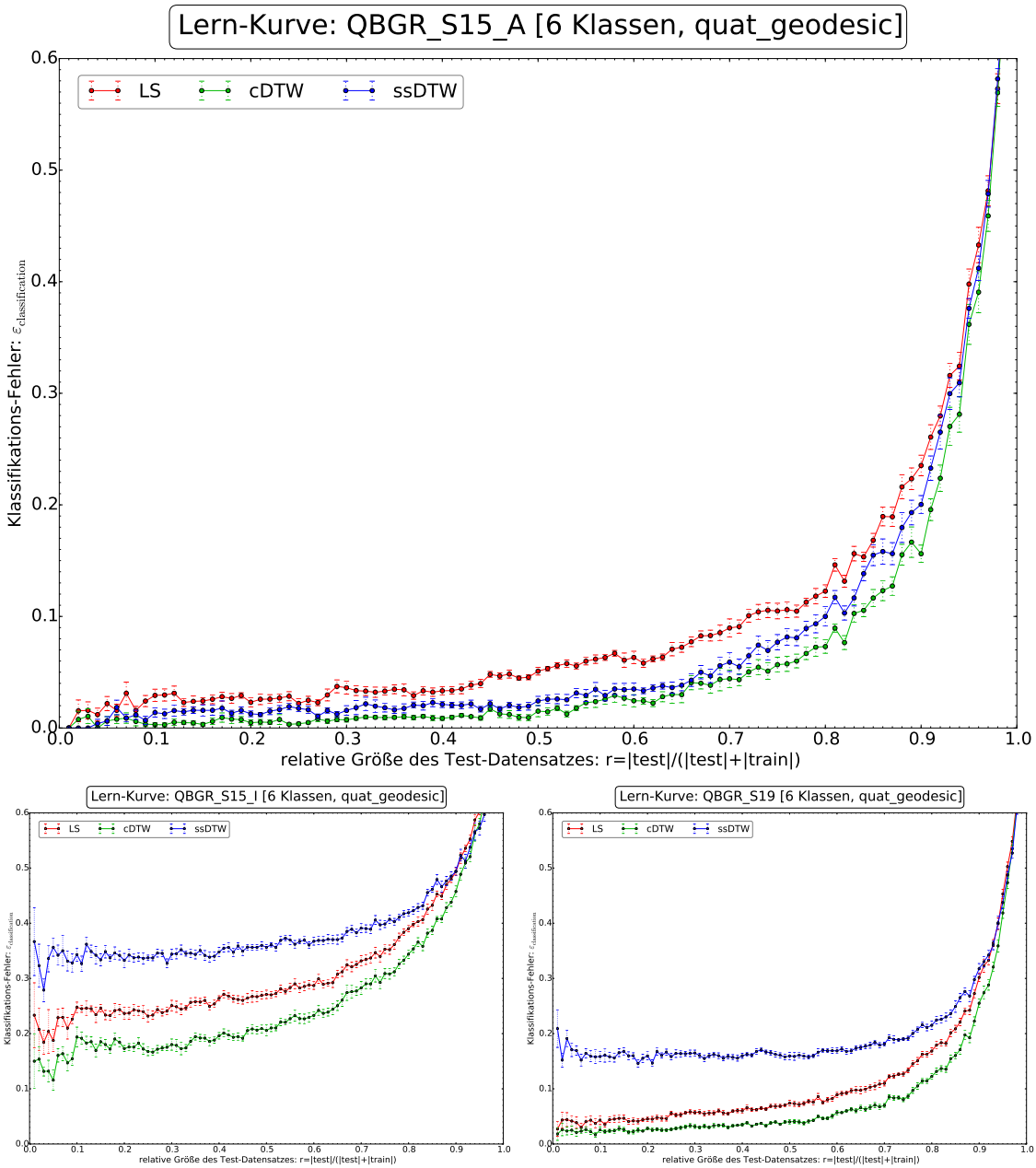


Abbildung 2.24: Lern-Kurve: QBGR-Datensatz – Sensoren: S15_A, S15_I und S19. Vergleich der **Zeitreihen**-Abstandsmaße: LS, cDTW, ssDTW (*punktweises* Abstandsmaß: „quat_geodesic“). Bei den Sensoren am aktiven Arm (Sxx_A) funktioniert die Klassifikation erwartungsgemäß deutlich besser als bei den Sensoren am inaktiven Arm (Sxx_I) bzw. am unteren Bauch (S19). Der Sensor S15_A ([**oben**]) ist der einzige Teil-Datensatz, bei dem ssDTW immer bessere Klassifikations-Ergebnisse liefert, als das Lockstep-Abstandsmaß.

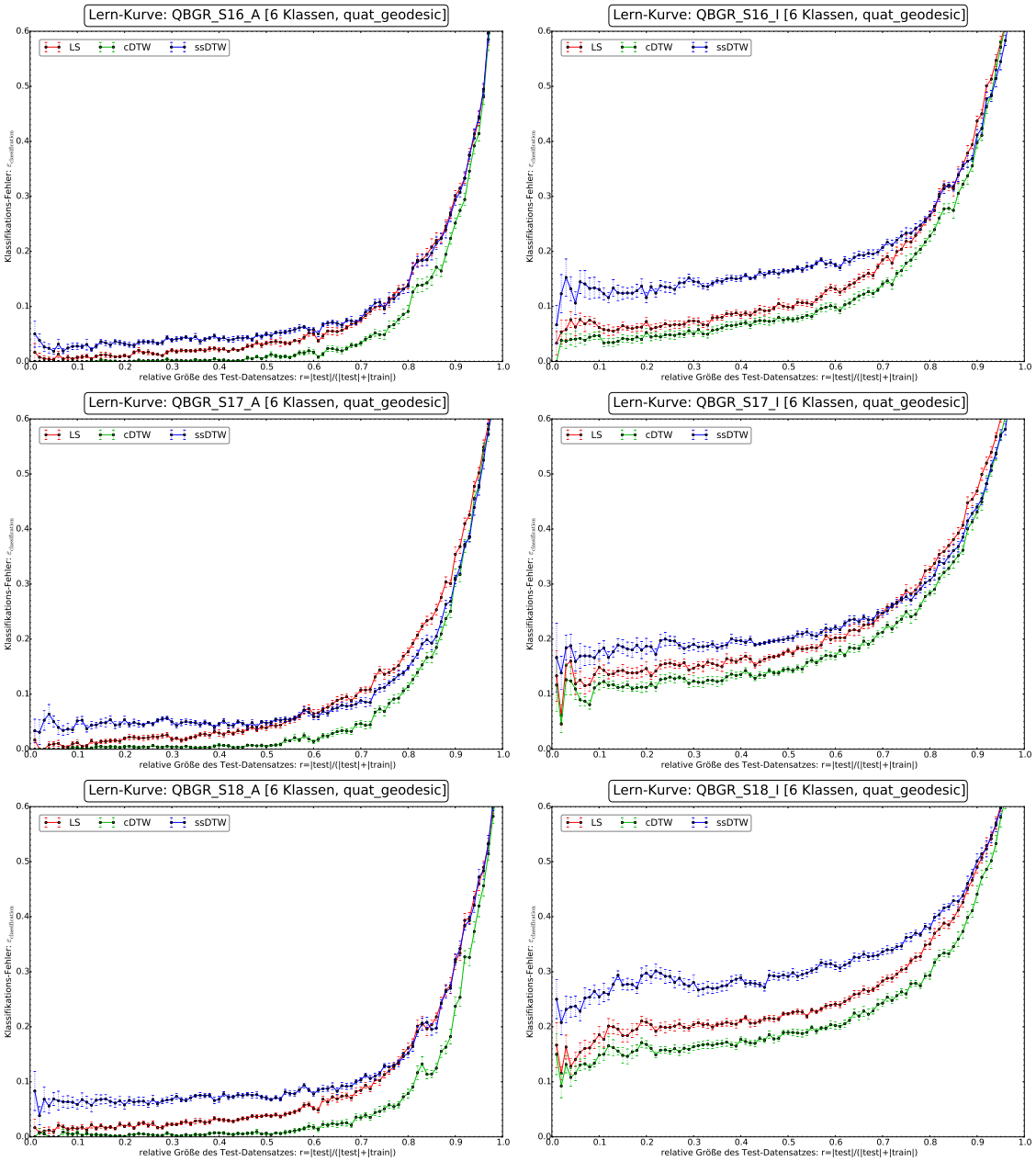


Abbildung 2.25: Lern-Kurve: QBGR-Datensatz (Fortsetzung) – Sensoren: **[linke Seite]** aktiver Arm: S16_A, S17_A, S18_A. **[rechte Seite]** inaktiver Arm: S16_I, S17_I, S18_I. Vergleich der **Zeitreihen**-Abstandsmaße: LS, cDTW, ssDTW. (*punktweises* Abstandsmaß: „quat_geodesic“).
Bei den Sensoren am inaktiven Arm ist – genau wie beim Opportunity-Datensatz – ssDTW das schlechteste Abstandsmaß. Bei den Sensoren am aktiven Arm liegen die Ergebnisse dichter beieinander. Teilweise (für $r_{\text{test}} > 0.5$) ist ssDTW dort sogar etwas besser als Lockstep. In allen Fällen ist jedoch cDTW das beste Abstandsmaß in Bezug auf den Klassifikations-Fehler.

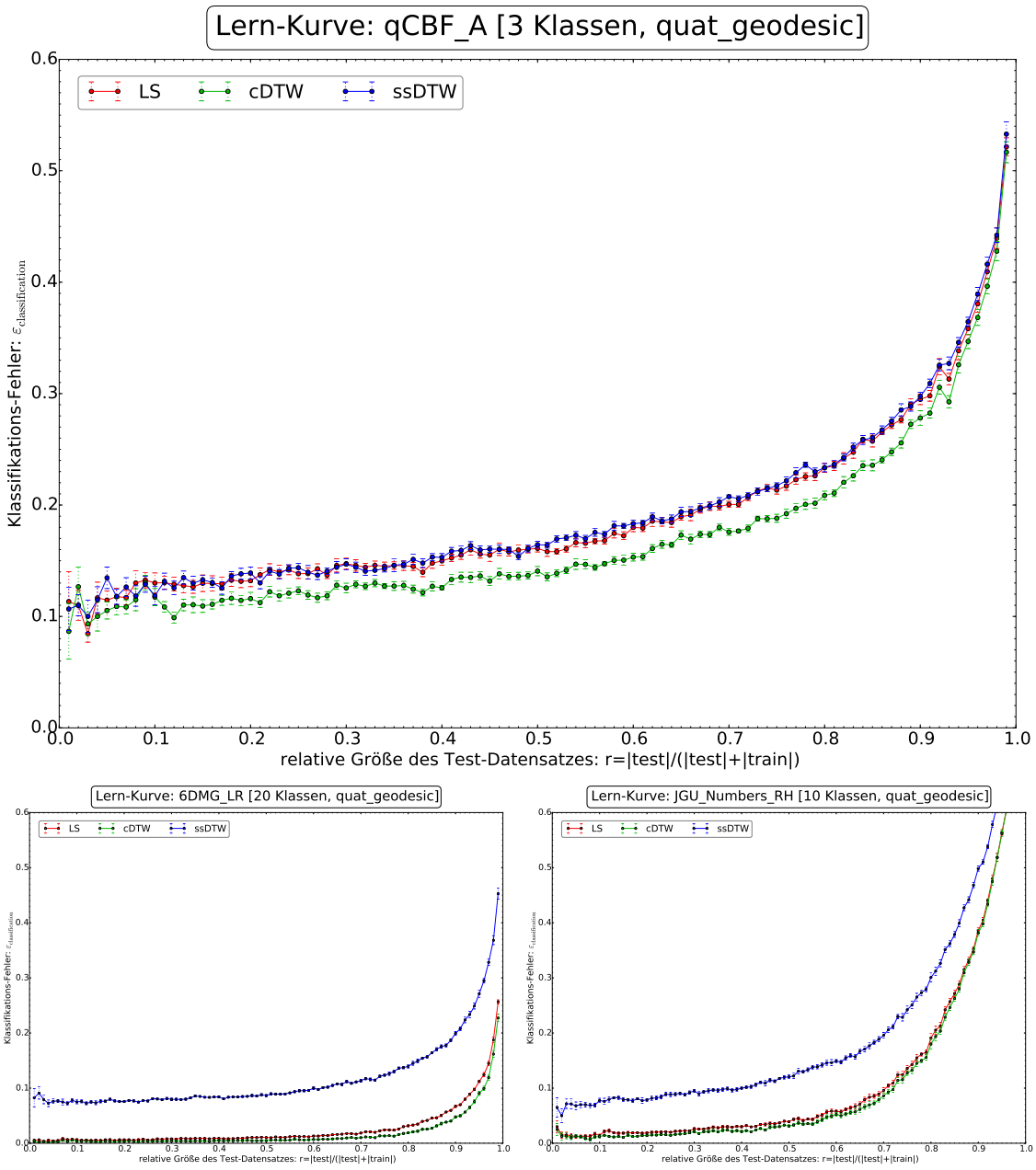


Abbildung 2.26: Lern-Kurve: Vergleich der **Zeitreihen**-Abstandsmaße: LS, cDTW, ssDTW (*punktweises* Abstandsmaß: „quat_geodesic“). **[oben]** qCBF-Datensatz („Sensor“: A), **[unten links]** 6DMG-Datensatz – Sensor: LR, **[unten rechts]** JGU_Numbers-Datensatz – Sensor: RH. Bei qCBF klassifizieren ssDTW und Lockstep etwa gleich, während cDTW etwas besser abschneidet. Bei 6DMG und JGU_Numbers ist ssDTW wieder mit Abstand das schlechteste Abstandsmaß, während cDTW jeweils etwas besser ist als Lockstep.

Ergebnisse der einzelnen Datensätze

Im Folgenden werden zunächst die Klassifikations-Ergebnisse für alle untersuchten Datensätze kurz diskutiert. Im Anschluss daran erfolgt dann ein direkter Vergleich der verschiedenen Zeitreihen-Abstandsmaße. Bei den hier vorgestellten Confusion-Matrizen (siehe folgender Abschnitt) wurde das *punktweise* Abstandsmaß auf „quat_geodesic“ festgelegt. Für die Aufspaltung der Daten in Test- und Trainings-Datensatz wurde dabei exemplarisch $r_{\text{test}} = 0.5$ gewählt, also gleiche Größe von Test- und Trainings-Datensatz.

Bei dem **Opportunity-Datensatz** (siehe Abb. 2.22 und 2.23) wird mit Lockstep und cDTW jeweils ein Klassifikations-Fehler zwischen 10% und 20% erreicht – für den Sensor LLA sogar noch höher – was zunächst einmal recht hoch erscheint. Dies lässt sich jedoch damit begründen, dass dieser Datensatz einige sehr schwer unterscheidbare Klassen enthält. Insbesondere die Klassen für das Öffnen und Schließen der drei Schubladen sind sehr ähnlich, da die Schubladen in der experimentellen Testumgebung direkt übereinander angebracht waren [10]. Da wir nur Rotations-Daten zur Verfügung haben und keine Absolut-Positionsdaten, sind die Greifbewegungen zu den unterschiedlich hoch angebrachten Schubladen nur sehr schwer zu unterscheiden. Bei der Klassifikation zählt es jedoch auch als Klassifikations-Fehler wenn z.B. „öffne Schublade 1“ mit „öffne Schublade 2“ verwechselt wird. Damit man besser erkennen kann, welche Klassen untereinander verwechselt werden, kann man eine sogenannte *Confusion-Matrix* erzeugen. Dort wird dargestellt, welcher Anteil der Anfragen für jedes Anfrage-Shape welchem Antwort-Shape (also welcher Klasse) zugeordnet wurde. In Abb. 2.27 ist die Confusion-Matrix exemplarisch für den Sensor RLA dargestellt. Man erkennt deutlich, dass die „Öffnen“-Shapes („Open ...“) – insbesondere der Schubladen 1,2 und 3 („Drawer 1,2,3“), des Kühlschranks („Fridge“) und der Spülmaschine („Dishwasher“) – untereinander häufig verwechselt wurden. Dasselbe gilt für die entsprechenden „Schließen“-Shapes („Close ...“). Zudem werden auch die Öffnen- und Schließen-Bewegungen gegenseitig zum Teil verwechselt. Dies ist darauf zurückzuführen, dass die entsprechenden Greifbewegungen eine gewisse zeitliche Symmetrie aufweisen und damit z.B. die Bewegung für das Öffnen einer Schublade und die entsprechende Bewegung für das Schließen der Schublade relativ ähnlich zueinander sind. Nehmen wir nun an, dass lediglich die drei Klassen „öffne Schublade 1“, „öffne Schublade 2“ und „öffne Schublade 3“ untereinander *nicht* unterscheidbar wären, während diese von den anderen Klassen unterscheidbar wären. Die anderen Klassen untereinander seien ebenfalls unterscheidbar. Dann würden die Shapes aus diesen drei Klassen durch „bloßes Raten“ gemäß Gleichung (2.5) zu zwei Dritteln falsch klassifiziert werden. Damit würden insgesamt $\frac{2}{3} \cdot \frac{3}{17} \approx 11.76\%$[(7)] der Shapes falsch zugeordnet werden. Die gleiche Überlegung gilt auch für die Klassen „schließe Schublade 1“, „schließe Schublade 2“, und „schließe Schublade 3“. Damit läge der Anteil am Gesamt-Klassifikations-Fehler, der allein durch diese 6 Klassen verursacht würde schon bei $\frac{2}{17} + \frac{2}{17} \approx 23.53\%$. Betrachten wir nun alle 6 Klassen (Öffnen- und Schließen-Klassen der

[(7)] Dies sind $\frac{2}{3}$ falsch zugeordnete Shapes der insgesamt $\frac{3}{17}$ aller Shapes aus den „öffne Schublade“-Klassen.

drei Schubladen 1,2 und 3) als untereinander *nicht* unterscheidbar, so würde der Klassifikations-Fehler mindestens $\frac{5}{6} \cdot \frac{6}{17} \approx 29.41\%$ betragen. Da der Klassifikations-Fehler für die meisten Sensoren jedoch deutlich unter 20% und damit auch deutlich unter dieser theoretischen Grenze liegt, ist das angewendete Verfahren offenbar schon in der Lage, auch diese sehr ähnlichen Klassen zumindest teilweise zu unterscheiden. Zum Vergleich: Für „bloßes Raten“ *aller* Klassen läge der mittlere Klassifikations-Fehler für die N=17 Klassen des Opportunity-Datensatzes nach Gleichung (2.5) bei $\varepsilon_{\text{classification,random}}(N = 17) = \frac{16}{17} \approx 94.12\%$. Zum Vergleich der verschiedenen Sensoren des Opportunity-Datensatzes sind in Abb. 2.32 (oben) die Lern-Kurven für die verschiedenen Sensoren – exemplarisch für das Zeitreihen-Abstandsmaß cDTW und das punktweise Abstandsmaß quat_geodesic – dargestellt. Die Klassifikations-Fehler der Sensoren am rechten Arm (RLA und RUA) liegen deutlich unter denen der Sensoren am linken Arm (LLA und LUA). Dies ist dadurch erklärbar, dass alle Probanden die Bewegungen – wie etwa das Öffnen und Schließen des Kühlschranks – mit dem rechten Arm durchgeführt haben. Eine Klassifikation der Bewegungen mit Hilfe der Sensoren am inaktiven linken Arm ist dennoch möglich, wenn auch mit schlechterem Ergebnis. Interessanterweise ist der Klassifikations-Fehler bei dem Sensor am Rücken (BACK) sogar noch etwas geringer als bei den Sensoren am aktiven (rechten) Arm. Offenbar ist bei den hier verglichenen Bewegungen auch der Oberkörper maßgeblich beteiligt, sodass eine Unterscheidung der Bewegungen anhand des dort angebrachten Sensors sogar besser funktioniert als bei den Sensoren am Arm.

Beim **QBGR-Datensatz** (siehe Abb. 2.24 und 2.25) ist der Klassifikations-Fehler bei den Sensoren am aktiven (bewegten) Arm (S15_A, S16_A, S17_A, S18_A) erwartungsgemäß deutlich geringer als bei den Sensoren am inaktiven (still gehaltenen) Arm (S15_I, S16_I, S17_I, S18_I). Allerdings ist eine Klassifikation anhand der Sensoren am inaktiven Arm dennoch – wenn auch mit entsprechend schlechterem Ergebnis – möglich. Für die $N = 6$ Klassen dieses Datensatzes würde der (mittlere) Klassifikations-Fehler bei „bloßem Raten“ nach Gleichung (2.5) $\varepsilon_{\text{classification,random}}(N = 6) \approx 83.33\%$ betragen. Die für die Sensoren am inaktiven Arm bestimmten Klassifikations-Fehler liegen bei cDTW jedoch bei lediglich $5 - 20\%$. Eine Unterscheidung der Bewegungen anhand der Sensoren am inaktiven Arm ist also teilweise möglich. Bei den Sensoren am aktiven Arm werden sogar Klassifikations-Fehler im Bereich von $0 - 5\%$ erreicht. Für cDTW und $r_{\text{test}} \leq 0.5$ ist die Klassifikation auf diesen Teildatensätzen nahezu fehlerfrei. Der Klassifikations-Fehler für den Sensor am unteren Bauch (S19) liegt nur wenig oberhalb des Klassifikations-Fehlers für die Sensoren am aktiven Arm. Offenbar können die Bewegungen auch mit diesem Sensor noch relativ gut unterschieden werden. Zum direkten Vergleich der verschiedenen Sensoren des QBGR-Datensatzes sind in Abb. 2.32 (unten) ebenfalls die Lern-Kurven für die verschiedenen Sensoren – auch hier wieder exemplarisch für das Zeitreihen-Abstandsmaß cDTW und das punktweise Abstandsmaß quat_geodesic – dargestellt. Abb. 2.28 zeigt die Confusion-Matrix für den Sensor S15_A. Man erkennt, dass die Klassifikation bei diesem Datensatz sehr gut funktioniert. Lediglich die Klassen „Lift“ und „Block“ werden unter dem Lockstep-Abstandsmaß teilwei-

se mit „Uppercut“ verwechselt, sowie seltener auch „Jab“ mit „Throw“. Unter den anderen beiden Abstandsmaßen cDTW und ssDTW ist die Klassifikation nahezu fehlerfrei. Der Teil-Datensatz QBGR S15_A ist der einzige untersuchte Datensatz, bei dem das ssDTW-Abstandsmaß für jeden Wert von r_{test} bessere Klassifikations-Ergebnisse liefert als das Lockstep Abstandsmaß. In allen Fällen ist jedoch das cDTW-Abstandsmaß am besten für die Klassifikation geeignet.

Die Klassifikation des **6DMG-Datensatzes** (siehe Abb. 2.26, unten links) für Lockstep und cDTW ist nahezu fehlerfrei. Erst ab $r_{\text{test}} \approx 0.7$ (cDTW), bzw. $r_{\text{test}} \approx 0.5$ (Lockstep) beginnt der Klassifikations-Fehler über die 1%-Marke zu steigen. Selbst für $r_{\text{test}} = 0.99$ – bei dem also nur 1% der gesamten Daten als Trainings-Datensatz verwendet wird! – liegt für diesen Datensatz der Klassifikations-Fehler unter Lockstep und cDTW bei nur ca. 20%. Auch hier sei noch einmal der Vergleich angeführt: „Bloßes Raten“ bei 20 Klassen würde einem Klassifikations-Fehler von 95% entsprechen. Lediglich das ssDTW-Abstandsmaß liegt mit ca. 8% mittlerem Klassifikationsfehler wieder deutlich über den anderen beiden Zeitreihen-Abstandsmaßen. Obwohl dieser Datensatz mit 20 Gesten die meisten Klassen aufweist, sind die Klassifikations-Ergebnisse hier am besten. Trotz der Tatsache, dass die Klassen zum Teil sehr ähnlich sind (wie z.B. „wischen nach oben“ („swipe up“) und etwa „wischen diagonal nach rechts oben“ („swipe upright“) sowie „wischen diagonal nach links oben“ („swipe upleft“)), können sie mittels 1NN-Klassifikation sehr gut voneinander unterschieden werden. Während die Bewegungen z.B. beim Opportunity-Datensatz teilweise recht komplex sind (z.B. „reinige Tisch“ oder „trinke aus Tasse“) und damit auch mehr Spielraum für individuelle Durchführung lassen, sind die Bewegungen beim 6DMG-Datensatz deutlich einfacher und zudem auch sehr klar definiert. Dadurch ist eine Unterscheidung der einzelnen Bewegungen bei der Klassifizierung deutlich besser möglich. In Abb. 2.29 ist die Confusion-Matrix für diesen Datensatz dargestellt: Die Klassifikation unter cDTW ist fast fehlerfrei. Lediglich unter Lockstep und ssDTW werden Klassifikations-Fehler von mehr als 2% erreicht. Wenn es zur Verwechslung zwischen zwei oder mehreren Klassen kommt, so sind es meistens sehr ähnliche Klassen wie z.B. „swipe left“, „swipe upleft“ und „swipe downleft“ oder „swipe right“, „swipe upright“ und „swipe downright“. Ebenfalls verwechselt werden „poke left / right“ mit „swipe left / right“. Solche Verwechslungen kommen jedoch insgesamt recht selten vor, obwohl all diese Klassen sehr ähnlich zueinander sind.

Bei dem von uns selbst erzeugten **JGU_Numbers-Datensatz** (siehe Abb. 2.26, unten rechts) ist das Klassifikations-Ergebnis zwar etwas schlechter aber dennoch vergleichbar mit dem des 6DMG-Datensatzes. Für $r_{\text{test}} \leq 0.5$ bleibt der mittlere Klassifikations-Fehler für cDTW unterhalb von 3%. Für das Lockstep-Abstandsmaß gilt dies für Werte von $r_{\text{test}} \leq 0.43$. Auch hier liefert das ssDTW-Abstandsmaß wieder deutlich schlechtere Ergebnisse als die anderen beiden Abstandsmaße. Trotz der Tatsache, dass für die Klassifikation wieder ausschließlich die Rotations-Informationen verwendet wurden, funktioniert die Klassifikation auf diesem Datensatz also recht gut. Auch hier wieder zum Vergleich: „Bloßes Raten“ würde

bei 10 Klassen 90% Klassifikations-Fehler bedeuten. Dieser Wert wird selbst für $r_{\text{test}} = 0.99$ nicht erreicht: $\varepsilon_{\text{classification}}(r_{\text{test}} = 0.99) \approx 80\%$ für cDTW und Lockstep bzw. $\varepsilon_{\text{classification}}(r_{\text{test}} = 0.99) \approx 85\%$ für ssDTW. Abb. 2.30 zeigt die Confusion-Matrix für diesen Datensatz. Man erkennt deutlich, dass das ssDTW Abstandsmaß hier wesentlich schlechter abschneidet als die anderen beiden. Dies lässt sich damit erklären, dass beim Schreiben der Ziffern gewisse Teilbewegungen bei verschiedenen Ziffern vorkommen. Beispielsweise kommt eine kreisförmige Bewegung bei den Ziffern „0“, „6“, „8“ und „9“ vor. Während diese auch unter cDTW und Lockstep manchmal verwechselt werden, kommt es unter ssDTW auch z.B. zwischen den Ziffern „1“ und „2“, sowie „2“ und „3“ häufiger zu Verwechslungen. Offenbar werden hier wieder manche Bewegungen als Teil-Bewegungen einer anderen Klasse fehlinterpretiert. Hier zeigt sich wieder, dass die *globale* Alignierungs-Eigenschaft von cDTW und Lockstep für diese Aufgabe besser geeignet ist als die *lokale* Alignierung unter ssDTW.

Die Lern-Kurven für den synthetisch erzeugten **qCBF-Datensatz** sind in Abb. 2.26 (oben) dargestellt. Hier liefern ssDTW und Lockstep etwa gleiche Ergebnisse, während cDTW wieder etwas besser ist. Insgesamt sind die Klassifikations-Ergebnisse für diesen Datensatz relativ schlecht. Der Klassifikations-Fehler liegt für alle untersuchten Zeitreihen-Abstandsmaße über 10%. Dies ist zwar immer noch deutlich geringer als der Klassifikations-Fehler für „bloßes Raten“ – welcher bei drei Klassen bei einem Wert von $\varepsilon_{\text{classification,random}}(N = 3) \approx 66.67\%$ liegen würde – aber dennoch immer noch relativ hoch. In der Confusion-Matrix (siehe Abb. 2.31) erkennt man, dass es relativ häufig zu Verwechslungen zwischen allen drei Klassen kommt. Dies liegt in erster Linie daran, dass jedes Shape dieses Datensatzes in ein zufällig gewähltes Koordinatensystem transformiert wurde. Damit lassen sich die einzelnen Klassen nur schwer unterscheiden. Wir werden allerdings später noch feststellen, dass dieser Datensatz gerade deshalb sehr von der Klassifikation anhand der *inkrementellen Rotationen* profitiert, die dieses Problem durch eine vom gewählten Bezugssystem unabhängige Darstellung löst.

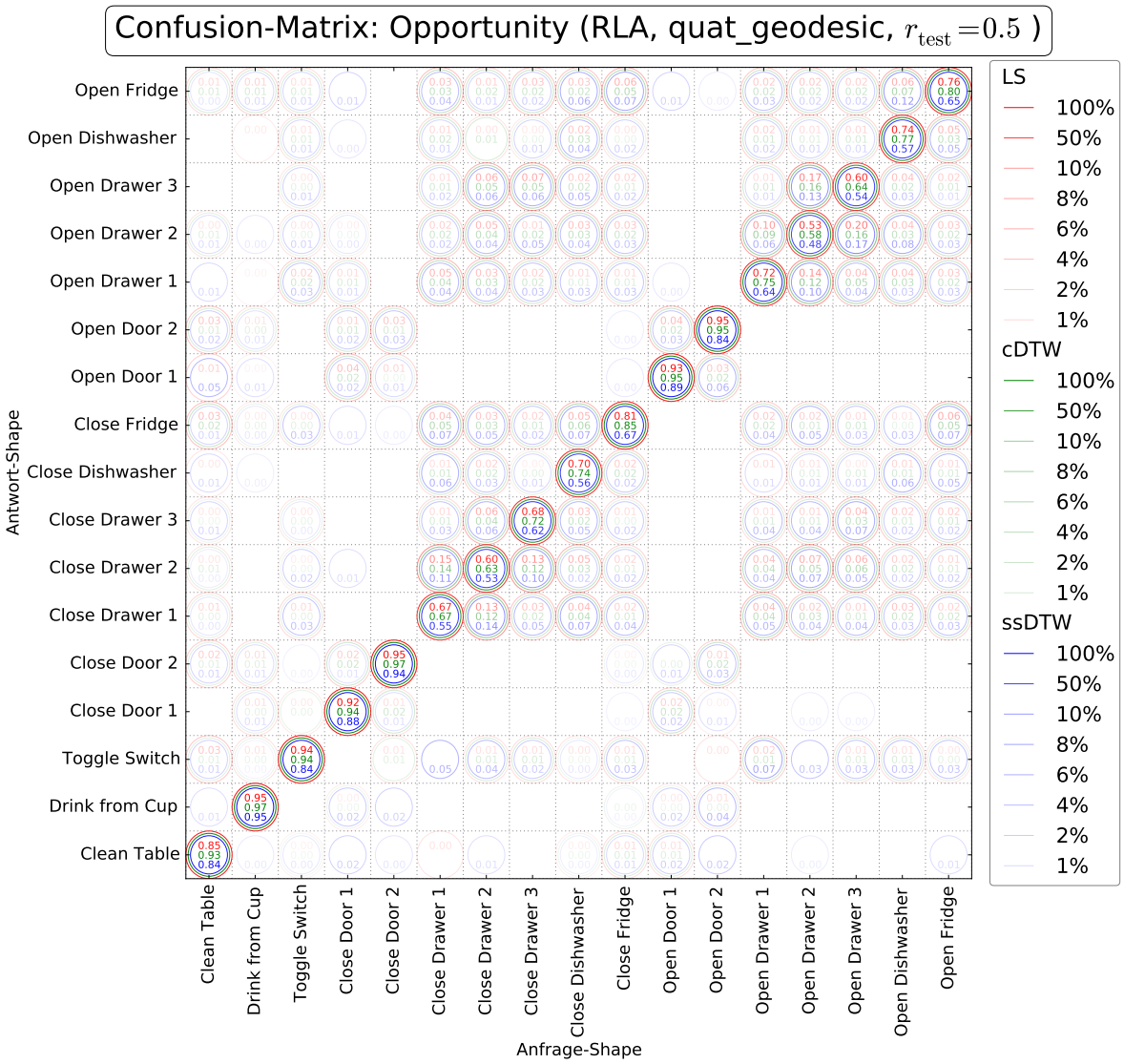


Abbildung 2.27: Confusion-Matrix: Opportunity-Datensatz: (Sensor RLA, punktweises Abstandsmaß: quat_geodesic, r_{test} = 0.5). Die dargestellten Zahlen repräsentieren den Anteil der Anfrage-Shapes (Spalten), der einer bestimmten Klasse (Antwort-Shape (Zeilen)) zugeordnet wurde. Zur besseren Verdeutlichung der Ergebnisse sind die Anteile mit dem Alpha-Kanal kodiert: 100% entspricht komplett opaque dargestellten Kreisen und Zahlen, 0% entspricht völlig transparent dargestellten Kreisen und Zahlen.
Man erkennt deutlich, dass die „Öffnen“- („Open ...“) und „Schließen“- („Close ...“) Shapes untereinander sehr oft verwechselt werden. Besonders stark betroffen sind hierbei die Shapes der drei Schubladen („Drawer 1,2,3“), der Spülmaschine („Dishwasher“) und des Kühlschranks („Fridge“). Dadurch, dass die Greifbewegungen zum Öffnen und Schließen eine gewisse zeitliche Symmetrie aufweisen, kommt es auch zu Verwechslungen zwischen Öffnen- und Schließen-Bewegungen.

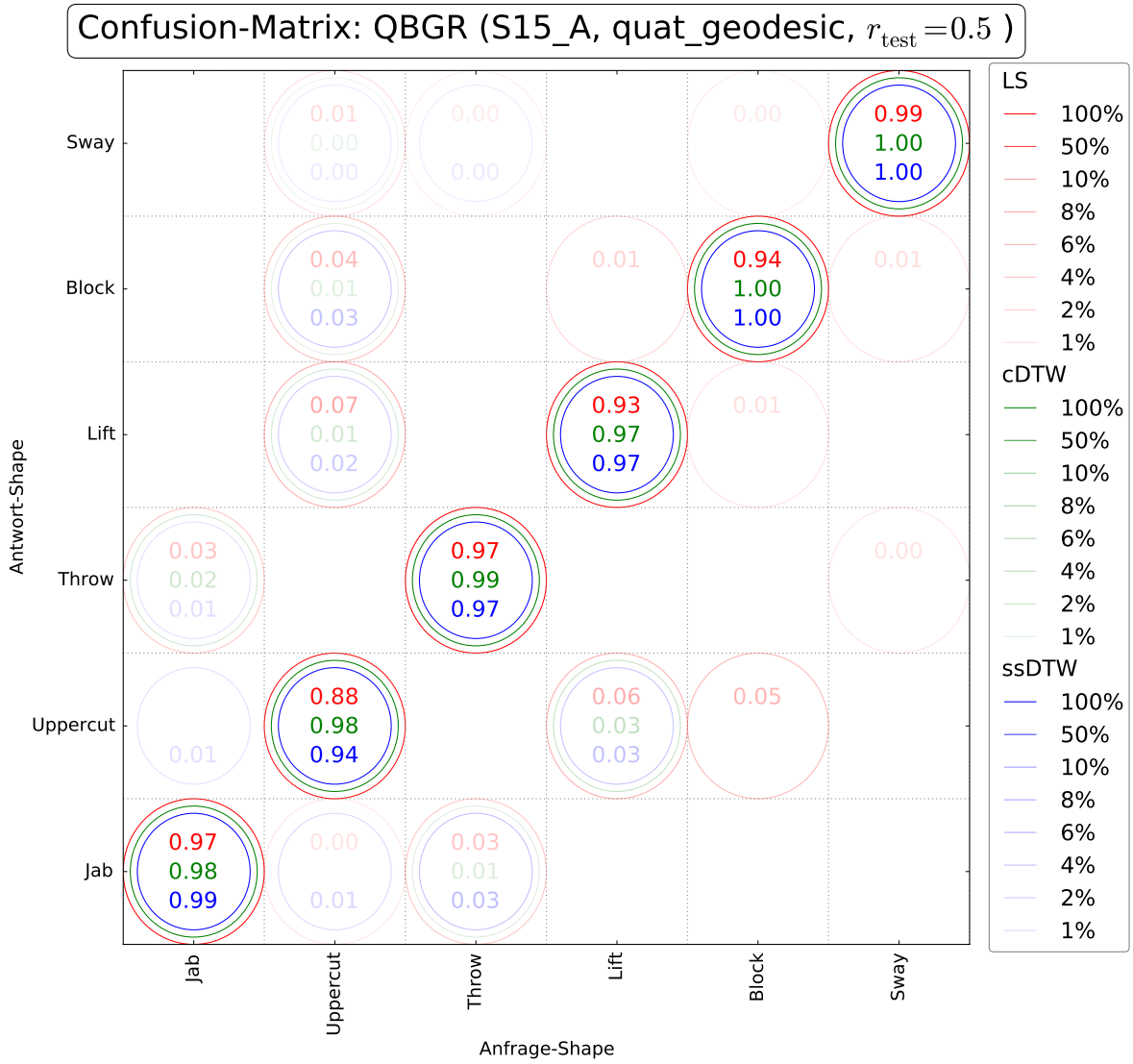


Abbildung 2.28: Confusion-Matrix: QBGR-Datensatz: (Sensor S15_A, punktweises Abstandsmaß: quat_geodesic, r_{test} = 0.5). Man erkennt, dass relativ wenige Klassen untereinander verwechselt werden. Lediglich die Klassen „Lift“ und „Block“ werden unter dem Lockstep-Abstandsmaß zum Teil mit „Uppercut“ verwechselt. Etwas seltener wird „Jab“ mit „Throw“ verwechselt. QBGR S15_A ist der einzige Datensatz, bei dem ssDTW stets bessere Klassifikations-Ergebnisse liefert als Lockstep.

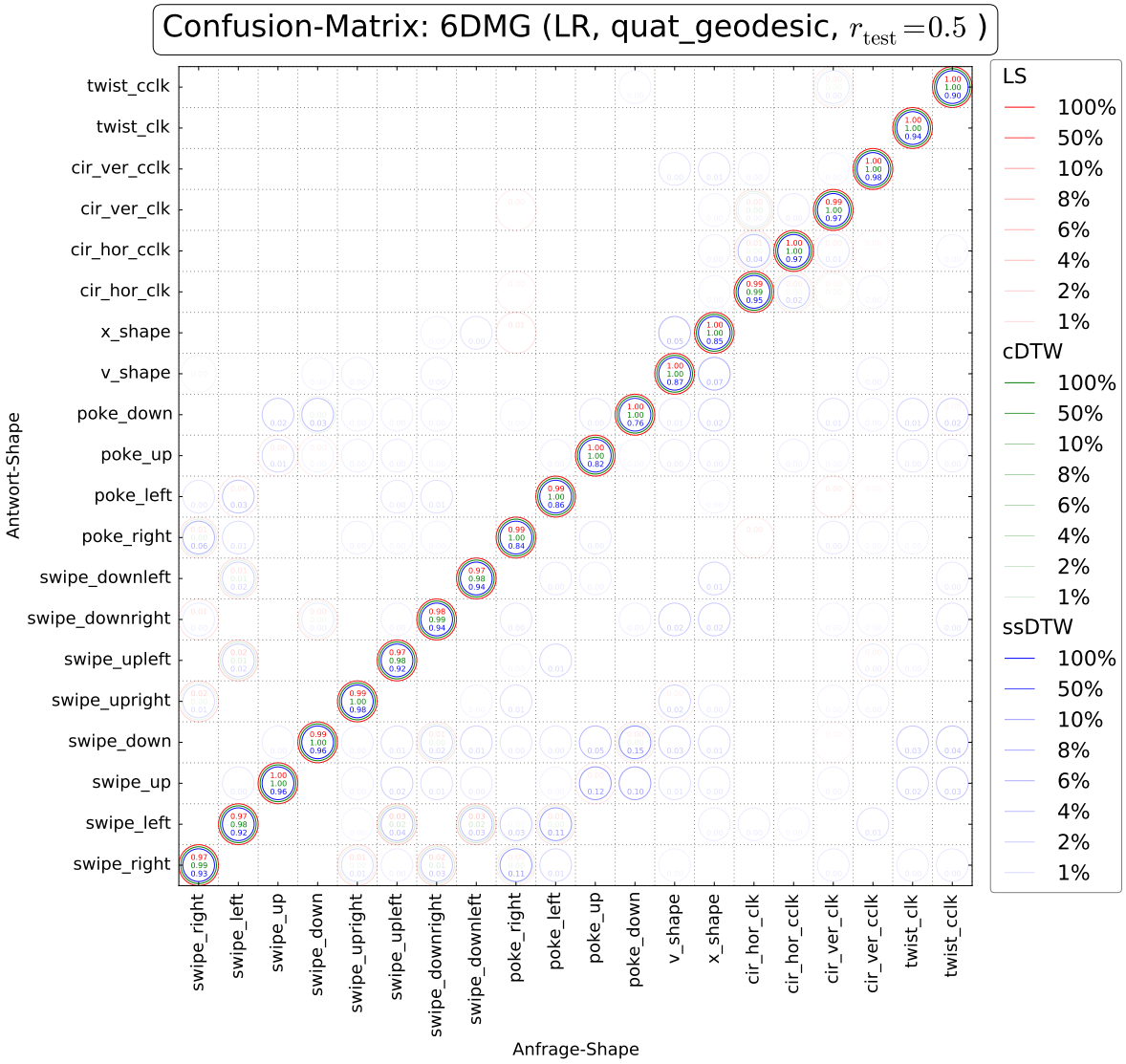


Abbildung 2.29: Confusion-Matrix: 6DMG-Datensatz: (Sensor LR, punktweises Abstandsmaß: quat_geodesic, r_{test} = 0.5). Obwohl dieser Datensatz von allen untersuchten mit 20 Klassen die meisten Klassen enthält, sind hier die Klassifikations-Ergebnisse am besten. Unter cDTW gibt es kaum falsch zugeordnete Shapes. Lediglich unter ssDTW und Lockstep gibt es Klassifikationsfehler von mehr als 2%. Wenn Klassen untereinander verwechselt werden, sind es meistens sehr ähnliche Klassen wie z.B. „swipe left“ mit „swipe upleft“ und „swipe downleft“, „swipe right“ mit „swipe upright“ und „swipe downright“ oder „poke left / right“ mit „swipe left / right“. Obwohl diese Klassen sehr ähnlich zueinander sind, kommen solche Verwechslungen jedoch relativ selten vor.

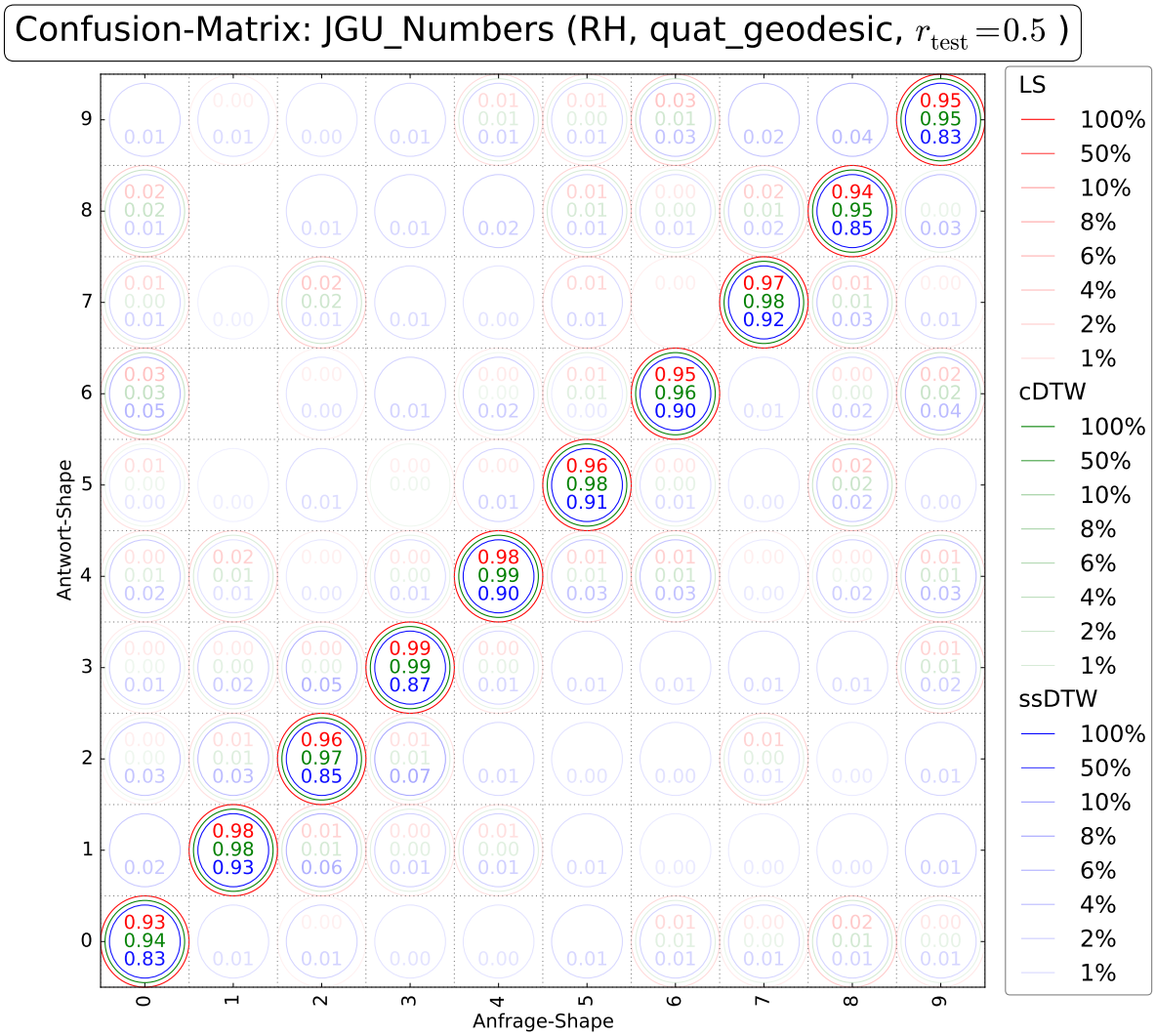


Abbildung 2.30: Confusion-Matrix: JGU_Numbers-Datensatz: (Sensor RH, punktweises Abstandsmaß: quat_geodesic, $r_{\text{test}} = 0.5$). Auch hier werden sehr gute Klassifikations-Ergebnisse erreicht. Es zeigt sich deutlich, dass cDTW und Lockstep auch hier wieder dem ssDTW-Abstandsmaß überlegen sind. Verwechslungen treten bevorzugt zwischen den Ziffern auf, die kreisförmige Elemente enthalten: „0“, „6“, „8“ und „9“. Unter ssDTW werden jedoch auch z.B. „1“ mit „2“, sowie „2“ mit „3“ häufiger verwechselt.

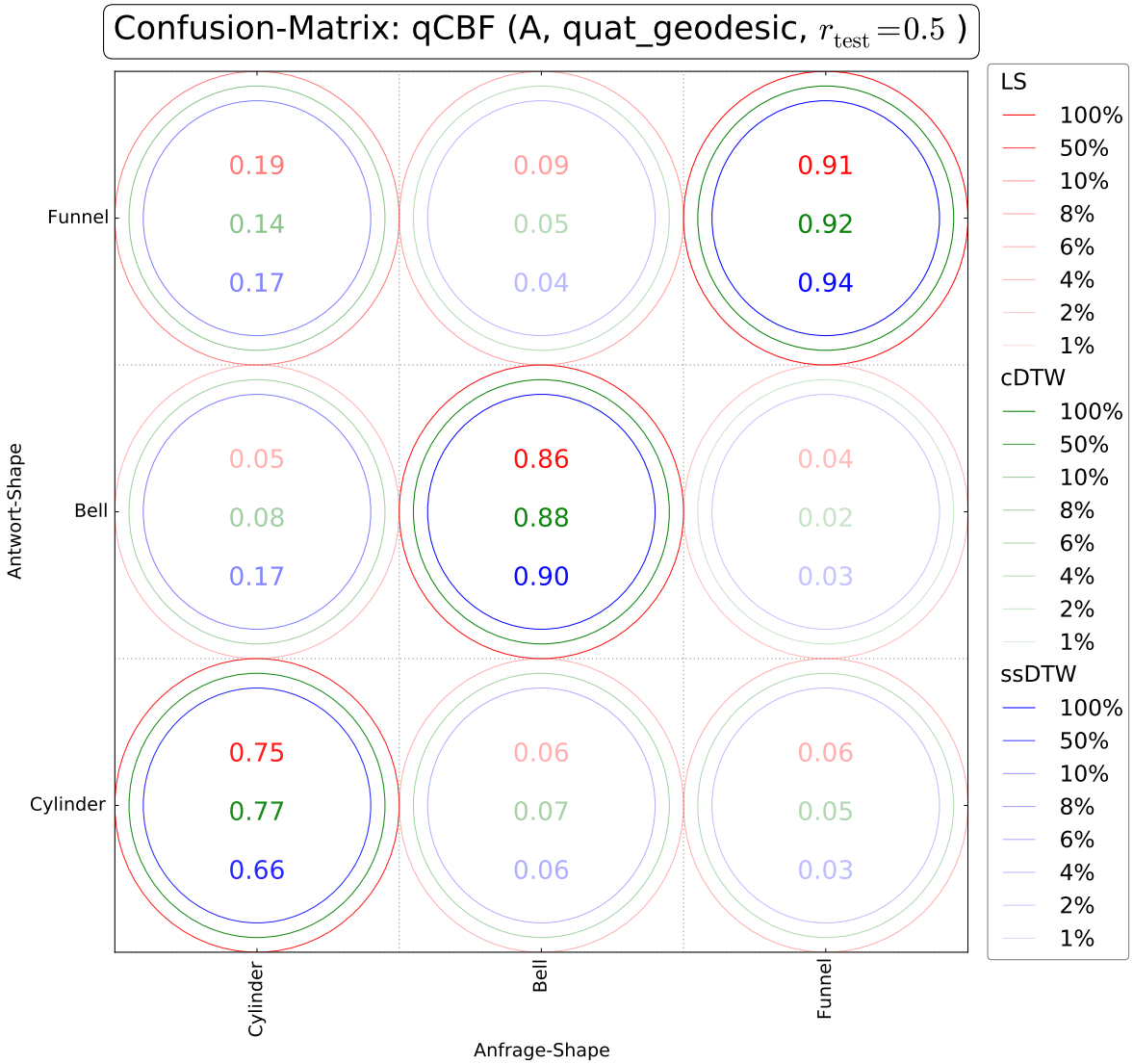


Abbildung 2.31: Confusion-Matrix: qCBF-Datensatz: („Sensor A“, punktweises Abstandsmaß: quat_geodesic, $r_{\text{test}} = 0.5$). Die Klassifikation auf den originalen (absoluten) Zeitreihen funktioniert relativ schlecht. Es kommt bei allen drei Klassen häufig zu Verwechslungen. Wie wir später sehen werden, profitiert dieser Datensatz jedoch sehr von der Verwendung der *inkrementellen Rotationen* (siehe weiter unten).

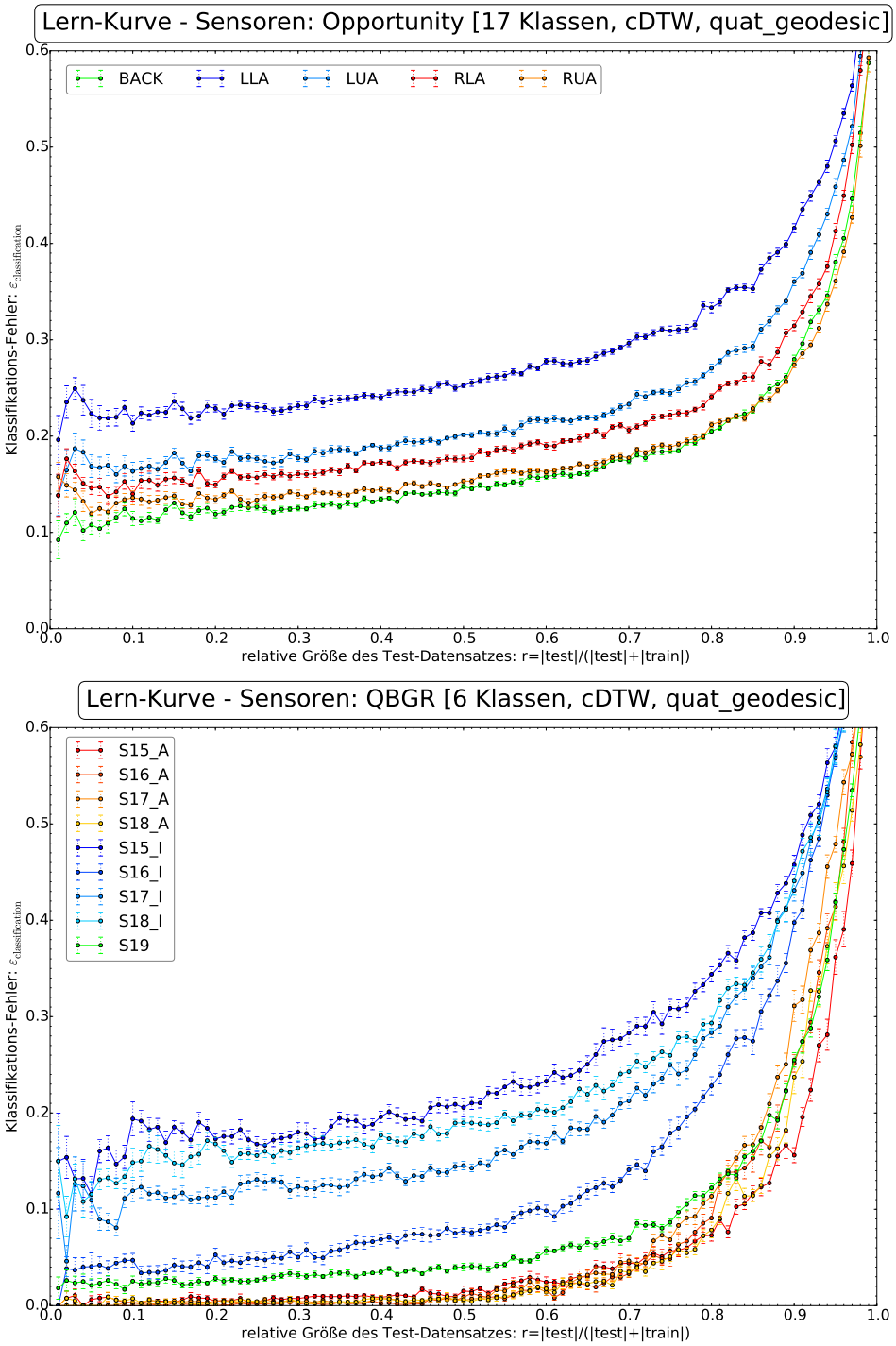


Abbildung 2.32: Lern-Kurve: Vergleich der Sensoren. Dargestellt sind die Klassifikations-Fehler für das Zeitreihen-Abstandsmaß cDTW beim punktweisen Abstandsmaß quat_geodesic. **[Oben]**: Opportunity-Datensatz: Die Sensoren am (aktiven) rechten Arm (RLA, RUA) sowie der Sensor am Rücken (BACK) liefern bessere Ergebnisse als die Sensoren am (inaktiven) linken Arm (LLA, LUA). Hierbei liefert der Sensor am Rücken (BACK) sogar die besten Klassifikations-Ergebnisse. **[Unten]**: QBGR-Datensatz: Die Sensoren am aktiven Arm (Sxx_A) liefern erwartungsgemäß deutlich bessere Ergebnisse als die Sensoren am inaktiven Arm (Sxx_I). Der Sensor am unteren Bauch (S19) liegt bezüglich der Klassifikations-Ergebnisse dazwischen.

Zusammenfassung: Vergleich der Zeitreihen-Abstandsmaße

In Abb. 2.33 und 2.34 werden die Zeitreihen-Abstandsmaße direkt miteinander verglichen. Dazu wird für jeden (Teil-)Datensatz die Differenz der jeweiligen Klassifikations-Fehler bei je zwei verschiedenen Zeitreihen-Abstandsmaßen wie folgt berechnet:

$$\Delta\varepsilon_{\text{dist\_c-dist\_r,r}} := \varepsilon_{\text{dist\_c,r}} - \varepsilon_{\text{dist\_r,r}} \tag{2.6}$$

Hierbei ist $\varepsilon_{\text{dist\_r,r}}$ der (über 100 stratifizierte Aufteilungen des Datensatzes gemittelte) Klassifikations-Fehler für das Referenz-Abstandsmaß *dist_r* für $r_{\text{test}} = r$. Analog dazu ist $\varepsilon_{\text{dist\_c,r}}$ entsprechend der Klassifikations-Fehler für das Vergleichs-Abstandsmaß *dist_c* für den gleichen Wert $r_{\text{test}} = r$. Dabei werden nun die verschiedenen Zeitreihen-Abstandsmaße (Lockstep, cDTW und ssDTW) miteinander verglichen. Als Referenz-Maß (*dist_r*) wird jeweils cDTW verwendet und jeweils die Fehlerdifferenz zu diesem Abstandsmaß berechnet. Werte von $\Delta\varepsilon_{\text{dist\_c-dist\_r,r}} > 0$ bedeuten, dass der Klassifikations-Fehler des Referenz-Abstandsmaßes kleiner ist, als der des Vergleichs-Abstandsmaßes und damit das Referenz-Abstandsmaß unter diesen Bedingungen **besser** klassifiziert. Werte $\Delta\varepsilon_{\text{dist\_c-dist\_r,r}} < 0$ bedeuten entsprechend, dass das Referenz-Abstandsmaß **schlechter** klassifiziert als das Vergleichs-Abstandsmaß. Dargestellt sind die Werte von $\Delta\varepsilon_{\text{dist\_c-dist\_r,r}}$ für jeweils drei verschiedene Werte von r_{test}: Einmal $r_{\text{test}} = 0.1$, für den der Trainings-Datensatz relativ groß ist, dann $r_{\text{test}} = 0.5$, bei dem Test- und Trainings-Datensatz gleich groß sind und schließlich noch $r_{\text{test}} = 0.9$, bei dem der Trainings-Datensatz relativ klein ist.

Es zeigt sich hierbei – deutlich und für alle Datensätze konsistent – dass die beiden Zeitreihen-Abstandsmaße cDTW und Lockstep sehr ähnliche Klassifikations-Fehler liefern. Dabei ist cDTW immer etwas besser als Lockstep. Der Unterschied zwischen diesen beiden Abstandsmaßen ist relativ klein. Es zeigte sich, dass der auf den Trainings-Daten gelernte Wert für die Warping-Fenster-Breite für cDTW in den meisten Fällen relativ klein ist (meistens $W \leq 5\%$). Dies bedeutet, dass die Warping-Pfade für cDTW immer recht nah an der Diagonalen der Penalty-Matrix verlaufen und damit dem festen Pfad des Lockstep-Abstandes (welcher exakt auf der Diagonalen verläuft) sehr ähnlich sind. Deshalb sind auch die Klassifikations-Ergebnisse der beiden Zeitreihen-Abstandsmaße sehr ähnlich, da die Abstandswerte zweier Zeitreihen unter cDTW mit kleinem Warping-Fenster und Lockstep sehr ähnlich sind.

Das ssDTW-Abstandsmaß ist in (fast) allen Fällen deutlich schlechter als die anderen beiden Abstandsmaße. Dies ist sehr wahrscheinlich darauf zurückzuführen, dass bei Lockstep und cDTW alle Zeitreihen auf dieselbe Länge skaliert werden, während bei ssDTW jeweils die kleinere als Teilsequenz in der größeren gesucht wird. Während bei Lockstep und cDTW jeder Punkt der beiden zu vergleichenden Zeitreihen beachtet werden muss und Anfang und Ende der beiden Zeitreihen jeweils immer aufeinander abgebildet werden (*globale* Alignierung), kann ssDTW Teile der größeren Zeitreihe komplett abschneiden und nur einen

Teil der größeren Sequenz verwenden (*lokale* Alignierung). Die Länge der abgeschnittenen bzw. beibehaltenen Teile der größeren Zeitreihe werden bei der Bewertung des ssDTW-Abstandsmaßes nicht berücksichtigt. Es kann also vorkommen, dass eine kleine Bewegung wie z.B. „betätige Lichtschalter“ als Teil einer größeren Bewegung wie z.B. „reinige Tisch“ erkannt wird und dann fälschlicherweise dieser Klasse zugeordnet wird. Da Lockstep und cDTW nur *globale* Alignierungen erzeugen, kann dieser Fall dort nicht auftreten.

Nachdem wir nun verschiedene *Zeitreihen*-Abstandsmaße miteinander verglichen haben, wollen wir im nächsten Abschnitt verschiedene *punktweise* Abstandsmaße miteinander vergleichen.

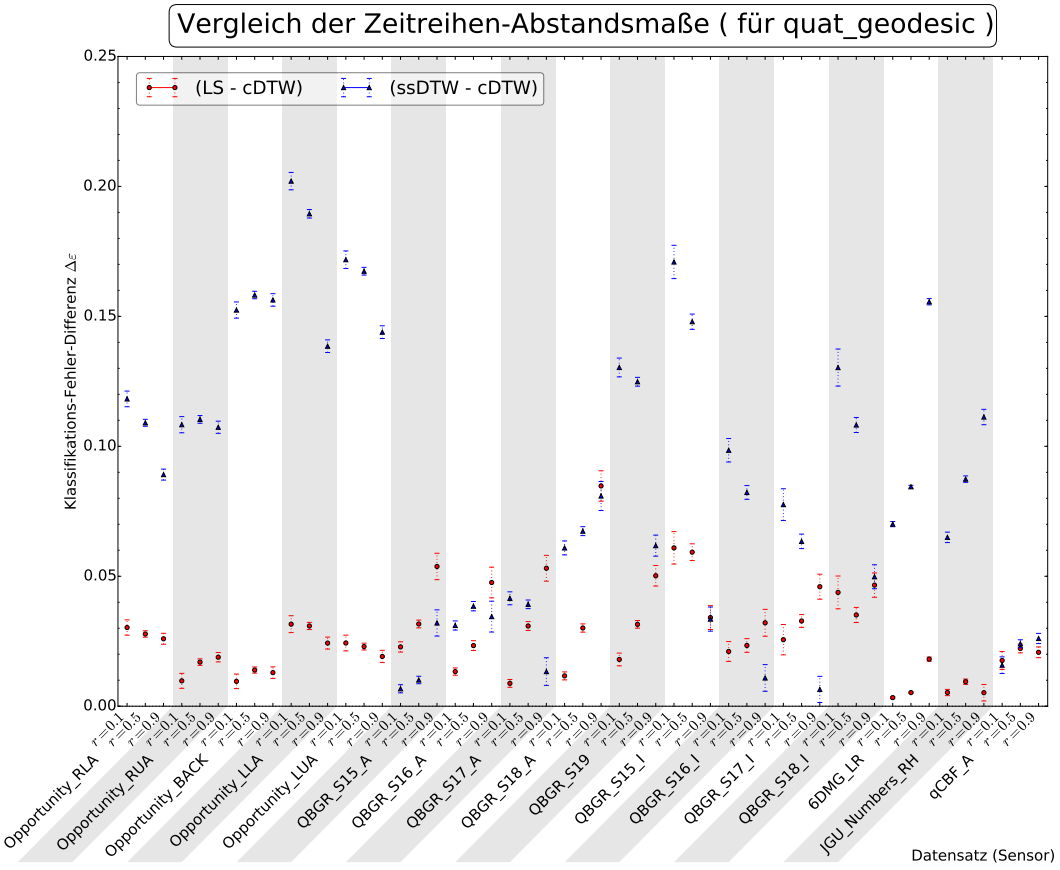


Abbildung 2.33: Vergleich der **Zeitreihen**-Abstandsmaße bei jeweils festem *punktweisen* Abstandsmaß. Dargestellt sind jeweils Mittelwert und Standard-Fehler der Klassifikations-Fehler-Differenzen über 100 stratifizierte Aufteilungen des jeweiligen Datensatzes für $r_{\text{test}} \in \{0.1, 0.5, 0.9\}$. Man erkennt deutlich, dass ssDTW für (fast) alle Datensätze und Sensoren deutlich schlechtere Klassifikations-Ergebnisse liefert als das Referenz-Abstandsmaß cDTW. Das Lockstep-Abstandsmaß liefert jeweils etwas schlechtere Ergebnisse als cDTW. Hier wurde als *punktweises* Abstandsmaß quat_geodesic festgehalten.

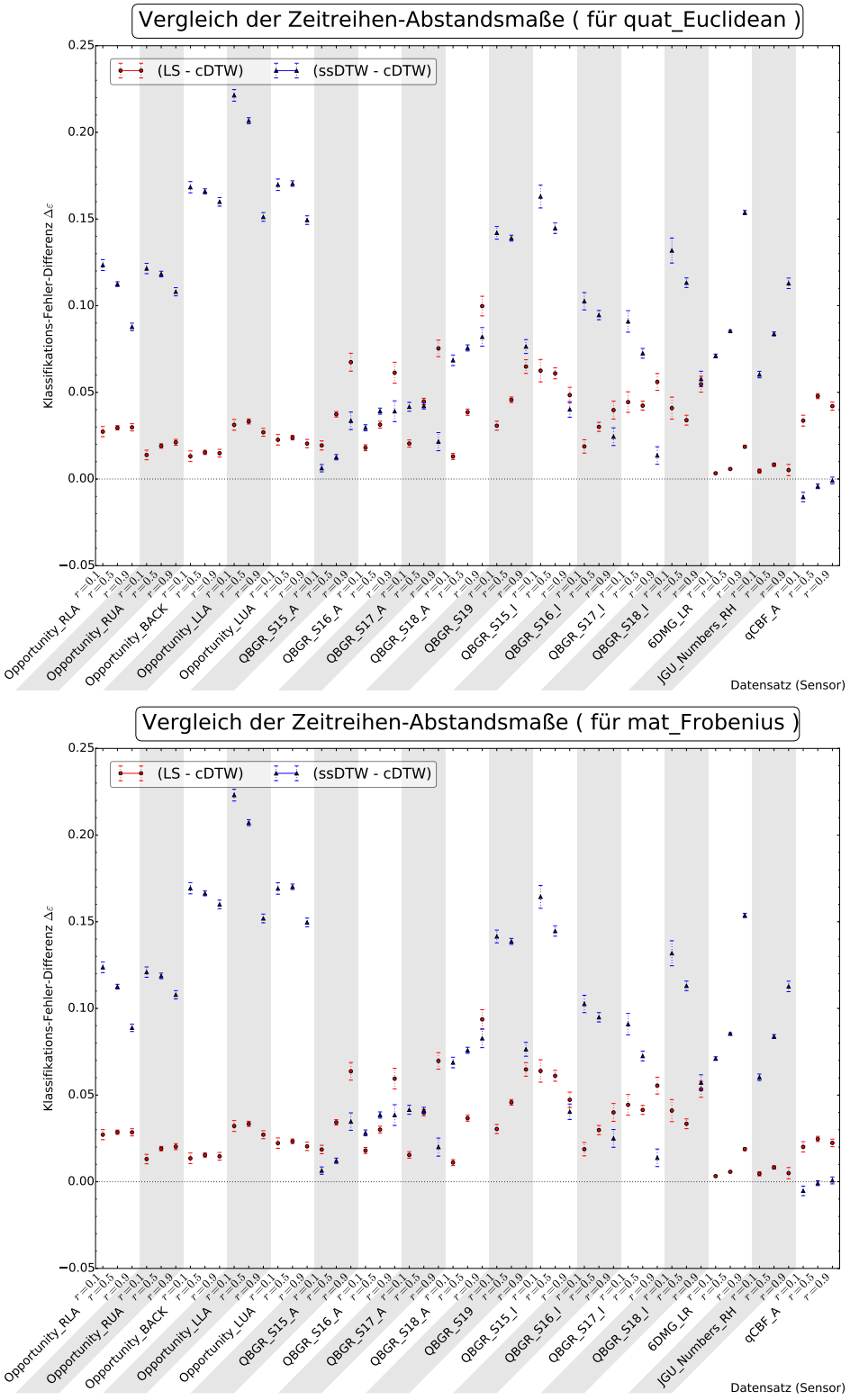


Abbildung 2.34: Vergleich der **Zeitreihen**-Abstandsmaße bei jeweils festem *punktweisen* Abstandsmaß (Fortsetzung). Für die anderen beiden *punktweisen* Abstandsmaße quat_Euclidean **[oben]** und mat_Frobenius **[unten]** ergeben sich im Wesentlichen die gleichen Ergebnisse beim Vergleich der **Zeitreihen**-Abstandsmaße: ssDTW ist in fast allen Fällen deutlich schlechter und Lockstep ist etwas schlechter als cDTW.

2.5.2 Klassifikation: Vergleich der punktweisen Abstandsmaße

Für jedes *Zeitreihen*-Abstandsmaß muss ebenfalls ein *punktweises* Abstandsmaß zwischen je zwei Punkten der Zeitreihen gewählt werden. Wir untersuchen und vergleichen hierbei die folgenden drei **punktweisen** Abstandsmaße

- geodätischer Abstand auf Rotations-Quaternionen $a, b \in \mathbb{H}_1$:
  $\text{dist}(a,b) = \arccos(|\langle a|b\rangle|) \in \mathbb{R}_0^+$ („quat_geodesic“)
- Euklidischer Abstand (L_2^2-Norm) auf Rotations-Quaternionen $a, b \in \mathbb{H}_1$:
  $\text{dist}(a,b) = \min(\|a-b\|_2^2, \|a+b\|_2^2) \in \mathbb{R}_0^+$ („quat_Euclidean“)
- Frobenius-Norm auf Rotations-Matrizen $a, b \in SO(3)$:
  $\text{dist}(a,b) = \|a-b\|_{\text{Frob}} \in \mathbb{R}_0^+$ („mat_Frobenius“)

Die Antipodalität der Quaternionen wird hierbei wie folgt berücksichtigt: Bei zwei Quaternionen $a, b \in \mathbb{H}_1$ gibt es vier mögliche Vorzeichen-Kombinationen:

$$(a,b), \qquad (a,-b), \qquad (-a,b), \qquad (-a,-b)$$

Davon sind jeweils die Kombinationen (a,b) und $(-a,-b)$, sowie $(a,-b)$ und $(-a,b)$ äquivalent, da für den „Abstand“ nur das *relative* Vorzeichen eine Rolle spielt. Beim geodätischen Abstand wird nun der Absolutbetrag des Skalarproduktes $|\langle a|b\rangle|$ verwendet. Geometrisch bedeutet dies, dass immer der kleinere Winkel zwischen den beiden Quaternionen $\pm a$ und $\pm b$ gewählt wird. Beim Euklidischen Abstand wird die Antipodalität berücksichtigt, indem das Minimum beider Fälle $\|a-b\|$ und $\|a+b\|$ als Abstandsmaß definiert wird. Da die Darstellung einer Drehung als Rotations-Matrix eindeutig ist (siehe Kapitel 1), muss bei diesem Abstandsmaß keine Unterscheidung berücksichtigt werden.

In Abb. 2.35 und 2.36 sind, analog zum vorigen Abschnitt, die Lern-Kurven aller Datensätze für eine Auswahl an Sensoren dargestellt. Hierbei werden nun bei festgelegtem *Zeitreihen*-Abstandsmaß (Lockstep, cDTW, ssDTW) die **punktweisen** Abstandsmaße (quat_geodesic, quat_Euclidean und mat_Frobenius) miteinander verglichen. Die Ergebnisse für den Opportunity-Datensatz (Sensor: RLA) sind in Abb. 2.35 dargestellt. Die Ergebnisse für den QBGR-Datensatz (Sensoren S15_A, S15_I und S19), sowie für den 6DMG-Datensatz, den JGU_Numbers-Datensatz und den qCBF-Datensatz finden sich in Abb. 2.36. Es ist deutlich erkennbar, dass die Lern-Kurven der unterschiedlichen **punktweisen** Abstandsmaße bei allen Datensätzen sehr nahe beieinander liegen.

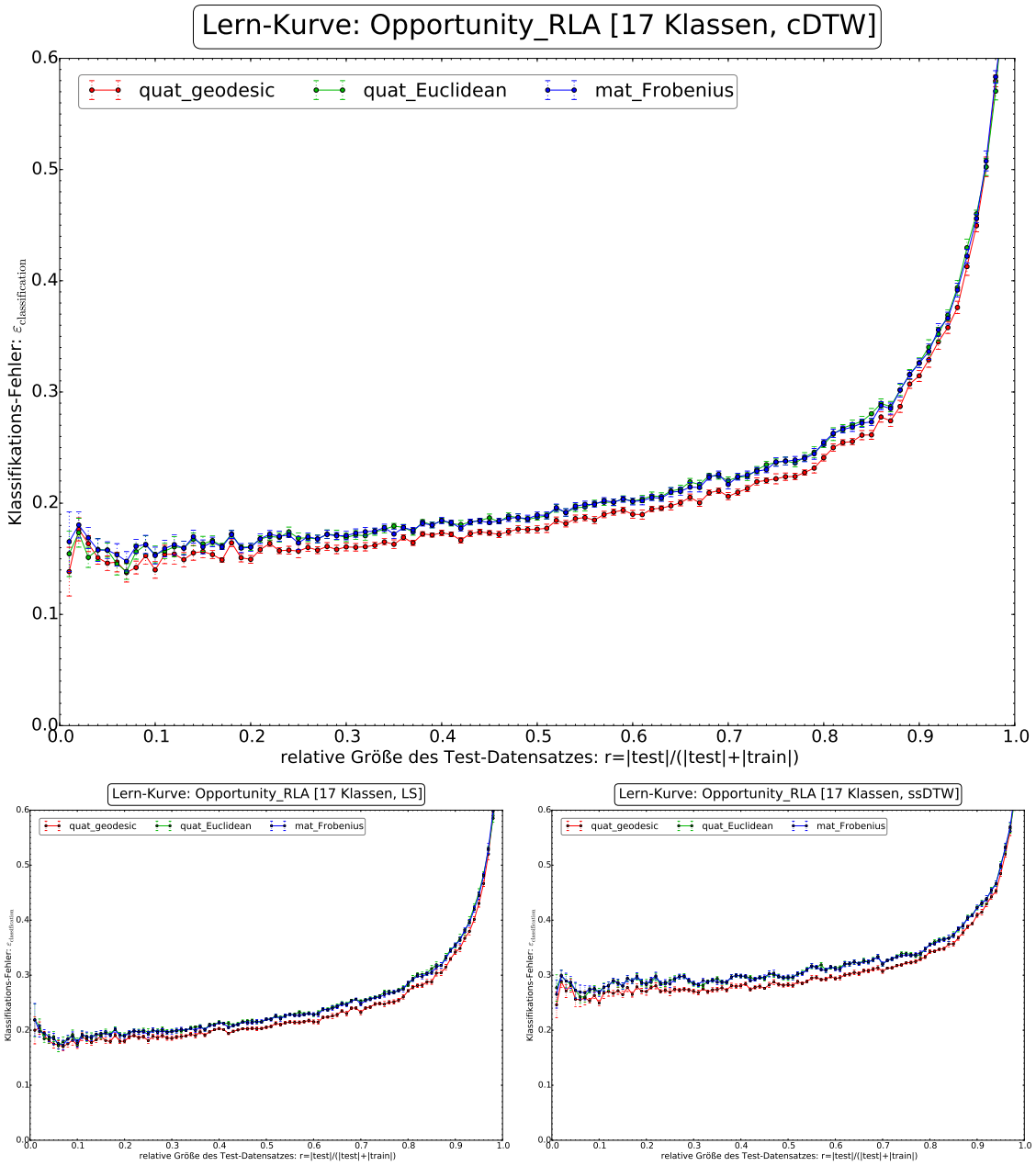


Abbildung 2.35: Lern-Kurve: Opportunity Datensatz – Sensor: RLA (rechter Unterarm).
Vergleich der **punktweisen** Abstandsmaße: quat_geodesic, quat_Euclidean, mat_Frobenius. Dargestellt ist der Mittelwert des Klassifikations-Fehlers über 10 stratifizierte Aufteilungen des Datensatzes (mit dem Standard-Fehler als Fehlerbalken) gegen die relative Größe des Test-Datensatzes. Wie man an diesem Beispiel sieht, sind die Kurven für die verschiedenen *punktweisen* Abstandsmaße (quat_geodesic, quat_Euclidean, mat_Frobenius) sehr ähnlich. Man erkennt ebenfalls deutlich, dass dies für alle *Zeitreihen*-Abstandsmaße (cDTW **[oben]**, Lockstep **[unten links]** und ssDTW **[unten rechts]**) gilt. Da die unterschiedlichen *Zeitreihen*-Abstandsmaße bereits im vorangehenden Abschnitt umfassend untersucht worden sind, konzentrieren wir uns nun auf den Vergleich der **punktweisen** Abstandsmaße. Deshalb werden in der folgenden Abbildung nur die Lernkurven für festgehaltenes *Zeitreihen*-Abstandsmaß cDTW dargestellt.

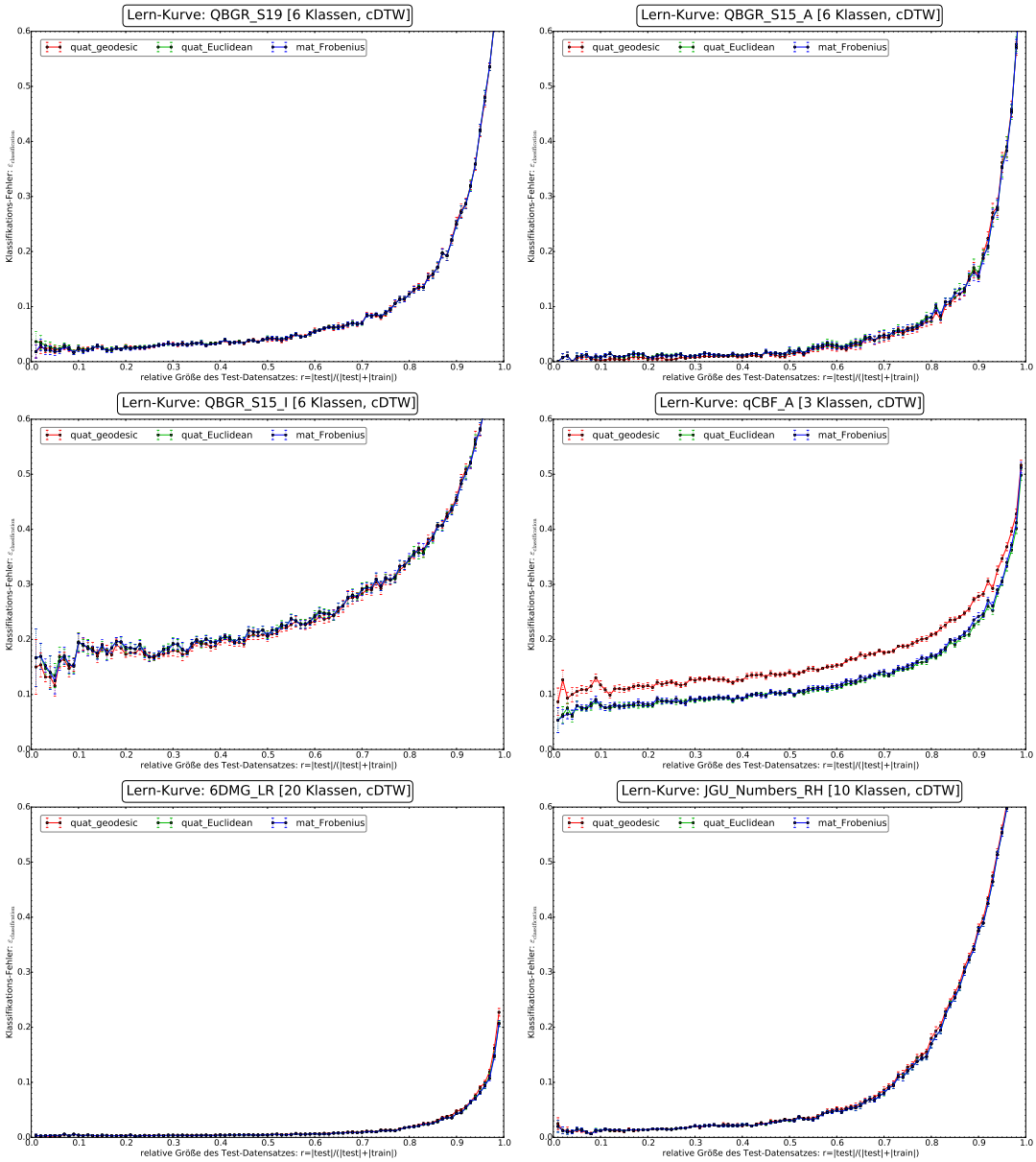


Abbildung 2.36: Lern-Kurven der weiteren Datensätze:
[oben links] QBGR-Datensatz – Sensor S19, **[oben rechts]** QBGR-Datensatz – Sensor S15_A
[mittle links] QBGR-Datensatz – Sensor S15_I, **[mitte rechts]** qCBF-Datensatz („Sensor A“)
[unten links] 6DMG-Datensatz – Sensor LR, **[unten rechts]** JGU_Numbers-Datensatz – RH.

Vergleich der **punktweisen** Abstandsmaße: quat_geodesic, quat_Euclidean, mat_Frobenius. (*Zeitreihen*-Abstandsmaß: cDTW). Es ist deutlich erkennbar, dass die verschiedenen **punktweisen** Abstandsmaße sehr ähnliche (teilweise fast identische) Klassifikations-Ergebnisse liefern. Nur beim qCBF-Datensatz liefert quat_geodesic deutlich schlechtere Ergebnisse als die anderen beiden punktweisen Abstandsmaße.

Zum direkten Vergleich der *punktweisen* Abstandsmaße wurde wieder die Klassifikations-Fehler-Differenz $\Delta\varepsilon_{\text{dist\_c-dist\_r,r}}$ gemäß Gleichung (2.6) ausgerechnet. Als Referenz (dist_r) wurde nun das geodätische Abstandsmaß „quat_geodesic“ verwendet und dieses jeweils mit „quat_Euclidean“ und „mat_Frobenius“ verglichen. In Abb. 2.37 und 2.38 sind wieder der Mittelwert und Standard-Fehler der Klassifikations-Fehler-Differenz $\Delta\varepsilon$ über 100 stratifizierte Aufteilungen des jeweiligen Datensatzes für $r_{\text{test}} \in \{0.1, 0.5, 0.9\}$ dargestellt. Werte $\Delta\varepsilon < 0$ bedeuten: Die Referenz („quat_geodesic“) ist **schlechter**, Werte $\Delta\varepsilon > 0$ bedeuten: die Referenz ist **besser** als das zu vergleichende punktweise Abstandsmaß für die Klassifikation geeignet.

Man sieht, dass „quat_Euclidean“ und „mat_Frobenius“ bezüglich der Differenz zum Referenz-Abstandsmaß „quat_geodesic“ relativ gleich gut beziehungsweise schlecht abschneiden. Beim *Zeitreihen*-Abstandsmaß cDTW ist das geodätische *punktweise* Abstandsmaß „quat_geodesic“ etwas besser als die beiden Vergleichs-Abstandsmaße „quat_Euclidean“ und „mat_Frobenius“: $\Delta\varepsilon > 0$. Dieses Ergebnis ist für cDTW zwar nicht ganz so deutlich, aber es ist zu erkennen, dass das geodätische Abstandsmaß in den meisten Fällen leicht bessere Ergebnisse liefert. Bei den *Zeitreihen*-Abstandsmaßen Lockstep und ssDTW ist das geodätische *punktweise* Abstandsmaß – mit wenigen Ausnahmen – erkennbar deutlich besser als die anderen beiden in Bezug auf den Klassifikations-Fehler.

Nachdem wir nun verschiedene Zeitreihen-Abstandsmaße, sowie verschiedene punktweise Abstandsmaße untereinander verglichen haben, beschäftigen wir uns im kommenden Abschnitt mit der Klassifikation anhand der *inkrementellen Rotationen*.

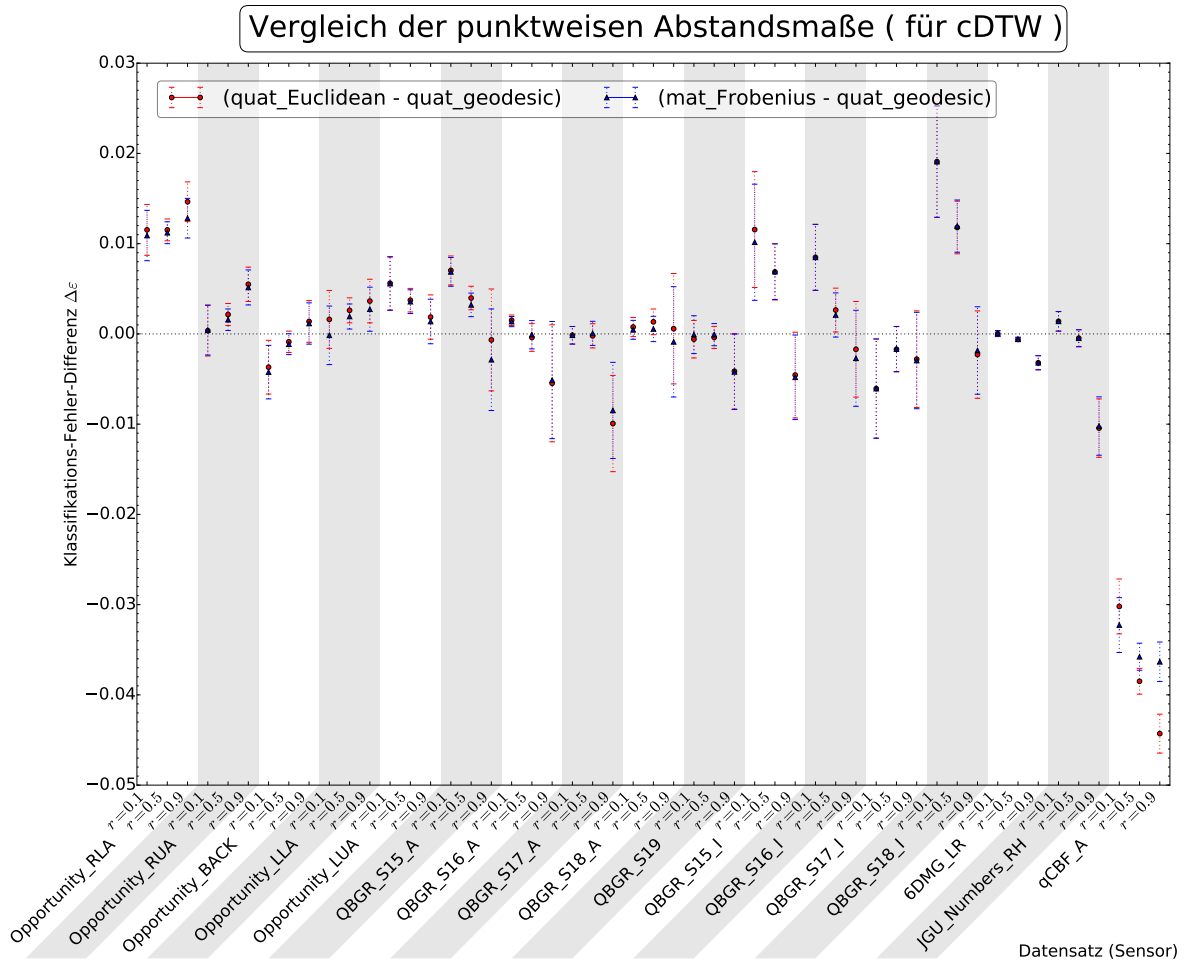


Abbildung 2.37: Vergleich der **punktweisen** Abstandsmaße bei jeweils festem *Zeitreihen*-Abstandsmaß (hier: cDTW). Dargestellt sind jeweils Mittelwert und Standard-Fehler der Klassifikations-Fehler-Differenzen $\Delta\varepsilon$ über 100 stratifizierte Aufteilungen des jeweiligen Datensatzes für $r_{\text{test}} \in \{0.1, 0.5, 0.9\}$. Während „quat_Euclidean“ und „mat_Frobenius“ fast identische Ergebnisse liefern, ist das geodätische Abstandsmaß „quat_geodesic“ meist etwas besser ($\Delta\varepsilon > 0$). Dieser Unterschied ist bei festgehaltenem *Zeitreihen*-Abstandsmaß ssDTW und Lockstep (siehe nächste Abbildung) noch deutlicher erkennbar als bei cDTW.

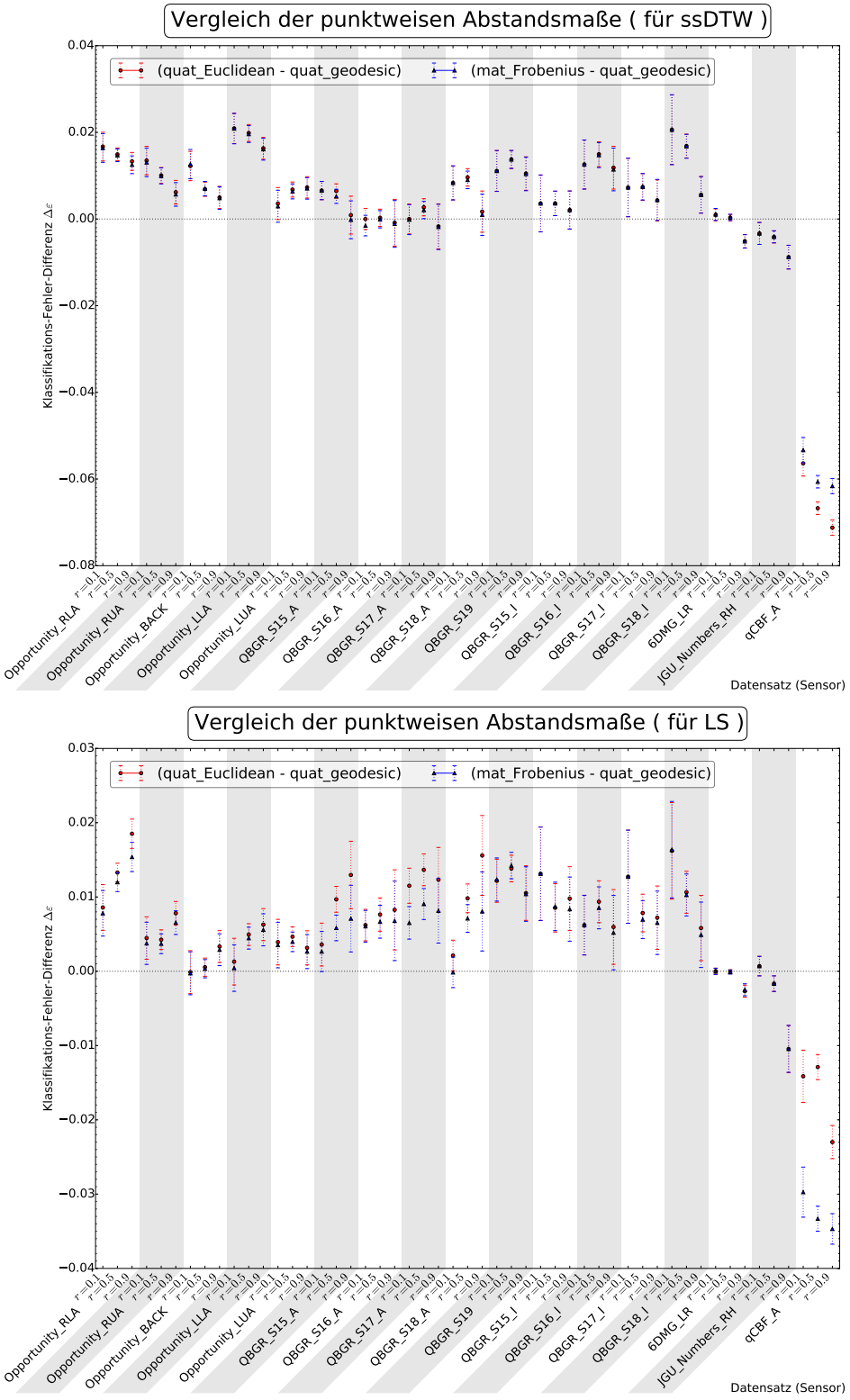


Abbildung 2.38: Vergleich der **punktweisen** Abstandsmaße bei jeweils festem *Zeitreihen*-Abstandsmaß (Fortsetzung). **[oben]**: *Zeitreihen*-Abstandsmaß: ssDTW. **[unten]**: *Zeitreihen*-Abstandsmaß: Lockstep.

2.5.3 Klassifikation: inkrementelle Rotationen

Eine bisher außer Acht gelassene Tatsache ist, dass die bisher als „absolut“ betrachteten Rotationen in Wirklichkeit jeweils von einem (willkürlich gewählten) Bezugssystem abhängen. Die Rotationen stellen immer die **relative** Drehung von einer **Referenz-Orientierung** („Koordinaten-Ursprung“) zur aktuellen Orientierung dar. Bei einer IMU ist dies immer eine bei der Kalibrierung festgelegte Ursprungs-Orientierung. Alle gemessenen Rotationen beschreiben dann die Rotation, die benötigt wird, um die IMU aus ihrer Ursprungs-Orientierung in die aktuelle Orientierung zu bewegen. Ein offensichtlicher Nachteil dabei ist, dass wenn z.B. unterschiedliche Probanden eine Geste ausführen, die aufgenommenen Rotations-Daten immer relativ zur bei der jeweiligen Kalibrierung der IMU bestimmten Referenz-Lage sind. Eine solche Kalibrierung muss jedoch für jeden Probanden erneut durchgeführt werden. Wenn bei den Messungen unterschiedlicher Probanden keine feste gemeinsame Referenz-Orientierung gewählt wird, sind die Rotations-Daten unterschiedlicher Probanden im Allgemeinen folglich in unterschiedlichen Bezugssystemen gemessen. Eine solche feste gemeinsame Referenz-Orientierung könnte man beispielsweise erzeugen, indem man eine IMU, die an der Hand eines Probanden angebracht ist, immer in der gleichen Position kalibriert. Dazu könnte man z.B. die Hand in einer festgelegten Orientierung auf einem Tisch ablegen und in dieser Position – die für alle Probanden gleich sein muss – die Kalibrierung durchführen. Aber selbst bei diesem Vorgehen tritt das oben beschriebene Problem auf, denn die Orientierung der IMU relativ zur Hand kann bei unterschiedlichen Probanden trotzdem immer noch unterschiedlich sein. Auch wenn die verwendete(n) IMU(s) „fest“ an einem Ganzkörper-Anzug oder einem Handschuh angebracht sind, sitzt dieser nicht bei allen Menschen gleich und deshalb können selbst dann unterschiedliche Referenz-Orientierungen vorhanden sein, wenn ein solcher einheitlicher Kalibrierungs-Schritt vorgenommen wurde.

Auch wenn optische Systeme die „absolute“ Pose eines Markers im Raum bestimmen können, so tritt das Problem dennoch auch dort auf. Zum einen ist auch hier die Anbringung des Markers am Körper der Probanden nicht immer exakt gleich. Zum anderen müssten alle Probanden die Gesten immer im exakt gleichen Winkel zur Kamera durchführen, da bei diesen Systemen die Referenz-Pose (Orientierung und Position) in der Regel im Ursprung des Bezugssystems der Kamera liegt. Sitzt oder steht ein Proband in einem anderen Winkel zur Kamera als ein anderer Proband, so sind auch hier wiederum effektiv unterschiedliche Referenz-Orientierungen vorhanden, da die Gesten relativ zum Kamera-Ursprung in einem anderen Winkel aufgenommen werden.

Eine Möglichkeit, dieses Problem zu lösen ist es, statt den originalen Messwerten die Zeitreihen der **inkrementellen Rotationen** zu betrachten. Hierbei werden nicht die Rotationen selbst, sondern jeweils die relative Änderung einer Rotation zu der (zeitlich) nachfolgenden Rotation betrachtet. Dadurch ist diese Darstellung unabhängig von der gewählten Re-

ferenz-Lage und damit für alle Probanden gleich (Beweis: siehe unten).

Die inkrementellen Rotationen lassen sich ähnlich wie eine „Zeitableitung“ definieren[8]. Für eine sinnvolle Definition einer Quaternionen-wertigen Zeitreihe bestehend aus inkrementellen Rotationen betrachten wir zunächst einmal zum Vergleich die Zeitableitung einer eindimensionalen Zeitreihe. Bei einer durch diskrete Stützstellen gegebenen Zeitreihe lässt sich die Zeitableitung (approximativ) durch Differenzen-Quotienten definieren:

Definition 24: Zeitableitung einer reellwertigen Zeitreihe

Sei $X = \{x_0, \ldots, x_{n-1}\}$ mit $x_i \in \mathbb{R}\,\forall\, i$ eine reellwertige Zeitreihe. Dann ist die (diskrete) Zeit-Ableitung X'_h dieser Zeitreihe mit der Schrittweite $h \in \mathbb{N}$ definiert als:

$$X'_h := \{x'_0, \ldots, x'_{n-h-1}\} \quad \text{mit} \quad x'_i := \frac{\Delta x}{\Delta t} = \frac{x_{i+h} - x_i}{t + h - t} = \frac{1}{h}(x_{i+h} - x_i)$$

Analog dazu lässt sich für eine Quaternionen-wertige Zeitreihe die zugehörige Zeitreihe der inkrementellen Rotationen definieren. Da die (normierten) Quaternionen $q \in \mathbb{H}_1$ eine **multiplikative Gruppe** bilden, wird **statt der Differenz** $x_{i+h} - x_i$ (also der Addition der Inversen $-x_i$) **die Multiplikation mit der Inversen** $q_{i+h} * q_i^{-1}$ verwendet. Da das Ergebnis wieder ein Gruppenelement und damit ein normiertes Quaternion ist, entfällt der Faktor $\frac{1}{h}$ hierbei, da dieser durch die Normierung keine Rolle spielt. Damit lässt sich für eine gegebene Quaternionen-wertige Zeitreihe die zugehörige **Zeitreihe der inkrementellen Rotationen** nun wie folgt definieren:

Definition 25: Zeitreihe inkrementeller Rotationen

Sei $Q = \{q_0, \ldots, q_{n-1}\}$ mit $q_i \in \mathbb{H}_1\,\forall\, i$ eine Quaternionen-wertige Rotations-Zeitreihe. Dann ist die zugehörige Zeitreihe inkrementeller Rotationen Q'_h mit Schrittweite $h \in \mathbb{N}$ definiert als:

$$Q'_h := \{q'_0, \ldots, q'_{n-h-1}\} \quad \text{mit} \quad q'_i := q_{i+h} * q_i^{-1}$$

Dabei ist $(*)$ das Quaternionen-Produkt in $\mathbb{H}_1$.

Nun gilt es noch zu zeigen, dass diese Darstellung unabhängig von der gewählten Referenz-Orientierung ist. Dies lässt sich wie folgt beweisen:

[8] Formal sind die inkrementellen Rotationen keine „echte“ Zeitableitung der Quaternionen, da für die mathematisch korrekte Zeitableitung der Logarithmus der inkrementellen Rotations-Quaternionen benötigt wird.

Beweis 2

Seien $q_1, q_2 \in \mathbb{H}_1$ zwei Rotationen. Dann ist die „Differenz"-Rotation Δq, die einen Körper aus der Lage q_1 in die Lage q_2 überführt, gegeben durch:

$$\Delta q = q_2 * q_1^{-1}$$

Seien nun $\tilde{q}_1, \tilde{q}_2, q_{\text{ref}} \in \mathbb{H}_1$ mit $\tilde{q}_1 := q_1 * q_{\text{ref}}^{-1}$ und $\tilde{q}_2 := q_2 * q_{\text{ref}}^{-1}$ die selben Rotationen, jedoch relativ zu einer Referenz-Orientierung q_{ref} gemessen. Die „Differenz"-Rotation $\Delta\tilde{q}$ ist dann gegeben als

$$\begin{aligned}
\Delta\tilde{q} = \tilde{q}_2 * \tilde{q}_1^{-1} &= (q_2 * q_{\text{ref}}^{-1}) * (q_1 * q_{\text{ref}}^{-1})^{-1} \\
= (q_2 * q_{\text{ref}}^{-1}) * (q_{\text{ref}} * q_1^{-1}) &= q_2 * (q_{\text{ref}}^{-1} * q_{\text{ref}}) * q_1^{-1} \\
= q_2 * q_{\mathbb{1}} * q_1^{-1} &= q_2 * q_1^{-1} \\
= \Delta q & \qquad \square
\end{aligned}$$

$(*)$ ist hierbei das Quaternionen-Produkt in $\mathbb{H}_1$.

Der Parameter $h \in \mathbb{N}$ ist hierbei die Schrittweite der inkrementellen Rotation. $h = 1$ bedeutet, dass jeweils die inkrementelle Rotation zwischen der Orientierung zu einem Zeitpunkt t_i zur Orientierung zum direkt nachfolgenden Zeitpunkt t_{i+1} in der Zeitreihe betrachtet wird. $h = 2$ bedeutet entsprechend die inkrementelle Rotation der Orientierung von einem Zeitpunkt t_i zur Orientierung am übernächsten Zeitpunkt t_{i+2}, usw.

Um einen ersten Überblick darüber zu erhalten, wie der Parameter h zu wählen ist, wurden für Werte zwischen $1 \leq h \leq 30$ die Lernkurven für alle Datensätze bestimmt. Um die dazu benötigte Rechenzeit zu reduzieren, wurde der Parameterbereich der relativen Größe des Test-Datensatzes r_{test} nur grob von $r_{\text{test}} = 0.1$ bis $r_{\text{test}} = 0.9$ in Schritten der Größe 0.1 abgetastet. Zu jedem Wert von r_{test} wurde dann wieder der über 10 stratifizierte Aufteilungen der Daten gemittelte Klassifikations-Fehler bestimmt. Um den Unterschied bezüglich des Klassifikations-Fehlers zwischen der Verwendung der „originalen" (absoluten) Orientierungen und der inkrementellen Rotationen quantitativ bewerten zu können, wurde die Differenz der jeweiligen Klassifikations-Fehler (analog zu Gleichung (2.6)) bestimmt:

$$\Delta\varepsilon_{\text{abs-incr,r}} := \varepsilon_{\text{abs,r}} - \varepsilon_{\text{incr,r}} \tag{2.7}$$

Für diesen wurden dann jeweils Mittelwert und Standard-Fehler über alle r_{test} berechnet. Werte von $\Delta\varepsilon_{\text{abs-incr,r}} < 0$ bedeuten, dass $\varepsilon_{\text{abs,r}} < \varepsilon_{\text{incr,r}}$ ist und damit die Verwendung der inkrementellen Rotationen höhere Klassifikations-Fehler – also **schlechtere** Ergebnisse – liefert. Werte von $\Delta\varepsilon_{\text{abs-incr,r}} > 0$ bedeuten entsprechend, dass die Verwendung der inkrementellen Rotationen **bessere** Ergebnisse liefert.

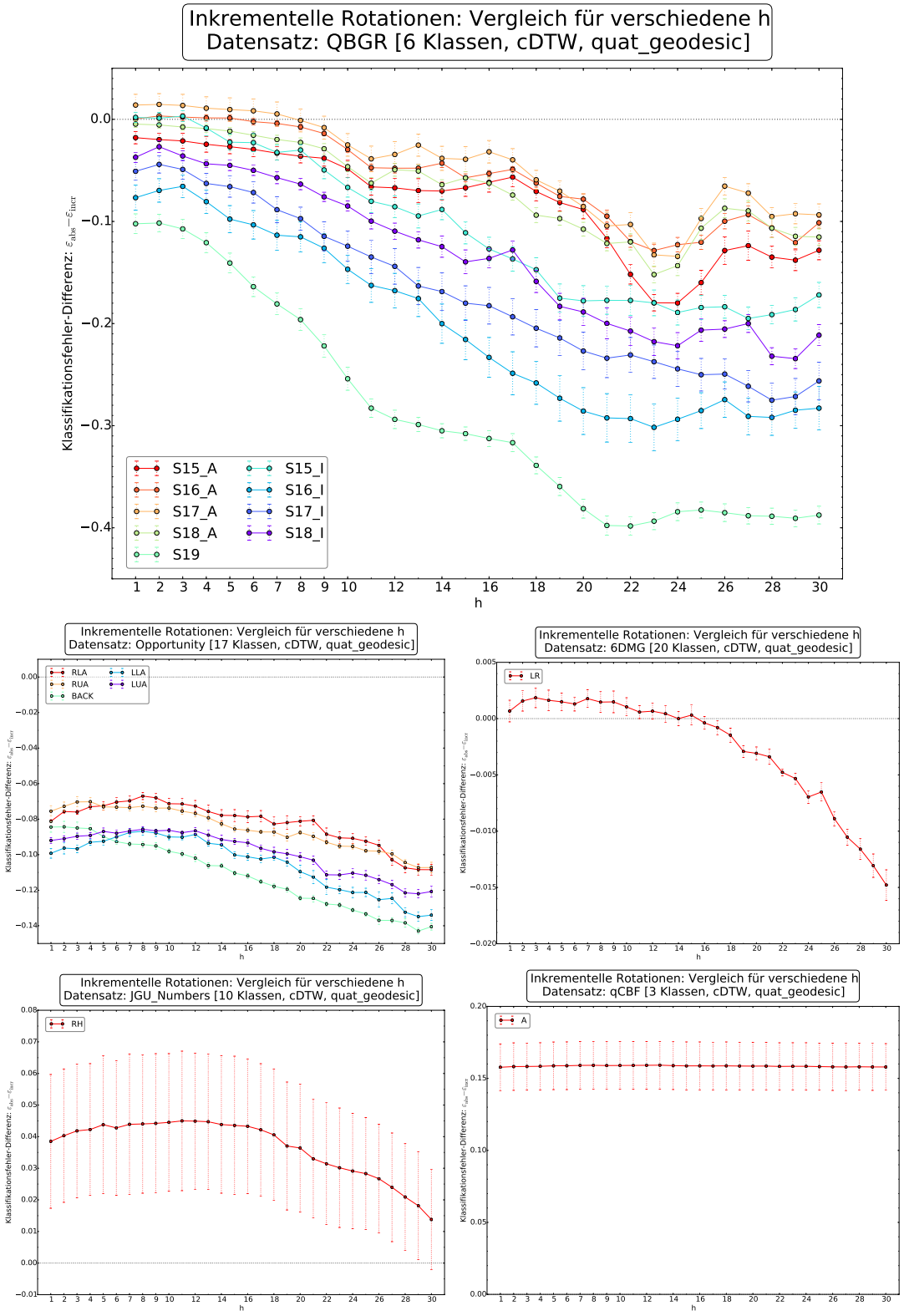


Abbildung 2.39: Inkrementelle Rotationen: Vergleich für verschiedene Werte der Schrittweite h. Dargestellt ist der Mittelwert der Klassifikationsfehler-Differenz der absoluten Orientierungen zu den inkrementellen Rotationen über 9 Werte von r_{test} (siehe Text). Der Standard-Fehler ist wieder durch die Fehlerbalken repräsentiert.

Die Ergebnisse sind in Abb. 2.39 dargestellt. Man erkennt, dass die Klassifikations-Genauigkeit für den Opportunity-Datensatz durch den Ansatz der inkrementellen Rotationen deutlich verschlechtert wird. Beim QBGR-Datensatz ist eine leichte Verbesserung der Klassifikations-Genauigkeit für die Sensoren 16_A und S17_A für $h \leq 5$ beobachtbar, während die anderen Sensoren – insbesondere die Sensoren am inaktiven Arm (Sxx_I) – jeweils deutlich schlechter abschneiden. Für den 6DMG-Datensatz ist die Verwendung der inkrementellen Rotationen für Werte von $h \leq 15$ besser geeignet als die absoluten Orientierungen. Für den JGU_Numbers-Datensatz, sowie für den qCBF-Datensatz ist die Klassifikations-Genauigkeit mittels inkrementeller Rotationen sogar für jeden Wert von $1 \leq h \leq 30$ deutlich besser als bei den absoluten Orientierungen.

Bei der Wahl des Parameters h muss auch beachtet werden, dass dieser nicht zu hoch gewählt werden sollte. Zum einen wird die Approximation der inkrementellen Rotationen durch den „Differenzen"-Quotienten immer ungenauer, je größer h gewählt wird. Genau genommen berechnet man damit die *mittlere* Änderung der Orientierung im Zeitintervall der Länge h. Zum anderen wird durch die Diskretisierung die Zeitreihe der inkrementellen Rotationen um h Elemente gegenüber der originalen Zeitreihe gekürzt. Deshalb sollte für eine Zeitreihe der Länge $|Q|$ der Parameter h immer $h \ll |Q|$ gewählt sein, um die Zeitreihe nicht zu sehr abzuschneiden. Aus diesen Gründen werden im Folgenden beispielhaft die Schrittweiten $h = 1$ und $h = 3$ näher untersucht.

Abb. 2.40 zeigt die Klassifikations-Fehler-Differenz der einzelnen Datensätze noch einmal aufgelöst für jeweils jeden der drei Werte von $r_{\text{test}} \in \{0.1, 0.5, 0.9\}$. Um die Genauigkeit zu erhöhen, wurde hierbei wieder über 100 stratifizierte Aufteilungen der Daten gemittelt. Es ist erkennbar, dass die Verwendung inkrementeller Rotationen für alle Teildatensätze (Sensoren) des **Opportunity-Datensatzes** die Klassifikations-Raten deutlich verschlechtert. Für den **QBGR-Datensatz** sind die Ergebnisse der einzelnen Sensoren unterschiedlich: Während die Klassifikations-Raten bei den Sensoren S15_A, S18_A, S19 und S18_I schlechter werden, so bringt es für die Sensoren S16_A, S17_A, S15_I eine teilweise Verbesserung der Klassifikations-Raten (insbesondere bei $r_{\text{test}} = 0.9$). Für die beiden Sensoren S16_I und S17_I ist eine Verbesserung nur für $r_{\text{test}} = 0.9$ vorhanden, während bei kleineren Werten von r_{test} eine deutliche Verschlechterung eintritt. Die drei Datensätze **6DMG**, **JGU_Numbers** und **qCBF** profitieren alle von der Verwendung inkrementeller Rotationen für jeden der drei Werte von r_{test}, am stärksten jedoch auch hier für $r_{\text{test}} = 0.9$.

Allgemein ist die größte Verbesserung der Klassifikations-Raten für den jeweils rechten Teil der Lern-Kurven (für große Werte von r_{test}) sichtbar. Dort ist der Test-Datensatz groß und der Trainings-Datensatz klein. In diesem Bereich liegt die Stärke der inkrementellen Rotationen: Wenn nur wenige Daten zum Lernen zur Verfügung stehen, dann ist die Klassifikation anhand der inkrementellen Rotationen genauer als bei Betrachtung der absoluten Orientierungen. Dies lässt sich dadurch erklären, dass bei Verwendung der absoluten Ori-

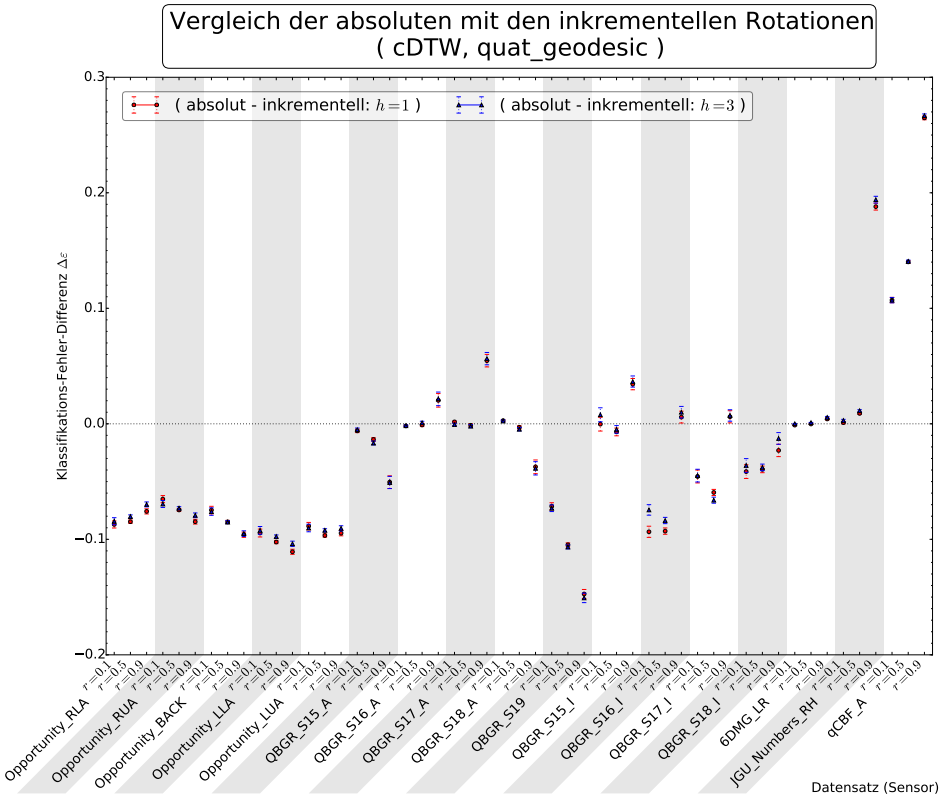


Abbildung 2.40: Vergleich: absolute vs. inkrementelle Rotationen. Zeitreihen-Abstandsmaß: cDTW, punktweises Abstandsmaß: quat_geodesic. Dargestellt sind wieder Mittelwert und Standard-Fehler der Klassifikations-Fehler-Differenz $\Delta\varepsilon$ über 100 stratifizierte Aufteilungen des jeweiligen Datensatzes.

entierung bei großem Trainings-Datensatz zu einem gegebenen Test-Kandidaten aus dem Test-Datensatz mit höherer Wahrscheinlichkeit ein passendes Shape gefunden wird, welches die passende Referenz-Orientierung besitzt (also im gleichen Bezugssystem gemessen wurde wie der Test-Kandidat). Je kleiner der Trainings-Datensatz ist, desto unwahrscheinlicher ist dieser Fall. Dadurch steigt der Klassifikations-Fehler bei kleiner werdendem Trainings-Datensatz. Betrachtet man die inkrementellen Rotationen, spielt die Referenz-Orientierung keine Rolle, sodass dieses Verfahren in diesem Bereich der Lernkurve besser abschneidet.

Beim Vergleich der **Zeitreihen**-Abstandsmaße (siehe Abb. 2.41) zeigt sich, dass cDTW den anderen beiden Maßen Lockstep und ssDTW auch bei der Verwendung der inkrementellen Rotationen überlegen ist. Auch hier ist ssDTW in den meisten Fällen wieder mit Abstand am schlechtesten, während Lockstep und cDTW wieder sehr ähnliche Ergebnisse liefern, wobei cDTW immer etwas besser ist als Lockstep. Dieses Resultat ist also sehr ähn-

lich wie bereits bei den absoluten Orientierungen (vergleiche Abb. 2.33 und 2.34). Das ss-DTW-Abstandsmaß ist bei Verwendung der inkrementellen Rotationen sogar noch deutlich schlechter als bei der Verwendung der absoluten Orientierungen. Die bereits erwähnte Problematik, dass Bewegungen als Teil einer anderen Bewegung gefunden werden, kommt bei den inkrementellen Rotationen noch stärker zum Vorschein.

Der Vergleich der **punktweisen** Abstandsmaße zeigt hier, dass das geodätische Abstandsmaß *quat_geodesic* bei der Verwendung der inkrementellen Rotationen gegenüber den anderen beiden (*quat_Euclidean* und *mat_Frobenius*) in den meisten Fällen signifikant bessere Klassifikations-Ergebnisse liefert (siehe Abb. 2.42, 2.43 und 2.44). Dieser Unterschied der punktweisen Abstandsmaße ist bei den absoluten Orientierungen nicht so stark ausgeprägt (vergleiche Abb. 2.37 und 2.38).

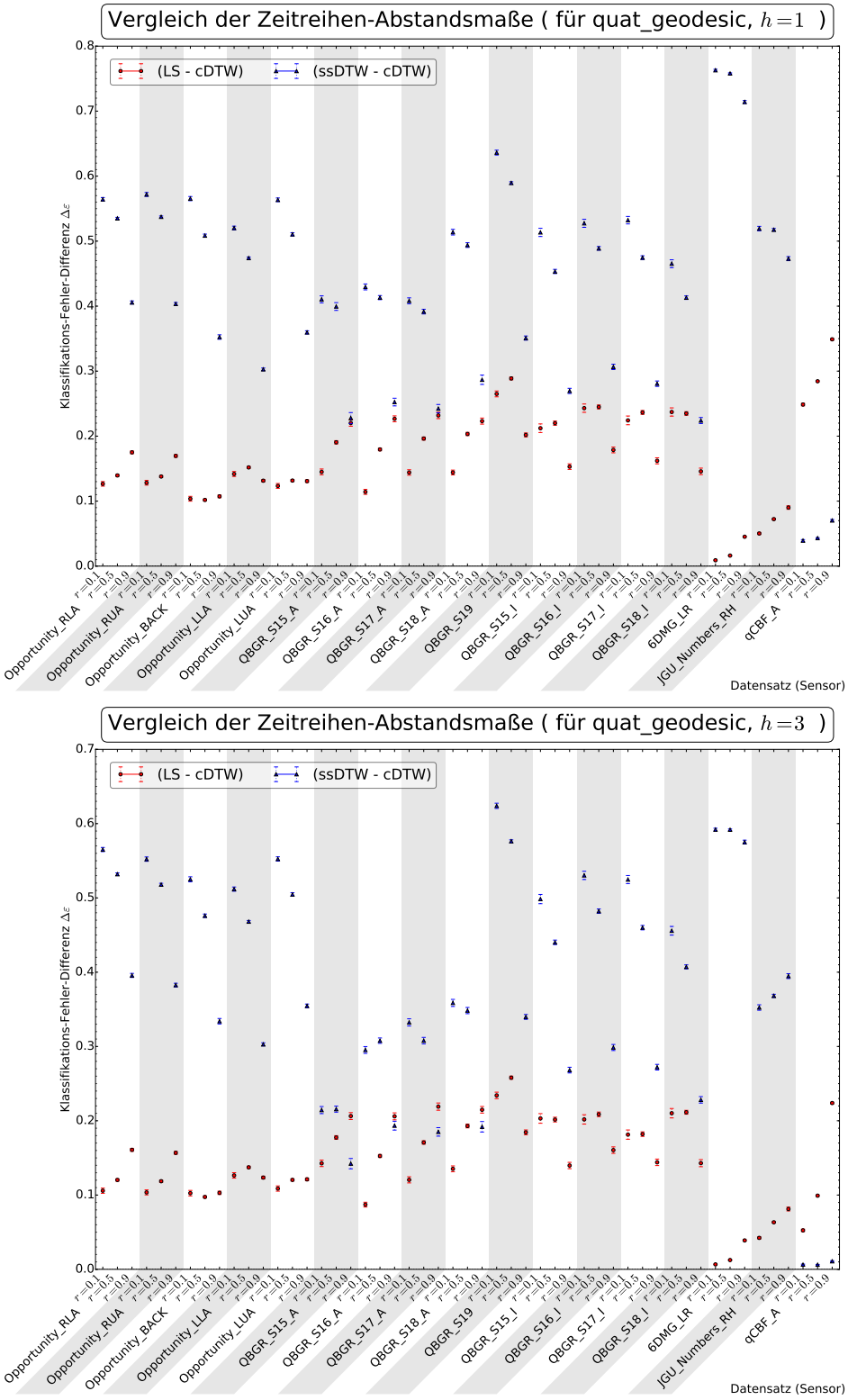


Abbildung 2.41: Vergleich der **Zeitreihen**-Abstandsmaße bei Verwendung der inkrementellen Rotationen für $h = 1$ **[oben]** und $h = 3$ **[unten]**. (*Punktweises* Abstandsmaß: quat_geodesic). Auch hier zeigt sich, dass ssDTW meist deutlich schlechtere Klassifikations-Ergebnisse liefert als Lockstep und cDTW. Nach wie vor ist cDTW wieder das beste *Zeitreihen*-Abstandsmaß.

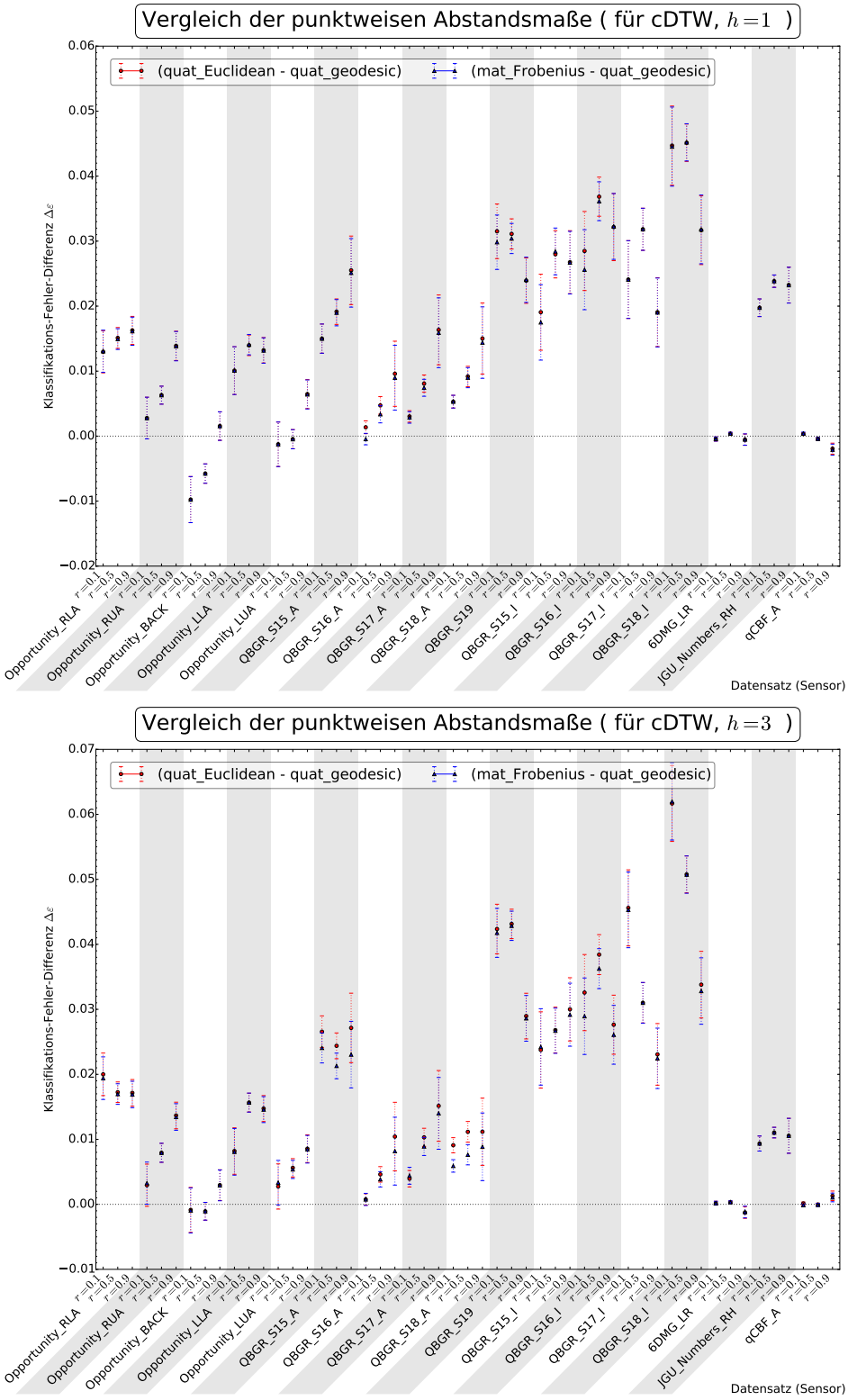


Abbildung 2.42: Vergleich der **punktweisen** Abstandsmaße bei Verwendung der inkrementellen Rotationen für $h = 1$ **[oben]** und $h = 3$ **[unten]**. (*Zeitreihen*-Abstandsmaß: cDTW). Bei der Verwendung inkrementeller Rotationen ist der Vorsprung des geodätischen Abstandsmaßes „quat_geodesic“ gegenüber „quat_Euclidean“ und „mat_Frobenius“ noch etwas stärker ausgeprägt als bei der Verwendung der absoluten Orientierungen (vgl. Abb. 2.37 und 2.38).

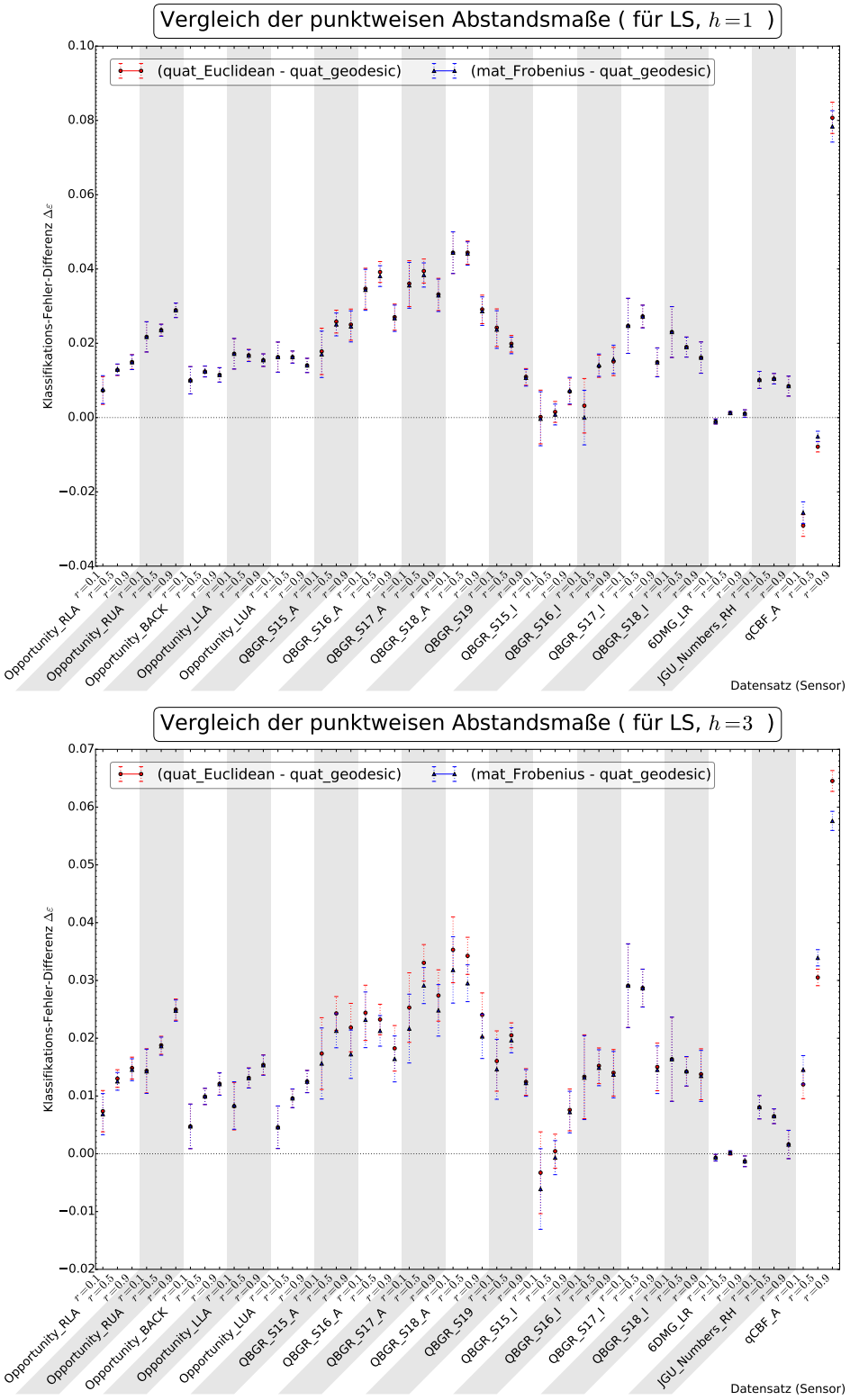


Abbildung 2.43: Vergleich der **punktweisen** Abstandsmaße bei Verwendung der inkrementellen Rotationen für $h = 1$ **[oben]** und $h = 3$ **[unten]** (Fortsetzung). (*Zeitreihen*-Abstandsmaß: Lockstep). Hier zeigt sich noch deutlicher als bei cDTW, dass das geodätische Abstandsmaß gegenüber den anderen beiden fast immer bessere Klassifikations-Raten erzielt.

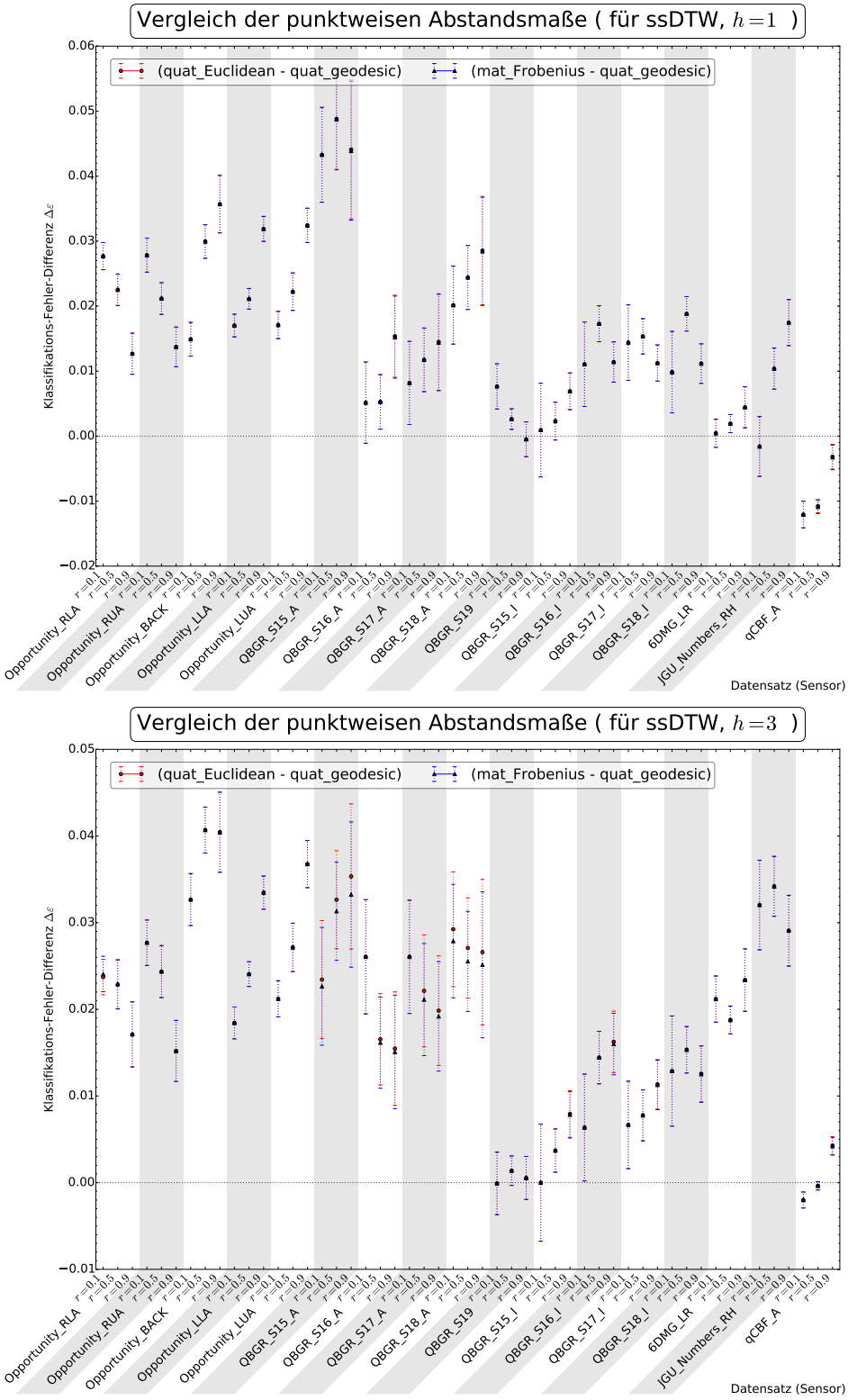


Abbildung 2.44: Vergleich der **punktweisen** Abstandsmaße bei Verwendung der inkrementellen Rotationen für $h = 1$ **[oben]** und $h = 3$ **[unten]** (Fortsetzung). (*Zeitreihen*-Abstandsmaß: ssDTW). Auch hier zeigt sich noch deutlicher als bei cDTW, dass das geodätische Abstandsmaß gegenüber den anderen beiden fast immer bessere Klassifikations-Raten erzielt.

Inkrementelle Rotationen beim Opportunity-Datensatz

Der Opportunity-Datensatz ist der einzige untersuchte Datensatz, bei dem die Klassifikation anhand der inkrementellen Rotationen ausschließlich deutlich schlechtere Ergebnisse liefert. Dies lässt sich wie folgt begründen: Beim Opportunity-Datensatz sind die „Gesten“ nicht alle klar unterscheidbar. Die Greifbewegungen beispielsweise zum Öffnen der Spülmaschine und die Greifbewegungen zum Öffnen einer der drei Schubladen sind sehr ähnlich zueinander. Betrachtet man die absoluten Orientierungen, so kann dadurch noch etwas besser unterschieden werden, ob der Proband z.B. nach der Schublade 1, 2 oder 3 greift, da diese in unterschiedlichen Höhen angebracht sind und damit der Winkel des greifenden Armes anhand der absoluten Orientierung noch erkennbar ist (auch wenn dies – wie im vorangehenden Abschnitt bereits beschrieben – nur eingeschränkt funktioniert). Aus diesem Grunde sind die Bewegungen bei Betrachtung der absoluten Orientierungen in diesem Falle zumindest besser klassifizierbar. Hinzu kommt, dass die Bewegungen im Vergleich zur Messfrequenz relativ langsam sind. Deshalb bestehen die inkrementellen Rotationen aus sehr „kleinen“ Werten. Im Raum der normierten Quaternionen bedeutet „klein“, dass die Werte recht nahe am Einheits-Quaternion $q_{\mathbb{1}} = (1,0,0,0)^\top \in \mathbb{H}_1$ liegen. Wenn man die Zeitreihen der absoluten Orientierungen im Vergleich zu den Zeitreihen der inkrementellen Rotationen betrachtet, wird dies deutlich. Eine Auswahl an Shapes aus dem Opportunity-Datensatz ist in Abb. 2.46 exemplarisch dargestellt. Man sieht deutlich, dass die inkrementellen Zeitreihen alle sehr ähnlich sind. Selbst wenn man nicht die direkten Nachbarn zweier Zeitreihenpunkte ($h = 1$) betrachtet sondern z.B. den drittnächsten Nachbarn ($h = 3$), ändert dies wenig daran, dass die inkrementellen Zeitreihen immer noch sehr ähnlich zueinander sind. Dies spiegelt sich auch in den Klassifikations-Ergebnissen wider: Die verschiedenen Bewegungen sind als inkrementelle Rotationen schwerer zu unterscheiden. Abb. 2.45 zeigt die Confusion-Matrix für den Sensor RLA. Man erkennt, dass es Verwechslungen zwischen den meisten Klassen gibt.

Inkrementelle Rotationen beim qCBF-Datensatz

Beim qCBF-Datensatz funktioniert die Klassifikation anhand der inkrementellen Rotationen bedeutend besser als bei der Verwendung der absoluten (originalen) Rotationen. Hier zeigt sich deutlich, dass die Koordinatensystem-unabhängige Darstellung der Orientierungen mit Hilfe der inkrementellen Rotationen sehr gut funktioniert. Die Voraussetzung hierfür ist jedoch, dass die Bewegungen im Vergleich zur Messfrequenz nicht zu langsam ablaufen, da die Zeitreihen der inkrementellen Rotationen sonst nicht mehr gut unterscheidbar sind, wie es z.B. beim Opportunity-Datensatz der Fall ist (siehe oben). Abb. 2.47 zeigt noch einmal die Lern-Kurven für die absoluten Orientierungen und die inkrementellen Rotationen für $h = 1$ und $h = 3$ im direkten Vergleich: Während der Klassifikations-Fehler auf den absoluten Orientierungen mit über 10% recht hoch ist, funktioniert die Klassifikation auf den inkrementellen Rotationen nahezu perfekt.

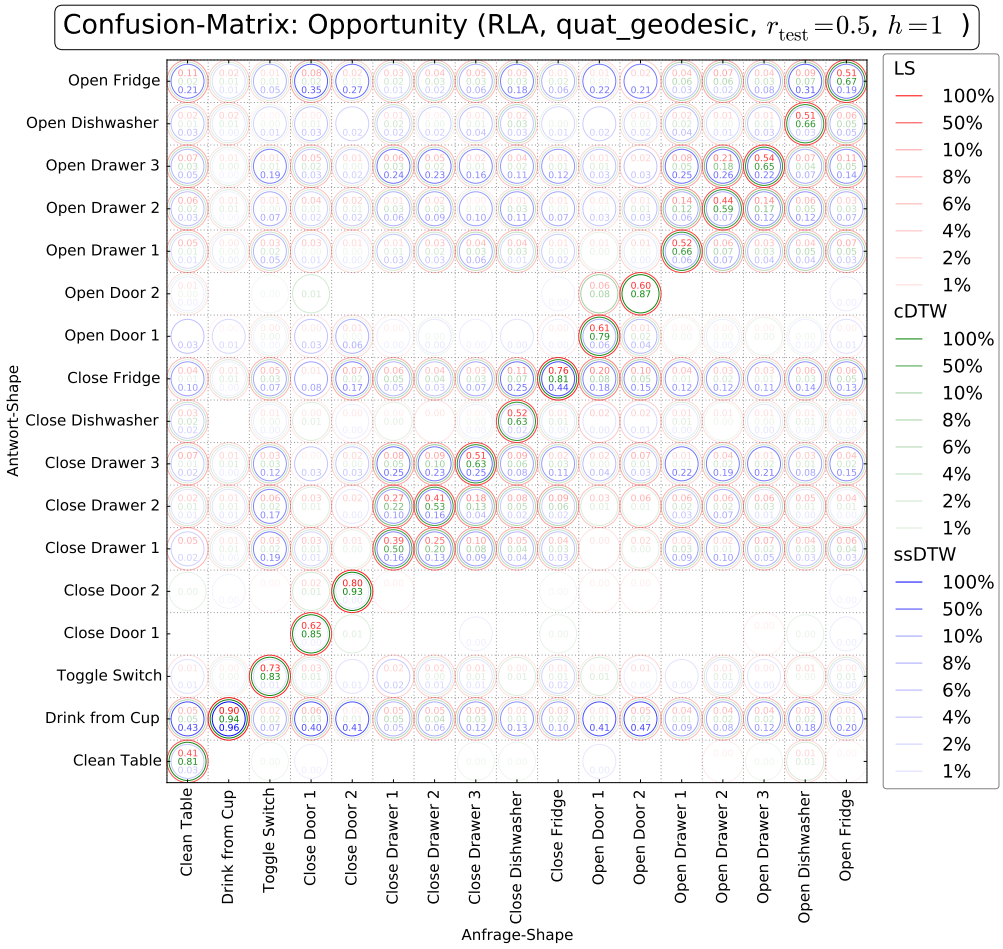


Abbildung 2.45: Confusion-Matrix: Opportunity-Datensatz: (Sensor RLA, punktweises Abstandsmaß: quat_geodesic, $r_{\text{test}} = 0.5$, $h = 1$). Es ist deutlich zu erkennen, dass durch die Verwendung der inkrementellen Rotationen sehr viele Klassen untereinander verwechselt werden.

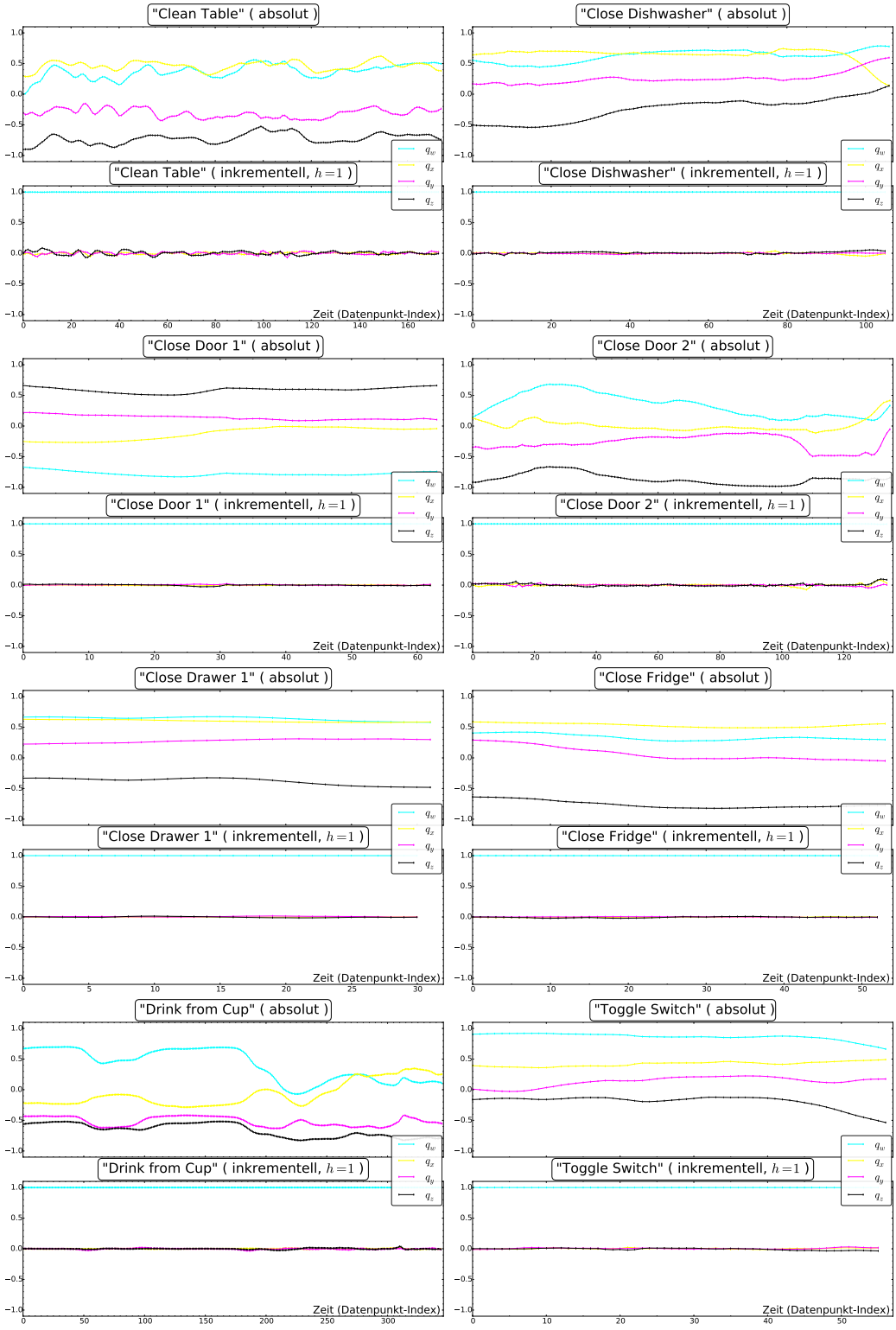


Abbildung 2.46: Shape-Vergleich: absolute Orientierungen vs. inkrementelle Rotationen – Datensatz: Opportunity (Sensor: RLA) [Auswahl]. **[Jeweils oben]**: (originale) Zeitreihe der absoluten Orientierungen, **[jeweils unten]**: Zeitreihe der inkrementellen Rotationen ($h = 1$). Durch die im Vergleich zur Messfrequenz relativ langsamen Bewegungen sehen die Zeitreihen der inkrementellen Rotationen alle sehr ähnlich aus.

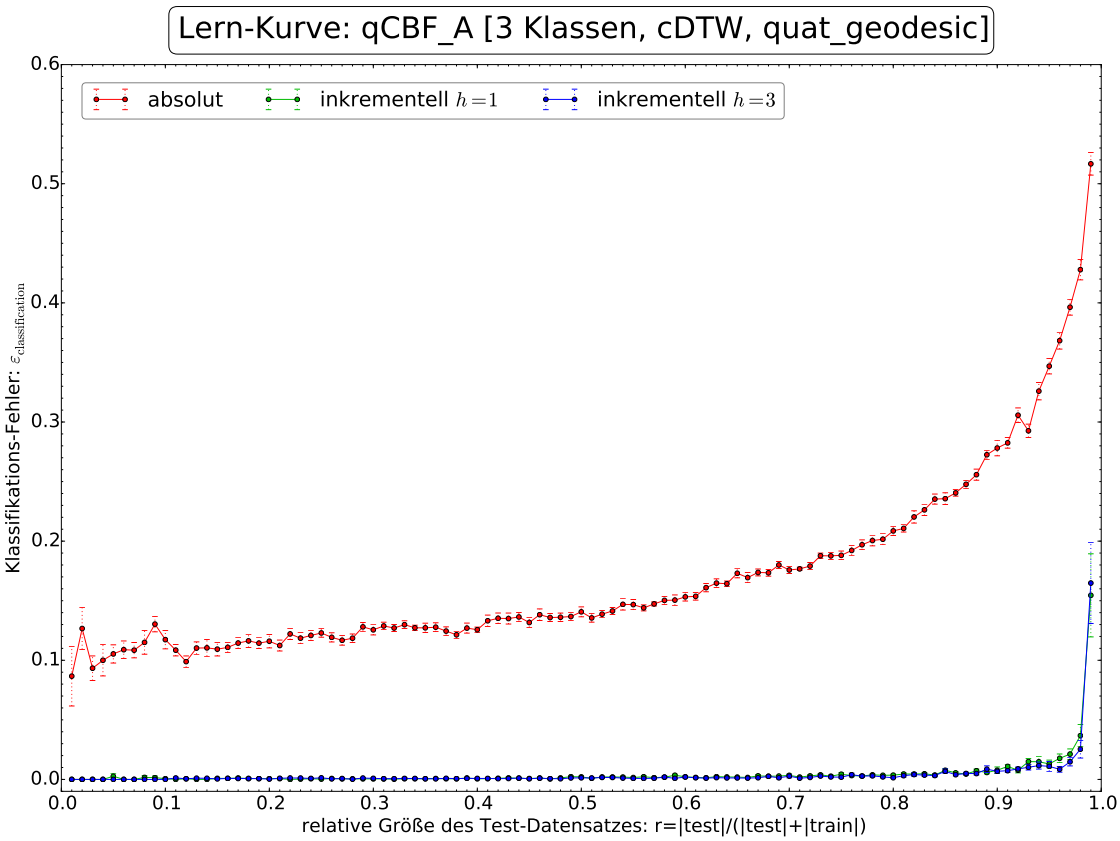


Abbildung 2.47: Lernkurven-Vergleich: qCBF-Datensatz. Absolute Orientierungen vs. inkrementelle Rotationen. Zeitreihen-Abstandsmaß: cDTW, punktweises Abstandsmaß: quat_geodesic. Während der Klassifikations-Fehler bei der Verwendung der absoluten Orientierungen mit über 10% recht hoch ist, funktioniert die Klassifikation auf den inkrementellen Rotationen nahezu perfekt. Der Klassifikations-Fehler ist fast für die gesamte Lernkurve identisch Null. Erst für $r_{\text{test}} > 0.9$ nimmt er leicht zu und liegt selbst bei $r_{\text{test}} = 0.98$ noch unter 4%.

2.5.4 Klassifikation: Zusammenfassung

In diesem Abschnitt wurden die drei *Zeitreihen*-Abstandsmaße *„Lockstep“*, *„cDTW“* und *„ssDTW“* sowie die drei *punktweisen* Abstandsmaße *„quat_geodesic“*, *„quat_Euclidean“* und *„mat_Frobenius“* bezüglich ihrer Eignung für eine 1NN-Klassifikation auf Rotations-Zeitreihen miteinander verglichen. Des Weiteren wurde die Klassifikation einmal auf den *absoluten Orientierungen* und auf den *inkrementellen Rotationen* einmal mit Schrittweite $h = 1$ und einmal mit Schrittweite $h = 3$ durchgeführt. Hierbei wurden fünf verschiedene Datensätze untersucht. Drei davon („Opportunity“, „QBGR“ und „6DMG“) sind im Internet frei verfügbar, einer wurde von uns selbst gemessen („JGU_Numbers“) und einer synthetisch erzeugt („qCBF“). In den Tabellen 2.3, 2.4 und 2.5 sind noch einmal die Klassifikations-Fehler unter den verschiedenen Abstandsmaßen zum Vergleich gegenüber gestellt. Für jeden Wert von $r_{\text{test}} \in \{0.1, 0.5, 0.9\}$ wurden hierbei 100 verschiedene stratifizierte Aufteilungen des jeweiligen Datensatzes in Test- und Trainings-Daten gewählt und dann Mittelwert und Standard-Fehler für den Klassifikations-Fehler bestimmt. Die in grüner Schrift dargestellten Werte sind die besten (niedrigsten) Klassifikations-Fehler für das jeweilige *Zeitreihen*-Abstandsmaß (Lockstep, cDTW, ssDTW). Die zusätzlich grün hinterlegten Werte sind die besten (niedrigsten) insgesamt für den entsprechenden Teil-Datensatz (für die jeweils links oben angegebenen Werte von r_{test} und h).

Vergleich: Datensätze

Von allen Datensätzen lieferte der **Opportunity-Datensatz** die schlechtesten Klassifikations-Ergebnisse. Bei diesem Datensatz waren die Bewegungen am ungenausten definiert. Jeder Proband konnte selbst bestimmen, wie die konkreten Bewegungs-Abläufe z.B. beim Öffnen und Schließen des Kühlschranks, beim Trinken aus einer Tasse oder dem Reinigen des Tisches durchgeführt wurden bzw. wie die genaue Hand- und Armhaltung bei der Durchführung der jeweiligen Bewegungen gewählt wurde. Neben diesen recht ungenau definierten Bewegungsmustern gab es auch sehr ähnliche Klassen, wie z.B. das Öffnen und Schließen der drei direkt übereinander angebrachten Schubladen, welche dementsprechend oft untereinander verwechselt wurden. Da die Aktionen von allen Probanden hauptsächlich mit der rechten Hand durchgeführt wurden, sind die Klassifikations-Ergebnisse der Sensoren am rechten Arm auch erwartungsgemäß besser als bei den Sensoren am linken Arm. Der Sensor am Rücken lieferte jedoch insgesamt die besten Ergebnisse für diesen Datensatz.

Der **QBGR-Datensatz** wurde unterteilt in Sensoren am aktiven (bewegten) Arm und Sensoren am inaktiven (still gehaltenen) Arm, sowie einen Sensor am unteren Bauch. Die einzelnen Klassen, die jeweils eine 4- bis 6-malige Wiederholung einer Grundbewegung (Sport-Übung) darstellen (siehe Kapitel 1), konnten am aktiven Arm sehr gut unterschieden werden. Die Klassifikations-Ergebnisse für den stillgehaltenen inaktiven Arm sind – wie zu erwarten – dagegen deutlich schlechter. Dennoch ist eine Unterscheidung grundsätzlich

auch hier möglich, aber verständlicherweise mit wesentlich höheren Klassifikations-Fehlern. Die Klassifikation anhand des Sensors am Bauch funktioniert auch verhältnismäßig gut, wenn man bedenkt, dass die Übungen im Wesentlichen mit den Armen durchgeführt wurden. Die Rotationen des Oberkörpers lassen eine Klassifikation der Bewegungen offenbar ebenfalls zu.

Auf dem **6DMG-Datensatz** wurden insgesamt (bei Verwendung der absoluten Orientierungen) die geringsten Klassifikations-Fehler erreicht. Obwohl dieser Datensatz die meisten Klassen (insgesamt 20) enthält, war die Klassifikation hier nahezu fehlerfrei. Dieser Datensatz zeigt deutlich, dass sehr klar definierte Bewegungen eine ideale Voraussetzung für das hier angewendete Verfahren der 1NN-Klassifikation darstellen. Obwohl in diesem Datensatz zum Teil sehr ähnliche Gesten (Klassen) enthalten waren, wurden diese sehr sauber und dadurch immer noch gut voneinander unterscheidbar durchgeführt.

Auch bei dem von uns selbst erzeugten **JGU_Numbers-Datensatz** wurden ebenfalls gute Klassifikations-Ergebnisse erzielt. Die meisten Verwechslungen traten hierbei bei Ziffern auf, die jeweils kreisförmige Elemente enthalten (Ziffern „0“, „6“, „8“ und „9“). Dennoch konnten die einzelnen Klassen dieses Datensatzes auch relativ gut voneinander unterschieden werden.

Beim synthetisch generierten **qCBF-Datensatz** waren die Klassifikations-Ergebnisse auf den absoluten Orientierungen relativ schlecht. Durch die Verwendung der inkrementellen Rotationen konnten die Ergebnisse hierbei signifikant verbessert werden: Die Klassifikation ist dabei fast fehlerfrei möglich.

Vergleich: Zeitreihen-Abstandsmaße

Im Allgemeinen ist **cDTW** das beste untersuchte *Zeitreihen*-Abstandsmaß, welches gegenüber Lockstep und ssDTW in fast allen Fällen bessere Ergebnisse lieferte. Dies ist auch in den nachfolgenden Tabellen deutlich zu sehen. **Lockstep** war hierbei immer etwas schlechter als cDTW. Das bei cDTW per Leave-One-Out-Cross-Validation (LOOCV) auf den Trainings-Daten gelernte Warping-Fenster war hierbei meist recht klein (in der Regel $W \leq$ 5% der Zeitreihenlänge), wodurch auch die Klassifikations-Ergebnisse von Lockstep und cDTW ähnlich ausfielen. Das **ssDTW**-Abstandsmaß mit seiner *lokalen Alignierung* stellte sich – im Vergleich zu Lockstep und cDTW mit jeweils *globaler Alignierung* – als das am schlechtesten für die Klassifikation per 1NN geeignete *Zeitreihen*-Abstandsmaß heraus, da einige Bewegungen bzw. Gesten mit Teilbewegungen anderer Bewegungen bzw. Gesten verwechselt wurden.

Vergleich: punktweise Abstandsmaße

Bezüglich der punktweisen Abstandsmaße ist der Unterschied in der Klassifikations-Genauigkeit nicht so groß wie zwischen den verschiedenen Zeitreihen-Abstandsmaßen. Dennoch stellt sich das (physikalisch motivierte) geodätische Abstandsmaß **„quat_geodesic“** in den meisten Fällen als das beste heraus. Besonders bei Verwendung der inkrementellen Rotationen (siehe Tabellen 2.4 und 2.5) ist „quat_geodesic“ deutlich besser als die anderen beiden *punktweisen* Abstandsmaße. Die Ergebnisse für **„quat_Euclidean“** und **„mat_Frobenius“** sind jeweils sehr ähnlich zueinander. Keines dieser beiden Abstandsmaße schneidet signifikant besser oder schlechter als das andere ab.

Vergleich: absolute Orientierungen vs. inkrementelle Rotationen

Die Klassifikation anhand der **inkrementellen Rotationen** lieferte im Vergleich zur Klassifikation anhand der (originalen) **absoluten Orientierungen** nur teilweise Verbesserungen. Insbesondere beim Opportunity-Datensatz wurden die Klassifikations-Ergebnisse dabei sogar drastisch schlechter, da auf diesem Datensatz die Bewegungen als inkrementelle Rotationen fast nicht unterscheidbar waren. Die Ursache dafür ist, dass aufgrund der im Vergleich zur Messfrequenz verhältnismäßig langsamen Durchführung der einzelnen Bewegungen die jeweiligen inkrementellen Rotationen zwischen zwei Zeitpunkten alle nur minimal von der „Null-Rotation“ ($q_{\mathbb{1}} = (1,0,0,0)^\top$) abweichen und die meisten Shapes der inkrementellen Rotationen damit fast identisch sind. Für den QBGR-Datensatz gab es bei manchen Sensoren Verbesserungen. Auf dem 6DMG-Datensatz und dem JGU_Numbers-Datensatz konnten durch die Verwendung der inkrementellen Rotationen leichte Verbesserungen der Klassifikations-Ergebnisse erreicht werden. Gerade für große Werte von r_{test}, bei denen der Trainings-Datensatz klein ist, lieferte die Verwendung der inkrementellen Rotationen die höchste Verbesserung. Beim synthetischen qCBF-Datensatz konnten die Klassifikations-Ergebnisse unter Verwendung der inkrementellen Rotationen signifikant verbessert werden. Hier zeigt sich die Stärke dieses Verfahrens: Durch die Koordinatensystem-unabhängige Darstellung konnten die inkrementellen CBF-Bewegungen um die (relative) z-Achse sehr gut unterschieden werden.

Datensatz	Lockstep			cDTW			ssDTW		
(absolut, $r_{\text{test}} = 0.1$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	17.72 ± 0.22	18.58 ± 0.21	18.50 ± 0.21	14.69 ± 0.20	15.84 ± 0.20	15.78 ± 0.20	26.51 ± 0.23	28.18 ± 0.24	28.15 ± 0.24
Opportunity RUA	14.18 ± 0.20	14.63 ± 0.20	14.56 ± 0.20	13.21 ± 0.20	13.24 ± 0.19	13.25 ± 0.19	24.04 ± 0.24	25.39 ± 0.23	25.35 ± 0.23
Opportunity BACK	13.20 ± 0.19	13.18 ± 0.22	13.17 ± 0.22	12.24 ± 0.20	11.87 ± 0.22	11.82 ± 0.22	27.48 ± 0.24	28.71 ± 0.24	28.75 ± 0.24
Opportunity LLA	25.16 ± 0.23	25.29 ± 0.21	25.20 ± 0.21	22.00 ± 0.23	22.16 ± 0.22	21.98 ± 0.23	42.20 ± 0.24	44.29 ± 0.25	44.29 ± 0.25
Opportunity LUA	19.25 ± 0.21	19.64 ± 0.22	19.60 ± 0.22	16.82 ± 0.21	17.38 ± 0.20	17.38 ± 0.20	34.00 ± 0.26	34.36 ± 0.26	34.29 ± 0.26
QBGR S15_A	2.78 ± 0.18	3.14 ± 0.23	3.05 ± 0.20	0.50 ± 0.08	1.20 ± 0.14	1.19 ± 0.14	1.17 ± 0.13	1.83 ± 0.16	1.83 ± 0.16
QBGR S16_A	1.36 ± 0.14	1.98 ± 0.16	1.97 ± 0.16	0.03 ± 0.02	0.18 ± 0.05	0.17 ± 0.05	3.14 ± 0.17	3.14 ± 0.17	2.98 ± 0.16
QBGR S17_A	1.09 ± 0.13	2.24 ± 0.20	1.74 ± 0.18	0.21 ± 0.07	0.20 ± 0.06	0.20 ± 0.06	4.36 ± 0.24	4.36 ± 0.25	4.35 ± 0.25
QBGR S18_A	1.53 ± 0.14	1.74 ± 0.16	1.52 ± 0.15	0.36 ± 0.07	0.44 ± 0.08	0.41 ± 0.08	6.45 ± 0.26	7.29 ± 0.29	7.29 ± 0.29
QBGR S19	4.47 ± 0.19	5.69 ± 0.22	5.71 ± 0.22	2.67 ± 0.15	2.61 ± 0.15	2.66 ± 0.15	15.71 ± 0.33	16.82 ± 0.34	16.82 ± 0.34
QBGR S15_I	23.11 ± 0.43	24.42 ± 0.46	24.42 ± 0.46	17.02 ± 0.45	18.17 ± 0.46	18.03 ± 0.46	34.11 ± 0.45	34.47 ± 0.47	34.47 ± 0.47
QBGR S16_I	6.29 ± 0.29	6.91 ± 0.28	6.91 ± 0.28	4.18 ± 0.25	5.03 ± 0.27	5.03 ± 0.27	14.03 ± 0.38	15.29 ± 0.42	15.29 ± 0.42
QBGR S17_I	13.36 ± 0.44	14.64 ± 0.45	14.64 ± 0.45	10.80 ± 0.39	10.20 ± 0.39	10.20 ± 0.39	18.56 ± 0.47	19.29 ± 0.48	19.29 ± 0.48
QBGR S18_I	17.98 ± 0.45	19.61 ± 0.47	19.62 ± 0.47	13.61 ± 0.44	15.52 ± 0.43	15.52 ± 0.43	26.64 ± 0.56	28.70 ± 0.58	28.70 ± 0.58
6DMG LR	0.63 ± 0.03	0.63 ± 0.03	0.62 ± 0.03	0.30 ± 0.02	0.30 ± 0.02	0.30 ± 0.02	7.30 ± 0.10	7.40 ± 0.10	7.41 ± 0.10
JGU_Numbers RH	1.76 ± 0.09	1.83 ± 0.09	1.83 ± 0.09	1.23 ± 0.08	1.37 ± 0.08	1.37 ± 0.08	7.73 ± 0.19	7.40 ± 0.17	7.40 ± 0.17
qCBF A	12.49 ± 0.25	11.08 ± 0.24	9.52 ± 0.22	10.73 ± 0.23	7.71 ± 0.20	7.51 ± 0.20	12.31 ± 0.22	6.67 ± 0.19	6.98 ± 0.19

Datensatz	Lockstep			cDTW			ssDTW		
(absolut, $r_{\text{test}} = 0.5$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	20.45 ± 0.09	21.78 ± 0.09	21.65 ± 0.09	17.67 ± 0.09	18.82 ± 0.09	18.79 ± 0.09	28.58 ± 0.10	30.07 ± 0.11	30.05 ± 0.10
Opportunity RUA	17.06 ± 0.09	17.49 ± 0.09	17.43 ± 0.09	15.36 ± 0.08	15.58 ± 0.09	15.52 ± 0.08	26.40 ± 0.12	27.40 ± 0.14	27.39 ± 0.14
Opportunity BACK	15.86 ± 0.09	15.91 ± 0.09	15.89 ± 0.09	14.47 ± 0.08	14.38 ± 0.08	14.35 ± 0.08	30.29 ± 0.12	30.98 ± 0.12	30.99 ± 0.12
Opportunity LLA	28.31 ± 0.10	28.80 ± 0.11	28.76 ± 0.11	25.22 ± 0.09	25.48 ± 0.10	25.41 ± 0.10	44.17 ± 0.13	46.15 ± 0.14	46.13 ± 0.14
Opportunity LUA	22.03 ± 0.09	22.49 ± 0.09	22.42 ± 0.09	19.73 ± 0.09	20.10 ± 0.09	20.09 ± 0.09	36.47 ± 0.12	37.15 ± 0.13	37.11 ± 0.13
QBGR S15_A	4.62 ± 0.12	5.59 ± 0.13	5.21 ± 0.13	1.46 ± 0.09	1.86 ± 0.09	1.78 ± 0.09	2.47 ± 0.11	3.12 ± 0.12	2.99 ± 0.12
QBGR S16_A	3.27 ± 0.15	4.03 ± 0.16	3.94 ± 0.16	0.94 ± 0.11	0.90 ± 0.11	0.93 ± 0.11	4.78 ± 0.14	4.81 ± 0.14	4.78 ± 0.14
QBGR S17_A	3.93 ± 0.14	5.30 ± 0.16	4.84 ± 0.15	0.85 ± 0.10	0.83 ± 0.09	0.85 ± 0.09	4.77 ± 0.14	5.04 ± 0.14	4.98 ± 0.15
QBGR S18_A	3.93 ± 0.12	4.91 ± 0.15	4.64 ± 0.14	0.92 ± 0.10	1.06 ± 0.10	0.98 ± 0.10	7.66 ± 0.14	8.62 ± 0.15	8.56 ± 0.15
QBGR S19	7.03 ± 0.12	8.41 ± 0.13	8.45 ± 0.13	3.88 ± 0.09	3.84 ± 0.09	3.87 ± 0.09	16.36 ± 0.15	17.74 ± 0.15	17.74 ± 0.15
QBGR S15_I	27.44 ± 0.24	28.29 ± 0.23	28.31 ± 0.23	21.51 ± 0.22	22.20 ± 0.22	22.20 ± 0.22	36.30 ± 0.20	36.66 ± 0.20	36.66 ± 0.20
QBGR S16_I	10.19 ± 0.20	11.12 ± 0.20	11.04 ± 0.20	7.85 ± 0.17	8.11 ± 0.17	8.06 ± 0.17	16.07 ± 0.20	17.57 ± 0.21	17.55 ± 0.21
QBGR S17_I	18.05 ± 0.18	18.84 ± 0.18	18.75 ± 0.18	14.77 ± 0.17	14.60 ± 0.18	14.60 ± 0.18	21.12 ± 0.22	21.85 ± 0.21	21.86 ± 0.21
QBGR S18_I	22.44 ± 0.20	23.50 ± 0.20	23.47 ± 0.20	18.93 ± 0.21	20.11 ± 0.20	20.12 ± 0.20	29.75 ± 0.20	31.43 ± 0.19	31.43 ± 0.19
6DMG LR	1.09 ± 0.02	1.08 ± 0.02	1.08 ± 0.02	0.56 ± 0.02	0.50 ± 0.02	0.50 ± 0.02	9.01 ± 0.05	9.04 ± 0.05	9.04 ± 0.05
JGU_Numbers RH	4.18 ± 0.07	4.01 ± 0.07	4.01 ± 0.07	3.24 ± 0.07	3.19 ± 0.06	3.19 ± 0.06	11.97 ± 0.10	11.56 ± 0.10	11.56 ± 0.10
qCBF A	16.38 ± 0.12	15.09 ± 0.11	13.05 ± 0.11	14.16 ± 0.11	10.31 ± 0.09	10.58 ± 0.10	16.56 ± 0.10	9.89 ± 0.10	10.50 ± 0.10

Datensatz	Lockstep			cDTW			ssDTW		
(absolut, $r_{\text{test}} = 0.9$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	33.73 ± 0.14	35.59 ± 0.14	35.27 ± 0.14	31.14 ± 0.16	32.60 ± 0.15	32.42 ± 0.15	40.05 ± 0.14	41.38 ± 0.15	41.30 ± 0.15
Opportunity RUA	29.37 ± 0.11	30.15 ± 0.11	30.03 ± 0.11	27.48 ± 0.14	28.03 ± 0.13	28.00 ± 0.13	38.22 ± 0.19	38.84 ± 0.20	38.79 ± 0.20
Opportunity BACK	29.43 ± 0.15	29.77 ± 0.15	29.72 ± 0.15	28.14 ± 0.16	28.28 ± 0.17	28.26 ± 0.16	43.77 ± 0.18	44.27 ± 0.19	44.26 ± 0.19
Opportunity LLA	43.94 ± 0.15	44.57 ± 0.15	44.50 ± 0.15	41.51 ± 0.17	41.88 ± 0.17	41.79 ± 0.17	55.37 ± 0.17	57.00 ± 0.18	56.98 ± 0.18
Opportunity LUA	36.90 ± 0.16	37.21 ± 0.16	37.16 ± 0.16	34.98 ± 0.17	35.17 ± 0.17	35.12 ± 0.18	49.38 ± 0.17	50.10 ± 0.18	50.08 ± 0.18
QBGR S15_A	24.35 ± 0.32	25.64 ± 0.33	25.05 ± 0.32	18.97 ± 0.40	18.90 ± 0.40	18.68 ± 0.40	22.17 ± 0.31	22.26 ± 0.31	22.15 ± 0.31
QBGR S16_A	29.77 ± 0.38	30.60 ± 0.38	30.45 ± 0.38	25.01 ± 0.45	24.47 ± 0.46	24.50 ± 0.46	28.46 ± 0.38	28.38 ± 0.38	28.35 ± 0.38
QBGR S17_A	33.80 ± 0.31	35.03 ± 0.30	34.61 ± 0.30	28.49 ± 0.39	27.49 ± 0.37	27.64 ± 0.37	29.82 ± 0.37	29.65 ± 0.37	29.64 ± 0.37
QBGR S18_A	30.03 ± 0.38	31.59 ± 0.38	30.83 ± 0.37	21.55 ± 0.44	21.61 ± 0.42	21.46 ± 0.42	29.64 ± 0.34	29.81 ± 0.33	29.74 ± 0.33
QBGR S19	29.35 ± 0.26	30.40 ± 0.26	30.39 ± 0.26	24.33 ± 0.30	23.91 ± 0.29	23.91 ± 0.29	30.51 ± 0.27	31.56 ± 0.27	31.55 ± 0.27
QBGR S15_I	50.04 ± 0.30	51.01 ± 0.30	50.87 ± 0.31	46.62 ± 0.34	46.17 ± 0.33	46.14 ± 0.32	49.98 ± 0.31	50.18 ± 0.31	50.18 ± 0.31
QBGR S16_I	42.77 ± 0.36	43.36 ± 0.36	43.29 ± 0.35	39.56 ± 0.38	39.39 ± 0.37	39.29 ± 0.38	40.64 ± 0.35	41.83 ± 0.35	41.79 ± 0.35
QBGR S17_I	46.71 ± 0.30	47.43 ± 0.30	47.36 ± 0.30	42.11 ± 0.38	41.83 ± 0.38	41.82 ± 0.38	42.76 ± 0.34	43.19 ± 0.33	43.20 ± 0.33
QBGR S18_I	47.69 ± 0.31	48.27 ± 0.31	48.18 ± 0.31	43.03 ± 0.35	42.80 ± 0.34	42.85 ± 0.34	48.01 ± 0.30	48.57 ± 0.30	48.57 ± 0.30
6DMG LR	6.38 ± 0.06	6.11 ± 0.06	6.13 ± 0.06	4.57 ± 0.06	4.25 ± 0.05	4.25 ± 0.05	20.13 ± 0.11	19.62 ± 0.11	19.62 ± 0.11
JGU_Numbers RH	37.72 ± 0.22	36.68 ± 0.23	36.68 ± 0.23	37.20 ± 0.23	36.16 ± 0.23	36.18 ± 0.23	48.33 ± 0.19	47.45 ± 0.20	47.45 ± 0.20
qCBF A	29.33 ± 0.15	27.03 ± 0.17	25.86 ± 0.14	27.25 ± 0.15	22.83 ± 0.16	23.62 ± 0.16	29.86 ± 0.13	22.74 ± 0.12	23.70 ± 0.12

Tabelle 2.3: Klassifikation: Ergebnisse der **absoluten Orientierungen**. Dargestellt sind jeweils Mittelwert und Standard-Fehler des **Klassifikations-Fehlers** über 100 stratifizierte Aufteilungen des jeweiligen Datensatzes. Alle Werte sind in % auf zwei Nachkommastellen gerundet angegeben. Die in grüner Schrift dargestellten Werte sind die besten (niedrigsten) Klassifikations-Fehler für das jeweilige *Zeitreihen*-Abstandsmaß (Lockstep, cDTW, ssDTW). Die zusätzlich grün hinterlegten Werte sind die besten (niedrigsten) insgesamt für den entsprechenden Datensatz. **[Oben]**: $r_{\text{test}} = 0.1$, **[Mitte]**: $r_{\text{test}} = 0.5$, **[Unten]**: $r_{\text{test}} = 0.9$.

Datensatz	Lockstep			cDTW			ssDTW		
($h = 1$, $r_{\text{test}} = 0.1$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	36.06 ± 0.26	36.79 ± 0.27	36.82 ± 0.27	23.40 ± 0.24	24.70 ± 0.22	24.71 ± 0.22	79.83 ± 0.16	82.60 ± 0.14	82.60 ± 0.14
Opportunity RUA	32.55 ± 0.28	34.73 ± 0.30	34.73 ± 0.30	19.73 ± 0.23	20.00 ± 0.22	20.00 ± 0.22	76.91 ± 0.19	79.69 ± 0.18	79.69 ± 0.18
Opportunity BACK	30.08 ± 0.26	31.08 ± 0.26	31.08 ± 0.26	19.71 ± 0.26	18.73 ± 0.24	18.73 ± 0.24	76.25 ± 0.19	77.75 ± 0.18	77.75 ± 0.18
Opportunity LLA	45.64 ± 0.28	47.35 ± 0.30	47.35 ± 0.30	31.45 ± 0.25	32.46 ± 0.27	32.46 ± 0.27	83.48 ± 0.13	85.18 ± 0.12	85.18 ± 0.12
Opportunity LUA	38.02 ± 0.28	39.65 ± 0.29	39.65 ± 0.29	25.68 ± 0.24	25.55 ± 0.25	25.55 ± 0.25	82.04 ± 0.16	83.75 ± 0.13	83.75 ± 0.13
QBGR S15_A	15.61 ± 0.42	17.39 ± 0.46	17.31 ± 0.46	1.11 ± 0.13	2.61 ± 0.18	2.61 ± 0.18	42.16 ± 0.52	46.48 ± 0.51	46.48 ± 0.51
QBGR S16_A	11.64 ± 0.38	15.11 ± 0.40	15.08 ± 0.40	0.21 ± 0.07	0.35 ± 0.07	0.17 ± 0.06	43.17 ± 0.45	43.68 ± 0.44	43.68 ± 0.44
QBGR S17_A	14.45 ± 0.42	18.06 ± 0.45	18.02 ± 0.45	0.05 ± 0.03	0.35 ± 0.09	0.33 ± 0.08	40.86 ± 0.45	41.68 ± 0.45	41.68 ± 0.45
QBGR S18_A	14.50 ± 0.36	18.94 ± 0.43	18.94 ± 0.43	0.09 ± 0.04	0.62 ± 0.09	0.62 ± 0.09	51.47 ± 0.43	53.48 ± 0.42	53.48 ± 0.42
QBGR S19	36.31 ± 0.36	38.73 ± 0.36	38.68 ± 0.36	9.82 ± 0.27	12.97 ± 0.32	12.80 ± 0.32	73.44 ± 0.25	74.20 ± 0.24	74.20 ± 0.24
QBGR S15_I	38.27 ± 0.54	38.28 ± 0.49	38.23 ± 0.49	17.05 ± 0.39	18.95 ± 0.44	18.80 ± 0.43	68.39 ± 0.50	68.48 ± 0.52	68.48 ± 0.52
QBGR S16_I	37.83 ± 0.49	38.15 ± 0.55	37.83 ± 0.55	13.53 ± 0.41	16.38 ± 0.45	16.09 ± 0.46	66.27 ± 0.46	67.38 ± 0.46	67.38 ± 0.46
QBGR S17_I	37.79 ± 0.54	40.26 ± 0.51	40.26 ± 0.51	15.36 ± 0.40	17.77 ± 0.44	17.77 ± 0.44	68.59 ± 0.41	70.03 ± 0.41	70.03 ± 0.41
QBGR S18_I	41.45 ± 0.49	43.76 ± 0.47	43.76 ± 0.47	17.74 ± 0.40	22.21 ± 0.46	22.20 ± 0.46	64.27 ± 0.45	65.26 ± 0.44	65.26 ± 0.44
6DMG LR	1.31 ± 0.05	1.21 ± 0.05	1.21 ± 0.05	0.41 ± 0.02	0.36 ± 0.02	0.36 ± 0.02	76.70 ± 0.15	76.74 ± 0.15	76.74 ± 0.15
JGU_Numbers RH	6.15 ± 0.15	7.17 ± 0.17	7.17 ± 0.17	1.13 ± 0.07	3.10 ± 0.12	3.10 ± 0.12	53.06 ± 0.34	52.91 ± 0.31	52.91 ± 0.31
qCBF A	24.88 ± 0.20	21.97 ± 0.21	22.32 ± 0.21	0.02 ± 0.01	0.05 ± 0.02	0.06 ± 0.02	3.95 ± 0.16	2.74 ± 0.13	2.74 ± 0.13

Datensatz	Lockstep			cDTW			ssDTW		
($h = 1$, $r_{\text{test}} = 0.5$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	40.10 ± 0.10	41.38 ± 0.11	41.39 ± 0.11	26.14 ± 0.12	27.65 ± 0.11	27.64 ± 0.11	79.65 ± 0.11	81.90 ± 0.22	81.90 ± 0.22
Opportunity RUA	36.61 ± 0.11	38.96 ± 0.11	38.96 ± 0.11	22.82 ± 0.09	23.45 ± 0.10	23.45 ± 0.10	76.56 ± 0.11	78.67 ± 0.22	78.67 ± 0.22
Opportunity BACK	33.14 ± 0.10	34.38 ± 0.10	34.38 ± 0.10	22.96 ± 0.11	22.39 ± 0.10	22.39 ± 0.10	73.81 ± 0.21	76.80 ± 0.15	76.80 ± 0.15
Opportunity LLA	50.64 ± 0.11	52.32 ± 0.11	52.31 ± 0.11	35.46 ± 0.11	36.85 ± 0.12	36.86 ± 0.12	82.85 ± 0.13	84.96 ± 0.09	84.96 ± 0.09
Opportunity LUA	42.57 ± 0.11	44.20 ± 0.12	44.20 ± 0.12	29.40 ± 0.11	29.35 ± 0.10	29.35 ± 0.10	80.45 ± 0.23	82.67 ± 0.18	82.67 ± 0.18
QBGR S15_A	21.87 ± 0.22	24.45 ± 0.21	24.38 ± 0.21	2.82 ± 0.13	4.73 ± 0.14	4.72 ± 0.15	42.76 ± 0.58	47.64 ± 0.52	47.63 ± 0.52
QBGR S16_A	18.99 ± 0.19	22.91 ± 0.21	22.80 ± 0.21	1.04 ± 0.09	1.51 ± 0.10	1.37 ± 0.10	42.35 ± 0.30	42.88 ± 0.29	42.88 ± 0.29
QBGR S17_A	20.60 ± 0.22	24.55 ± 0.23	24.44 ± 0.24	0.98 ± 0.08	1.78 ± 0.10	1.72 ± 0.10	40.13 ± 0.35	41.30 ± 0.35	41.30 ± 0.35
QBGR S18_A	21.56 ± 0.23	26.00 ± 0.22	25.98 ± 0.22	1.23 ± 0.10	2.15 ± 0.12	2.13 ± 0.12	50.62 ± 0.35	53.06 ± 0.35	53.06 ± 0.35
QBGR S19	43.23 ± 0.16	45.22 ± 0.16	45.18 ± 0.16	14.36 ± 0.15	17.47 ± 0.18	17.40 ± 0.18	73.31 ± 0.11	73.57 ± 0.12	73.57 ± 0.12
QBGR S15_I	44.21 ± 0.20	44.36 ± 0.20	44.29 ± 0.20	22.21 ± 0.25	25.01 ± 0.26	25.05 ± 0.25	67.54 ± 0.20	67.78 ± 0.21	67.78 ± 0.21
QBGR S16_I	41.63 ± 0.21	43.01 ± 0.21	43.04 ± 0.21	17.14 ± 0.21	20.82 ± 0.22	20.75 ± 0.21	66.02 ± 0.19	67.75 ± 0.20	67.75 ± 0.20
QBGR S17_I	44.36 ± 0.21	47.08 ± 0.22	47.08 ± 0.22	20.71 ± 0.20	23.90 ± 0.25	23.90 ± 0.25	68.18 ± 0.20	69.71 ± 0.18	69.71 ± 0.18
QBGR S18_I	46.36 ± 0.20	48.26 ± 0.19	48.26 ± 0.19	22.88 ± 0.20	27.39 ± 0.21	27.40 ± 0.21	64.18 ± 0.19	66.06 ± 0.18	66.06 ± 0.18
6DMG LR	2.20 ± 0.03	2.32 ± 0.03	2.32 ± 0.03	0.58 ± 0.01	0.61 ± 0.02	0.61 ± 0.02	76.37 ± 0.10	76.56 ± 0.10	76.56 ± 0.10
JGU_Numbers RH	9.57 ± 0.10	10.61 ± 0.10	10.61 ± 0.10	2.33 ± 0.06	4.72 ± 0.07	4.72 ± 0.07	54.08 ± 0.19	55.12 ± 0.25	55.12 ± 0.25
qCBF A	28.58 ± 0.09	27.80 ± 0.11	28.07 ± 0.11	0.14 ± 0.02	0.10 ± 0.01	0.10 ± 0.01	4.45 ± 0.08	3.37 ± 0.06	3.36 ± 0.06

Datensatz	Lockstep			cDTW			ssDTW		
($h = 1$, $r_{\text{test}} = 0.9$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	56.23 ± 0.15	57.72 ± 0.13	57.72 ± 0.13	38.72 ± 0.16	40.35 ± 0.15	40.33 ± 0.15	79.28 ± 0.18	80.55 ± 0.26	80.55 ± 0.26
Opportunity RUA	52.91 ± 0.14	55.80 ± 0.13	55.80 ± 0.13	35.96 ± 0.15	37.35 ± 0.17	37.34 ± 0.17	76.29 ± 0.17	77.67 ± 0.25	77.67 ± 0.25
Opportunity BACK	48.48 ± 0.14	49.63 ± 0.13	49.62 ± 0.13	37.74 ± 0.16	37.90 ± 0.15	37.90 ± 0.15	72.99 ± 0.31	76.56 ± 0.32	76.56 ± 0.32
Opportunity LLA	65.74 ± 0.12	67.28 ± 0.11	67.28 ± 0.11	52.59 ± 0.13	53.91 ± 0.14	53.91 ± 0.14	82.87 ± 0.16	86.06 ± 0.11	86.05 ± 0.11
Opportunity LUA	57.53 ± 0.14	58.93 ± 0.13	58.93 ± 0.13	44.45 ± 0.15	45.09 ± 0.16	45.09 ± 0.16	80.41 ± 0.20	83.65 ± 0.18	83.65 ± 0.18
QBGR S15_A	46.03 ± 0.30	48.54 ± 0.29	48.49 ± 0.29	24.02 ± 0.38	26.57 ± 0.37	26.53 ± 0.37	46.80 ± 0.75	51.21 ± 0.76	51.19 ± 0.76
QBGR S16_A	45.64 ± 0.26	48.35 ± 0.24	48.32 ± 0.24	22.98 ± 0.37	23.94 ± 0.33	23.88 ± 0.33	48.22 ± 0.46	49.76 ± 0.43	49.75 ± 0.44
QBGR S17_A	46.21 ± 0.31	49.52 ± 0.31	49.50 ± 0.31	23.02 ± 0.38	24.65 ± 0.38	24.61 ± 0.38	47.25 ± 0.51	48.70 ± 0.54	48.69 ± 0.54
QBGR S18_A	47.59 ± 0.25	50.50 ± 0.29	50.45 ± 0.29	25.28 ± 0.39	26.78 ± 0.38	26.71 ± 0.38	53.95 ± 0.60	56.80 ± 0.58	56.79 ± 0.58
QBGR S19	59.25 ± 0.15	60.34 ± 0.16	60.32 ± 0.16	39.06 ± 0.26	41.45 ± 0.24	41.47 ± 0.24	74.15 ± 0.19	74.10 ± 0.19	74.10 ± 0.19
QBGR S15_I	58.50 ± 0.25	59.20 ± 0.26	59.22 ± 0.26	43.19 ± 0.35	45.87 ± 0.34	45.86 ± 0.33	70.15 ± 0.20	70.84 ± 0.20	70.84 ± 0.20
QBGR S16_I	56.85 ± 0.27	58.36 ± 0.27	58.42 ± 0.27	38.97 ± 0.35	42.19 ± 0.38	42.20 ± 0.37	69.62 ± 0.22	70.76 ± 0.22	70.76 ± 0.22
QBGR S17_I	57.71 ± 0.28	59.20 ± 0.27	59.19 ± 0.27	41.50 ± 0.37	43.41 ± 0.38	43.41 ± 0.38	69.58 ± 0.19	70.70 ± 0.21	70.70 ± 0.21
QBGR S18_I	59.91 ± 0.29	61.53 ± 0.31	61.53 ± 0.31	45.33 ± 0.41	48.50 ± 0.34	48.51 ± 0.34	67.73 ± 0.22	68.85 ± 0.21	68.85 ± 0.21
6DMG LR	8.67 ± 0.07	8.78 ± 0.07	8.78 ± 0.07	4.15 ± 0.06	4.09 ± 0.06	4.10 ± 0.06	75.56 ± 0.21	76.00 ± 0.23	76.00 ± 0.23
JGU_Numbers RH	27.42 ± 0.19	28.26 ± 0.19	28.26 ± 0.19	18.40 ± 0.19	20.73 ± 0.20	20.72 ± 0.20	65.71 ± 0.24	67.45 ± 0.26	67.45 ± 0.26
qCBF A	35.67 ± 0.12	43.74 ± 0.40	43.51 ± 0.41	0.77 ± 0.07	0.58 ± 0.05	0.56 ± 0.05	7.79 ± 0.14	7.48 ± 0.13	7.47 ± 0.13

Tabelle 2.4: Klassifikation: Ergebnisse der **inkrementellen Rotationen mit h = 1**. Dargestellt sind jeweils Mittelwert und Standard-Fehler des **Klassifikations-Fehlers** über 100 stratifizierte Aufteilungen des jeweiligen Datensatzes. Alle Werte sind in % auf zwei Nachkommastellen gerundet angegeben. **[Oben]**: $r_{\text{test}} = 0.1$, **[Mitte]**: $r_{\text{test}} = 0.5$, **[Unten]**: $r_{\text{test}} = 0.9$.

Datensatz	Lockstep			cDTW			ssDTW		
($h=3$, $r_{\text{test}}=0.1$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	33.71 ± 0.26	34.44 ± 0.25	34.39 ± 0.24	23.12 ± 0.23	25.12 ± 0.23	25.06 ± 0.23	79.63 ± 0.16	82.00 ± 0.13	82.04 ± 0.13
Opportunity RUA	30.51 ± 0.26	31.94 ± 0.28	31.93 ± 0.28	20.15 ± 0.22	20.44 ± 0.24	20.47 ± 0.24	75.37 ± 0.20	78.14 ± 0.17	78.14 ± 0.17
Opportunity BACK	30.09 ± 0.27	30.56 ± 0.28	30.56 ± 0.28	19.84 ± 0.25	19.75 ± 0.24	19.74 ± 0.23	72.34 ± 0.22	75.61 ± 0.21	75.61 ± 0.21
Opportunity LLA	43.87 ± 0.27	44.69 ± 0.31	44.70 ± 0.31	31.22 ± 0.24	32.04 ± 0.26	32.03 ± 0.27	82.41 ± 0.14	84.25 ± 0.11	84.25 ± 0.11
Opportunity LUA	36.71 ± 0.25	37.17 ± 0.27	37.17 ± 0.27	25.84 ± 0.25	26.11 ± 0.24	26.17 ± 0.24	81.09 ± 0.15	83.21 ± 0.15	83.21 ± 0.15
QBGR S15_A	15.28 ± 0.41	17.02 ± 0.46	16.84 ± 0.45	1.02 ± 0.13	3.67 ± 0.21	3.42 ± 0.19	22.45 ± 0.46	24.80 ± 0.50	24.72 ± 0.50
QBGR S16_A	8.91 ± 0.32	11.35 ± 0.36	11.23 ± 0.36	0.20 ± 0.06	0.27 ± 0.07	0.27 ± 0.07	29.73 ± 0.45	32.33 ± 0.48	32.33 ± 0.48
QBGR S17_A	12.32 ± 0.41	14.85 ± 0.44	14.48 ± 0.43	0.27 ± 0.07	0.67 ± 0.11	0.71 ± 0.11	33.52 ± 0.48	36.12 ± 0.45	36.12 ± 0.45
QBGR S18_A	13.65 ± 0.39	17.18 ± 0.42	16.83 ± 0.42	0.12 ± 0.04	1.03 ± 0.11	0.71 ± 0.09	35.98 ± 0.47	38.91 ± 0.46	38.77 ± 0.45
QBGR S19	33.39 ± 0.36	35.00 ± 0.38	34.86 ± 0.37	9.98 ± 0.24	14.22 ± 0.29	14.16 ± 0.29	72.38 ± 0.25	72.37 ± 0.26	72.37 ± 0.26
QBGR S15_I	36.55 ± 0.51	36.22 ± 0.49	35.94 ± 0.48	16.23 ± 0.40	18.61 ± 0.43	18.66 ± 0.43	66.08 ± 0.47	66.08 ± 0.48	66.08 ± 0.48
QBGR S16_I	31.82 ± 0.50	33.15 ± 0.52	33.14 ± 0.53	11.64 ± 0.38	14.89 ± 0.45	14.53 ± 0.45	64.67 ± 0.40	65.30 ± 0.46	65.30 ± 0.46
QBGR S17_I	33.42 ± 0.49	36.33 ± 0.54	36.33 ± 0.54	15.29 ± 0.39	19.85 ± 0.44	19.82 ± 0.43	67.76 ± 0.38	68.42 ± 0.33	68.42 ± 0.33
QBGR S18_I	38.23 ± 0.49	39.86 ± 0.54	39.86 ± 0.54	17.21 ± 0.40	23.38 ± 0.43	23.41 ± 0.44	62.79 ± 0.43	64.08 ± 0.47	64.08 ± 0.47
6DMG LR	0.96 ± 0.04	0.89 ± 0.04	0.89 ± 0.04	0.29 ± 0.02	0.31 ± 0.02	0.31 ± 0.02	59.49 ± 0.19	61.60 ± 0.19	61.60 ± 0.19
JGU_Numbers RH	5.18 ± 0.13	5.99 ± 0.15	5.99 ± 0.15	0.95 ± 0.07	1.89 ± 0.09	1.89 ± 0.09	36.19 ± 0.36	39.40 ± 0.37	39.40 ± 0.37
qCBF A	5.27 ± 0.16	6.47 ± 0.18	6.73 ± 0.18	0.03 ± 0.01	0.04 ± 0.02	0.01 ± 0.01	0.66 ± 0.07	0.46 ± 0.06	0.46 ± 0.06

Datensatz	Lockstep			cDTW			ssDTW		
($h=3$, $r_{\text{test}}=0.5$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	37.73 ± 0.11	39.03 ± 0.11	38.98 ± 0.11	25.70 ± 0.11	27.42 ± 0.11	27.39 ± 0.11	78.89 ± 0.14	81.17 ± 0.24	81.18 ± 0.24
Opportunity RUA	34.53 ± 0.11	36.41 ± 0.11	36.39 ± 0.11	22.67 ± 0.10	23.46 ± 0.11	23.46 ± 0.11	74.46 ± 0.15	76.89 ± 0.26	76.89 ± 0.26
Opportunity BACK	32.72 ± 0.10	33.71 ± 0.10	33.71 ± 0.10	22.98 ± 0.10	22.87 ± 0.10	22.87 ± 0.10	70.57 ± 0.21	74.64 ± 0.16	74.64 ± 0.16
Opportunity LLA	48.71 ± 0.12	50.03 ± 0.13	50.02 ± 0.13	34.98 ± 0.10	36.55 ± 0.10	36.55 ± 0.11	81.78 ± 0.12	84.19 ± 0.08	84.19 ± 0.08
Opportunity LUA	40.99 ± 0.11	41.95 ± 0.11	41.95 ± 0.11	28.95 ± 0.10	29.51 ± 0.10	29.49 ± 0.10	79.41 ± 0.21	82.13 ± 0.18	82.13 ± 0.18
QBGR S15_A	20.87 ± 0.21	23.30 ± 0.21	23.00 ± 0.21	3.13 ± 0.13	5.56 ± 0.15	5.26 ± 0.14	24.69 ± 0.38	27.96 ± 0.42	27.82 ± 0.42
QBGR S16_A	16.11 ± 0.19	18.43 ± 0.19	18.24 ± 0.19	0.85 ± 0.08	1.31 ± 0.08	1.24 ± 0.08	31.63 ± 0.35	33.29 ± 0.39	33.25 ± 0.39
QBGR S17_A	18.13 ± 0.22	21.43 ± 0.23	21.04 ± 0.23	1.05 ± 0.09	2.08 ± 0.11	1.94 ± 0.11	31.84 ± 0.42	34.05 ± 0.49	33.95 ± 0.49
QBGR S18_A	20.69 ± 0.23	24.11 ± 0.22	23.64 ± 0.22	1.39 ± 0.10	2.51 ± 0.12	2.16 ± 0.12	36.22 ± 0.40	38.93 ± 0.41	38.78 ± 0.41
QBGR S19	40.34 ± 0.15	42.39 ± 0.15	42.31 ± 0.15	14.54 ± 0.15	18.85 ± 0.17	18.83 ± 0.17	72.17 ± 0.11	72.31 ± 0.12	72.31 ± 0.12
QBGR S15_I	42.15 ± 0.20	42.19 ± 0.21	42.08 ± 0.21	21.99 ± 0.24	24.67 ± 0.25	24.66 ± 0.25	66.01 ± 0.18	66.38 ± 0.18	66.38 ± 0.18
QBGR S16_I	37.09 ± 0.22	38.61 ± 0.21	38.57 ± 0.22	16.22 ± 0.21	20.06 ± 0.23	19.84 ± 0.23	64.43 ± 0.20	65.88 ± 0.22	65.88 ± 0.22
QBGR S17_I	39.58 ± 0.23	42.45 ± 0.23	42.45 ± 0.23	21.38 ± 0.20	24.48 ± 0.24	24.48 ± 0.24	67.38 ± 0.22	68.16 ± 0.19	68.15 ± 0.19
QBGR S18_I	43.84 ± 0.18	45.27 ± 0.18	45.27 ± 0.18	22.70 ± 0.19	27.78 ± 0.21	27.78 ± 0.21	63.42 ± 0.19	64.95 ± 0.19	64.95 ± 0.19
6DMG LR	1.71 ± 0.02	1.72 ± 0.03	1.73 ± 0.02	0.46 ± 0.01	0.50 ± 0.01	0.50 ± 0.01	59.64 ± 0.11	61.52 ± 0.12	61.51 ± 0.12
JGU_Numbers RH	8.44 ± 0.09	9.09 ± 0.09	9.09 ± 0.09	2.10 ± 0.05	3.21 ± 0.06	3.21 ± 0.06	38.92 ± 0.21	42.34 ± 0.28	42.34 ± 0.28
qCBF A	10.01 ± 0.10	13.07 ± 0.10	13.41 ± 0.10	0.10 ± 0.02	0.09 ± 0.01	0.09 ± 0.01	0.72 ± 0.03	0.69 ± 0.03	0.68 ± 0.03

Datensatz	Lockstep			cDTW			ssDTW		
($h=3$, $r_{\text{test}}=0.9$)	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.	quat_geo.	quat_Eucl.	mat_Frob.
Opportunity RLA	54.21 ± 0.14	55.69 ± 0.13	55.66 ± 0.13	38.12 ± 0.14	39.83 ± 0.15	39.81 ± 0.14	77.68 ± 0.23	79.39 ± 0.30	79.39 ± 0.30
Opportunity RUA	51.09 ± 0.13	53.58 ± 0.13	53.56 ± 0.13	35.41 ± 0.15	36.77 ± 0.14	36.75 ± 0.14	73.67 ± 0.20	75.19 ± 0.29	75.19 ± 0.29
Opportunity BACK	47.94 ± 0.14	49.15 ± 0.14	49.14 ± 0.14	37.64 ± 0.17	37.94 ± 0.16	37.93 ± 0.16	71.04 ± 0.33	75.08 ± 0.33	75.08 ± 0.33
Opportunity LLA	64.25 ± 0.12	65.79 ± 0.12	65.78 ± 0.12	51.90 ± 0.15	53.37 ± 0.14	53.36 ± 0.14	82.19 ± 0.15	85.53 ± 0.11	85.53 ± 0.11
Opportunity LUA	56.14 ± 0.14	57.39 ± 0.13	57.39 ± 0.13	44.03 ± 0.14	44.89 ± 0.16	44.88 ± 0.16	79.49 ± 0.20	83.17 ± 0.18	83.17 ± 0.18
QBGR S15_A	44.68 ± 0.30	46.86 ± 0.29	46.40 ± 0.29	24.05 ± 0.35	26.76 ± 0.41	26.35 ± 0.38	38.28 ± 0.60	41.81 ± 0.59	41.60 ± 0.59
QBGR S16_A	43.44 ± 0.28	45.27 ± 0.27	45.09 ± 0.28	22.85 ± 0.37	23.90 ± 0.37	23.67 ± 0.36	42.19 ± 0.46	43.74 ± 0.46	43.70 ± 0.47
QBGR S17_A	44.75 ± 0.31	47.49 ± 0.31	47.23 ± 0.32	22.85 ± 0.37	24.36 ± 0.40	24.25 ± 0.41	41.35 ± 0.44	43.33 ± 0.46	43.27 ± 0.46
QBGR S18_A	46.88 ± 0.25	49.29 ± 0.29	48.91 ± 0.29	25.41 ± 0.37	26.52 ± 0.36	26.29 ± 0.37	44.60 ± 0.60	47.26 ± 0.59	47.11 ± 0.59
QBGR S19	57.84 ± 0.17	59.08 ± 0.16	59.07 ± 0.16	39.40 ± 0.28	42.30 ± 0.22	42.26 ± 0.22	73.38 ± 0.17	73.43 ± 0.18	73.43 ± 0.18
QBGR S15_I	56.94 ± 0.26	57.70 ± 0.25	57.66 ± 0.25	42.96 ± 0.34	45.96 ± 0.35	45.88 ± 0.35	69.78 ± 0.18	70.58 ± 0.20	70.57 ± 0.20
QBGR S16_I	54.60 ± 0.29	56.00 ± 0.28	55.97 ± 0.28	38.55 ± 0.33	41.31 ± 0.31	41.16 ± 0.31	68.41 ± 0.24	70.04 ± 0.26	70.01 ± 0.26
QBGR S17_I	55.79 ± 0.29	57.30 ± 0.29	57.25 ± 0.29	41.39 ± 0.32	43.70 ± 0.35	43.64 ± 0.33	68.60 ± 0.20	69.74 ± 0.20	69.73 ± 0.20
QBGR S18_I	58.59 ± 0.30	59.97 ± 0.32	59.94 ± 0.32	44.30 ± 0.37	47.68 ± 0.36	47.58 ± 0.35	67.11 ± 0.23	68.37 ± 0.22	68.37 ± 0.23
6DMG LR	7.91 ± 0.06	7.78 ± 0.07	7.78 ± 0.07	4.02 ± 0.07	3.90 ± 0.06	3.90 ± 0.06	61.53 ± 0.25	63.87 ± 0.26	63.87 ± 0.26
JGU_Numbers RH	25.92 ± 0.17	26.08 ± 0.17	26.08 ± 0.17	17.80 ± 0.19	18.86 ± 0.19	18.85 ± 0.19	57.27 ± 0.27	60.18 ± 0.31	60.18 ± 0.31
qCBF A	22.97 ± 0.11	29.42 ± 0.15	28.73 ± 0.13	0.58 ± 0.05	0.71 ± 0.05	0.69 ± 0.05	1.66 ± 0.07	2.09 ± 0.08	2.08 ± 0.08

Tabelle 2.5: Klassifikation: Ergebnisse der **inkrementellen Rotationen mit h = 3**. Dargestellt sind jeweils Mittelwert und Standard-Fehler des **Klassifikations-Fehlers** über 100 stratifizierte Aufteilungen des jeweiligen Datensatzes. Alle Werte sind in % auf zwei Nachkommastellen gerundet angegeben. **[Oben]**: $r_{\text{test}} = 0.1$, **[Mitte]**: $r_{\text{test}} = 0.5$, **[Unten]**: $r_{\text{test}} = 0.9$.

KAPITEL 3

SENSORDATENFUSION

3.1 Einleitung: Transformation in gemeinsames Koordinatensystem

Wenn zwei Messreihen, die mit unterschiedlichen Sensoren gemessen wurden, fusioniert werden sollen, müssen diese erst in ein gemeinsames Koordinatensystem gebracht werden. Nehmen wir an, wir haben zwei Sensoren S_A und S_B. Diese seien zeitlich synchronisiert, so dass zu jedem Zeitpunkt t_i ein Messwert $S_A(t_i) =: S_{A,i}$ und ein Messwert $S_B(t_i) =: S_{B,i}$ existiert. Diese Messwerte seien mit einer festen Frequenz aufgezeichnet. Nehmen wir weiter an, dass die beiden Sensoren *fest* miteinander verbunden sind. Dies bedeutet, dass es *eine* starre Transformation $\mathcal{T} \in SE(3)$ gibt, sodass gilt:

$$S_{A,i} = \mathcal{T}(S_{B,i}), \quad \forall i$$

Die Messwerte, welche die Position und Orientierung des jeweiligen Sensors im dreidimensionalen Raum beschreiben, lassen sich – ebenso wie starre Transformationen $T \in SE(3)$ – als homogene 4×4-Matrizen darstellen. Somit gilt:

$$S_{A,i} = T \cdot S_{B,i}, \quad \forall i$$

mit $S_{A,i}, S_{B,i}, T \in SE(3)$. Der Punkt $(\cdot)$ ist hierbei die gewöhnliche Matrix-Multiplikation. Diese Annahme setzt voraus, dass die beiden Sensoren **fest** miteinander verbunden sind. Beispielsweise kann ein Sensor am unteren Arm direkt hinter dem Handgelenk befestigt sein, der andere am oberen Ende des Unterarms, direkt vor dem Ellbogengelenk (vergleiche Abb. 3.1). Sobald jedoch mindestens ein Gelenk zwischen den beiden Sensoren liegt, z.B. ein Sensor am Unterarm, der andere am Oberarm, gilt die Annahme nicht mehr, dass es eine *feste* starre Transformation gibt, die jeden Messpunkt des einen Sensors auf den entsprechenden Messpunkt des anderen Sensors abbildet.

Bestehen die Sensorsysteme jeweils aus zwei Teilen („Sensor“ und „Marker“) wie es beim Infrarot- bzw. Ultraschall-Tracking der Fall ist, ist diese Situation noch etwas komplizierter, da es zwei Transformationen gibt: eine zwischen den „Sensoren“ und eine zwischen den „Markern“ (siehe Abb. 3.2). Nehmen wir an, wir haben einen Messaufbau, bei dem zwei Sensoren S_A und S_B (z.B. eine Infrarot-Kamera und ein Ultraschall-Mikrofon) die Posen von jeweils einem Marker M_A bzw. M_B (z.B. Infrarot-LEDs und Ultraschall-Emitter) mit fester Frequenz messen. Wir haben die Messwerte der Sensoren vorliegen und möchten diese nun in einem einzigen Koordinatensystem darstellen. Hierzu müssen wir die Transformationen T und S finden, die die Koordinatensysteme der Sensoren und der Marker ineinander überführen. Die Beziehungen der jeweiligen Messgrößen lassen sich anschaulich in einem kommutativen Diagramm darstellen (siehe Abb. 3.2). Wenn wir nun z.B. die Posen $M_{A,i}|_B$ des Markers M_A im Koordinatensystem des Sensors S_B darstellen wollen, haben wir nun zwei Möglichkeiten.

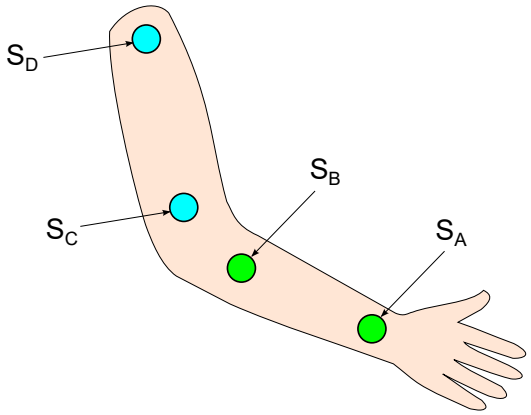


Abbildung 3.1: Beispiel für eine mögliche Positionierung der Sensoren. Die Sensoren S_A und S_B am Unterarm (grün), sowie die Sensoren S_C und S_D am Oberarm (cyan) können jeweils über eine starre Transformation miteinander fusioniert werden. Durch das dazwischen liegende Ellbogengelenk kann S_A bzw. S_B jeweils nicht mit S_C oder S_D fusioniert werden.

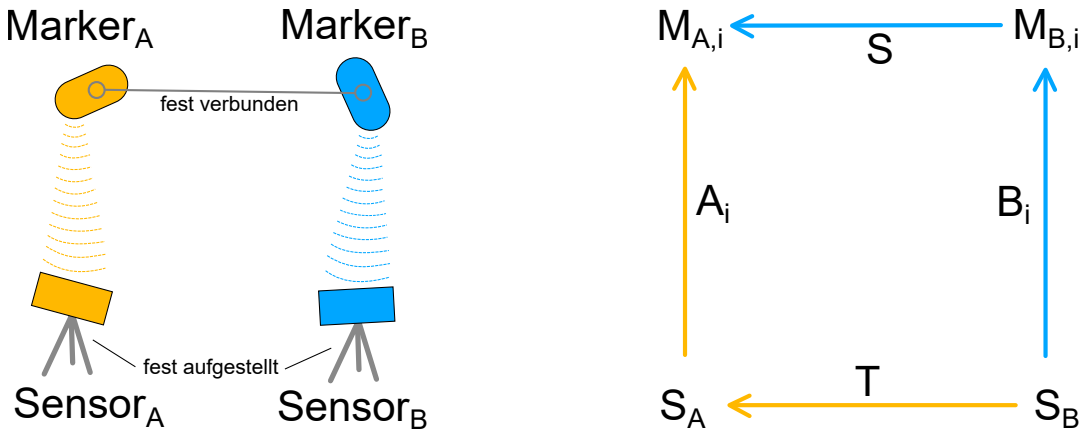


Abbildung 3.2: **[Links]**: Schematischer Messaufbau mit zwei Sensoren und zwei Markern. **[Rechts]**: Kommutatives Diagramm. Die Sensoren S_A und S_B (z.B. Infrarot-Kameras oder Ultraschall-Mikrofone) messen die Posen der Marker M_A und M_B (z.B. Infrarot-LEDs oder Ultraschall-Emitter) zum Zeitpunkt i. A_i ist die Pose des Markers M_A, gemessen von Sensor S_A zum Zeitpunkt i. Analog ist B_i die Pose des Markers M_B, gemessen von Sensor S_B zum selben Zeitpunkt i. Die (starre) Transformation T überführt das Koordinatensystem des Sensors S_B in das des Sensors S_A. Analog dazu überführt die (starre) Transformation S das Koordinatensystem des Markers M_B in das des Markers M_A. Die Posen A_i und B_i im jeweils eigenen Koordinatensystem werden von den Sensoren gemessen und bilden die gemessenen Zeitreihen. Gesucht sind die beiden Transformationen S und T. Möchte man nun z.B. die Posen des Markers $M_{A,i}$ in das Koordinatensystem des Sensors S_B überführen, gibt es zwei Möglichkeiten: Entweder über T und A_i (orange-farbene Pfeile) oder über B_i und S (blaue Pfeile). Beide Möglichkeiten repräsentieren $M_{A,i}$ relativ zu S_B. Folglich gilt: $A_i \cdot T = S \cdot B_i \; \forall i$.

Wie aus der Abb. 3.2 ersichtlich gilt:

$$M_{A,i}|_B = A_i \cdot T$$
$$M_{A,i}|_B = S \cdot B_i$$

Gleichsetzen liefert:

$$A_i \cdot T = S \cdot B_i, \quad \forall i \tag{3.1}$$

Möchte man umgekehrt die Posen $M_{B,i}|_A$ des Markers M_B im Koordinatensystem des Sensors S_A darstellen so gilt

$$M_{B,i}|_A = S^{-1} \cdot A_i$$
$$M_{B,i}|_A = B_i \cdot T^{-1}$$

Gleichsetzen liefert:

$$\begin{aligned} & S^{-1} \cdot A_i \quad = B_i \cdot T^{-1} && | \cdot T \\ \Leftrightarrow \quad & S^{-1} \cdot A_i \cdot T = B_i && | S \cdot \\ \Leftrightarrow \quad & A_i \cdot T \quad = S \cdot B_i \end{aligned}$$

was zur selben Gleichung (3.1) führt. Gesucht sind nun die Transformationen S und T.

Es gilt also, folgendes Problem zu lösen: Gegeben zwei Zeitreihen

$$A = \{A_0, A_1, \ldots, A_{n-1}\}, \quad A_i \in SE(3) \quad \forall i$$
$$B = \{B_0, B_1, \ldots, B_{n-1}\}, \quad B_i \in SE(3) \quad \forall i$$

Man finde zwei Transformationen $S, T \in SE(3)$[(1)], die Gleichung (3.1) erfüllen.

Unter der idealisierten Annahme, dass die Messdaten perfekt aufgenommen wurden und keinerlei Störungen oder Rauschen aufweisen, lässt sich dieses Problem exakt lösen. Da durch Sensoren aufgezeichnete Messwerte jedoch immer ein gewisses Rauschen aufweisen und bei bestimmten Sensortypen auch noch andere Störungen hinzukommen (wie z.B. die Drift bei Inertialsensoren), muss man die Fragestellung als (lineares) Optimierungsproblem auffassen:

Man bestimme S und T derart, dass gilt:

$$(S, T) = \underset{(S,T)}{\arg\min} \sum_{i=0}^{n-1} d(A_i \cdot T, S \cdot B_i) \tag{3.2}$$

(1) Im allgemeinen Fall können S und T beliebige Matrizen aus $\mathbb{R}^{4\times4}$ sein und beispielsweise noch eine Skalierung oder eine andere Transformation enthalten, allerdings beschränken wir uns bei diesem Problem auf starre Transformationen: $S, T \in SE(3)$.

Hierbei ist $d : SE(3) \times SE(3) \mapsto \mathbb{R}_0^+$ eine geeignet zu wählende Abstandsfunktion. Diese Problemstellung ist in der Robotik unter dem Namen „Hand-Eye Calibration“ bekannt. Hierzu gibt es wieder verschiedene Ansätze (für eine Übersicht, siehe z.B. [34]). Grundsätzlich gibt es die Problemstellung in der allgemeineren Form AX=**Y**B mit zwei Transformationen X und Y und in der speziellen Form AX=**X**B mit einer einzigen Transformation X. Die allgemeine Form lässt sich jedoch wie folgt in die spezielle Form überführen:

Man betrachte hierzu zwei verschiedene Messwertpaare (A_i, B_i) und (A_k, B_k) mit $i \neq k$. Mit Gleichung (3.1) erhalten wir folgendes Gleichungssystem:

$$
\begin{array}{lll}
(1) & A_i \cdot T = S \cdot B_i & |A_i^{-1}\cdot \\
(2) & A_k \cdot T = S \cdot B_k & |A_k^{-1}\cdot \\
\hline
(1) & T = A_i^{-1} \cdot S \cdot B_i & \\
(2) & T = A_k^{-1} \cdot S \cdot B_k & \\
\hline
(1)=(2) & A_i^{-1} \cdot S \cdot B_i = A_k^{-1} \cdot S \cdot B_k & |A_i\cdot \\
\Leftrightarrow & S \cdot B_i = A_i \cdot A_k^{-1} \cdot S \cdot B_k & |\cdot B_k^{-1} \\
\Leftrightarrow & S \cdot B_i \cdot B_k^{-1} = A_i \cdot A_k^{-1} \cdot S & \\
\Leftrightarrow & (A_i \cdot A_k^{-1}) \cdot S = S \cdot (B_i \cdot B_k^{-1}) &
\end{array}
$$

Damit lässt sich S mit Hilfe von AX=XB bestimmen. Hier ist $A_i' := (A_i \cdot A_k^{-1})$ und $B_i' := (B_i \cdot B_k^{-1})$. Analog lässt sich das Gleichungssystem umformen, um T ebenfalls via AX=XB zu bestimmen:

$$
\begin{array}{lll}
(1) & A_i \cdot T = S \cdot B_i & |\cdot B_i^{-1} \\
(2) & A_k \cdot T = S \cdot B_k & |\cdot B_k^{-1} \\
\hline
(1) & S = A_i \cdot T \cdot B_i^{-1} & \\
(2) & S = A_k \cdot T \cdot B_k^{-1} & \\
\hline
(1)=(2) & A_i \cdot T \cdot B_i^{-1} = A_k \cdot T \cdot B_k^{-1} & |\cdot B_i \\
\Leftrightarrow & A_i \cdot T = A_k \cdot T \cdot B_k^{-1} \cdot B_i & |A_k^{-1}\cdot \\
\Leftrightarrow & (A_k^{-1} \cdot A_i) \cdot T = T \cdot (B_k^{-1} \cdot B_i) &
\end{array}
$$

Dabei ist $A_i'' := (A_k^{-1} \cdot A_i)$ und $B_i'' := (B_k^{-1} \cdot B_i)$. Eine günstige Wahl des Referenzposen-Paares (A_k, B_k) wird später noch genauer untersucht.

Betrachten wir nun das Optimierungsproblem AX=XB. Da es sich bei den Posen um starre Transformationen handelt, gilt $A_i, B_i \in SE(3)$. Die Transformationsmatrix X muss im Allgemeinen nicht zwingend eine starre Transformation sein. Dies wird jedoch als zusätzliche Bedingung in der folgenden Herleitung gefordert. Wir nehmen an, dass die Konstituenten A_i, B_i, X als homogene 4×4-Matrizen dargestellt sind[(2)]:

$$
\begin{aligned}
& AX = XB \\
\Leftrightarrow \quad & \begin{pmatrix} \Theta_A & \vec{t}_A \\ 000 & 1 \end{pmatrix} \begin{pmatrix} \Theta_X & \vec{t}_X \\ 000 & 1 \end{pmatrix} = \begin{pmatrix} \Theta_X & \vec{t}_X \\ 000 & 1 \end{pmatrix} \begin{pmatrix} \Theta_B & \vec{t}_B \\ 000 & 1 \end{pmatrix} \\
\Leftrightarrow \quad & \begin{pmatrix} \Theta_A\Theta_X & \Theta_A\vec{t}_X + \vec{t}_A \\ 000 & 1 \end{pmatrix} = \begin{pmatrix} \Theta_X\Theta_B & \Theta_X\vec{t}_B + \vec{t}_X \\ 000 & 1 \end{pmatrix}
\end{aligned}
$$

Mit den jeweiligen Rotations-Anteilen $\Theta_i \in SO(3)$ und den Translations-Anteilen $\vec{t}_i \in \mathbb{R}^3$. Für diese Anteile gelten also die folgenden Bestimmungs-Gleichungen:

$$\Theta_A\Theta_X = \Theta_X\Theta_B$$

für den Rotations-Anteil und

$$\Theta_A\vec{t}_X + \vec{t}_A = \Theta_X\vec{t}_B + \vec{t}_X$$

für den Translations-Anteil. Auf gleiche Weise gilt bei AX=YB für die beiden Anteile:

$$\Theta_A\Theta_X = \Theta_Y\Theta_B$$

für den Rotations-Anteil und

$$\Theta_A\vec{t}_X + \vec{t}_A = \Theta_Y\vec{t}_B + \vec{t}_Y$$

für den Translations-Anteil (siehe auch [34]).

Es wurden von verschiedenen Autoren insgesamt drei mögliche Ansätze für diese Problemstellung evaluiert: separierbare Lösungen, simultane Lösungen und iterative Lösungen. Die Art der Lösungen unterscheidet sich darin, in welcher Reihenfolge der rotatorische und der translatorische Anteil gelöst werden.

Bei der separierbaren Lösung wird zuerst der rotatorische Anteil bestimmt und anschließend verwendet, um damit den translatorischen Anteil zu bestimmen. Ein solches Verfahren wurde ursprünglich von Shiu und Ahmad in [36] vorgeschlagen. Effizientere Methoden werden z.B. von Tsai und Lenz in [37] oder auch von Park und Martin in [38] beschrieben. Die Methode von Park und Martin berücksichtigt hierbei die Neben-Bedingung, dass die gefundene Lösung eine starre Transformation ist und wird weiter unten näher erläutert.

(2) Man kann auch eine Repräsentation in Form von Quaternionen oder dualen Quaternionen wählen (siehe z.B. [35]).

Beim simultanen Lösungsansatz werden rotatorischer und translatorischer Anteil gleichzeitig bestimmt, wie z.B. von Chen in [39] beschrieben. Daniilidis und Bayro-Corrochano zeigen in [40] einen Lösungsansatz mit Hilfe von dualen Quaternionen.

Beim iterativen Ansatz wird zunächst wie im ersten Fall der rotatorische Anteil bestimmt und dieser dann verwendet, um den translatorischen Anteil auszurechnen. Diese Ergebnisse werden dann als Startwerte für ein Iterationsverfahren verwendet, um in weiteren Iterationen die Lösungen Schritt für Schritt zu verbessern. Ein solches Verfahren wird z.B. in [41] von Zhuang und Shiu vorgestellt und untersucht.

3.2 Lösungsansatz via AX=XB

Das von uns verwendete Verfahren ist das von Park und Martin in [38] beschriebene separierbare Lösungsverfahren. Hierbei wird das Problem AX=XB gelöst *unter der Nebenbedingung*, dass $X \in SE(3)$ ist, also eine **starre Transformation** beschreibt. Das Problem wird dabei wie folgt formuliert:

Gegeben A_i und B_i mit $i \in \{0,\dots,n-1\}$ und $A_i, B_i \in SE(3)\,\forall i$. Finde ein $X \in SE(3)$, welches das Funktional $\eta = \sum_{i=0}^{n-1} d(A_i \cdot X, X \cdot B_i)$ minimiert. $d : SE(3) \times SE(3) \mapsto \mathbb{R}_0^+$ ist hierbei ein geeignet zu wählendes Abstandsmaß auf der Euklidischen Gruppe. Für das im Folgenden beschriebene Lösungsverfahren wird der Logarithmus von Rotationsmatrizen benötigt:

Definition 26: Matrix-Logarithmus

Sei $\Theta \in SO(3)$ eine Rotationsmatrix mit Spur $\mathrm{Tr}(\Theta) \neq -1$. Dann ist ihr Logarithmus definiert als:

$$\log(\Theta) := \frac{\phi}{2\sin\phi}\left(\Theta - \Theta^\top\right) \quad \text{mit} \quad \phi = \arccos\left(\frac{\mathrm{Tr}(\Theta)-1}{2}\right)$$

Dies ist eine schiefsymmetrische Matrix, welche die Komponenten des Drehvektors $\vec{\omega}$ enthält:

$$\vec{\omega} = \begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\end{pmatrix} \qquad \log(\Theta) =: [\omega] = \begin{pmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{pmatrix} \tag{3.3}$$

Für das Abstandsmaß $d(\cdot)$ schlagen Park et al. folgendes vor:

Sind $A = (\Theta_A, \vec{t}_A)$ und $B = (\Theta_B, \vec{t}_B)$ zwei gegebene Posen (Elemente aus $SE(3)$), dann wähle

$$d^2(A,B) = \|\log(\Theta_A^\top \Theta_B)\|^2 + \|\vec{t}_B - \vec{t}_A\|^2 \tag{3.4}$$

mit der Euklidischen (L_2^2-)Norm $\|\vec{x}\|_2^2 = \|(x_1, x_2, x_3)^\top\|_2^2 = x_1^2 + x_2^2 + x_3^2$ bei den Vektor-wertigen Ausdrücken, bzw. Frobenius-Norm $\|\Theta\|^2 = \|\Theta\|_F^2 = \sum_{i=1}^{3}\sum_{j=1}^{3} \theta_{i,j}^2$ bei den Matrix-wertigen Ausdrücken. Der erste Term beschreibt dabei den Winkelabstand zwischen den Posen, während der zweite Term den räumlichen Abstand der Posen darstellt. Dieses Abstandsmaß ist außerdem invariant unter der Wirkung einer starren Transformation von links:

$$d(A,B) = d(T\cdot A, T\cdot B) \quad \forall A,B,T \in SE(3)$$

Beweis 3

Seien $A = \begin{pmatrix} \Theta_A & \vec{t}_A \\ 000 & 1 \end{pmatrix}$, $B = \begin{pmatrix} \Theta_B & \vec{t}_B \\ 000 & 1 \end{pmatrix}$ und $T = \begin{pmatrix} \Theta_T & \vec{t}_T \\ 000 & 1 \end{pmatrix}$ starre Transformationen ($A, B, T \in SE(3)$). Damit ist

$$T\cdot A = \begin{pmatrix} \Theta_T\Theta_A & \Theta_T\vec{t}_A + \vec{t}_T \\ 000 & 1 \end{pmatrix} \quad \text{und analog} \quad T\cdot B = \begin{pmatrix} \Theta_T\Theta_B & \Theta_T\vec{t}_B + \vec{t}_T \\ 000 & 1 \end{pmatrix}$$

Für den Rotationsanteil gilt:

$$\Theta_A \mapsto \Theta_T\Theta_A \quad \text{bzw.} \quad \Theta_B \mapsto \Theta_T\Theta_B$$

Für den Translationsanteil gilt:

$$\vec{t}_A \mapsto \Theta_T\vec{t}_A + \vec{t}_T \quad \text{bzw.} \quad \vec{t}_B \mapsto \Theta_T\vec{t}_B + \vec{t}_T$$

Nach Gleichung (3.4) ist dann:

$$\begin{aligned} d^2(T\cdot A, T\cdot B) &= \|\log\left((\Theta_T\Theta_A)^\top(\Theta_T\Theta_B)\right)\|^2 + \|(\Theta_T\vec{t}_B + \vec{t}_T) - (\Theta_T\vec{t}_A + \vec{t}_T)\|^2 \\ &= \|\log\left(\Theta_A^\top\Theta_T^\top\Theta_T\Theta_B\right)\|^2 + \|\Theta_T\vec{t}_B + \vec{t}_T - \Theta_T\vec{t}_A - \vec{t}_T\|^2 \\ &= \|\log\left(\Theta_A^\top(\Theta_T^\top\Theta_T)\Theta_B\right)\|^2 + \|\Theta_T(\vec{t}_B - \vec{t}_A)\|^2 \\ &= \|\log\left(\Theta_A^\top(\mathbb{1})\Theta_B\right)\|^2 + \|\Theta_T(\vec{t}_B - \vec{t}_A)\|^2 \\ &= \|\log\left(\Theta_A^\top\Theta_B\right)\|^2 + \|\vec{t}_B - \vec{t}_A\|^2 \\ &= d^2(A,B) \end{aligned}$$ □

Hierbei wurde zum einen die Orthogonalität der Drehmatrizen verwendet:
$\Theta^\top\Theta = \Theta\Theta^\top = \mathbb{1} \quad \forall\Theta \in SO(3)$. Des Weiteren wurde verwendet, dass die Länge eines Vektors invariant unter Drehungen ist, also dass gilt: $\|\vec{t}\| = \|\Theta\vec{t}\| \quad \forall\Theta \in SO(3), \vec{t} \in \mathbb{R}^3$.

Park et al. führen das Problem $\Theta_A\Theta_X = \Theta_X\Theta_B$ mittels der Theorie der Lie-Algebra durch logarithmisches Mapping auf die Form $\Theta_X[\beta] = [\alpha]$ mit $[\alpha] = \log(\Theta_A)$ und $[\beta] = \log(\Theta_B)$

zurück. In dieser Darstellung gilt es zunächst ein $\Theta_X \in SO(3)$ zu bestimmen, welches das Funktional

$$\eta_{\text{rot}} := \sum_{i=0}^{n-1} \|\Theta_X \vec{\beta}_i - \vec{\alpha}_i\|^2$$

minimiert. Hierbei ist $\vec{\alpha}_i$ die aus dem Logarithmus $\log(\Theta_{A,i}) = [\omega_{A,i}]$ konstruierte Drehachse. Entsprechend wird die Drehachse $\vec{\beta}_i$ aus dem Logarithmus $\log(\Theta_{B,i}) = [\omega_{B,i}]$ konstruiert (siehe Gleichung (3.3)). Eine explizite Lösung für die Rotation Θ_X ist dann gegeben durch ([42],[38]):

$$\Theta_X = (M^\top M)^{-1/2} M^\top \tag{3.5}$$

mit $M = \sum_{i=0}^{n-1} \vec{\beta}_i \vec{\alpha}_i^\top$. Die Wurzel $(\cdot)^{1/2}$ ist hierbei die symmetrische, positiv definite Quadratwurzel, welche beispielsweise mit Hilfe einer Singulärwert-Zerlegung bestimmt werden kann. Diese Lösung des rotatorischen Anteils wird dann verwendet, um im nächsten Schritt die Translation $\vec{t}_X$ zu bestimmen, welche das folgende Funktional minimiert:

$$\eta_{\text{trans}} := \sum_{i=0}^{n-1} \|(\Theta_{A,i} - \mathbb{1})\vec{t}_X - \Theta_X \vec{t}_{B,i} + \vec{t}_{A,i}\|^2 = \sum_{i=0}^{n-1} \|(\mathbb{1} - \Theta_{A,i})\vec{t}_X - (\vec{t}_{A,i} - \Theta_X \vec{t}_{B,i})\|^2$$

Eine Standard-Least-Squares-Lösung hierfür ist ([42],[38]):

$$\vec{t}_X = (C^\top C)^{-1} C^\top d$$

$$\text{mit } C = \begin{pmatrix} \mathbb{1} - \Theta_{A,0} \\ \mathbb{1} - \Theta_{A,1} \\ \vdots \\ \mathbb{1} - \Theta_{A,n-1} \end{pmatrix} \text{ und } d = \begin{pmatrix} \vec{t}_{A,0} - \Theta_X \vec{t}_{B,0} \\ \vec{t}_{A,1} - \Theta_X \vec{t}_{B,1} \\ \vdots \\ \vec{t}_{A,n-1} - \Theta_X \vec{t}_{B,n-1} \end{pmatrix}.$$

Dieses Verfahren hat gegenüber anderen Lösungsverfahren für das AX=XB-Problem den Vorteil, dass die Lösung X per Konstruktion eine **starre Transformation** ist ($X = (\Theta_X, \vec{t}_X) \in SE(3)$ mit $\Theta_X \in SO(3)$ und $\vec{t}_X \in \mathbb{R}^3$). Außerdem ist die Lösung für den Rotations-Anteil unabhängig vom Translations-Anteil und damit auch von der Wahl dessen physikalischer Messeinheit (mm, cm, m, inch, etc.), was bei simultanen und iterativen Lösungsansätzen im Allgemeinen nicht der Fall ist.

3.2.1 Wahl des Referenz-Posenpaares

Wie bereits am Anfang dieses Abschnitts beschrieben, muss das eigentliche Problem (Gleichung (3.1)) erst durch Multiplikation jedes Posenpaares mit der Inversen eines **Referenz-Posenpaares** in die Form AX=XB gebracht werden. Je nachdem, ob die Multiplikation mit dem Referenz-Posenpaar von rechts oder von links erfolgt, lässt sich die Transformations-Matrix S oder T bestimmen. S lässt sich durch Multiplikation von rechts bestimmen:

$$A_i \mapsto A_i' := A_i \cdot A_{\text{ref}}^{-1} \quad \text{und} \quad B_i \mapsto B_i' := B_i \cdot B_{\text{ref}}^{-1}$$

Damit gilt (wie weiter oben bereits hergeleitet):

$$A'_i \cdot S = S \cdot B'_i$$

Analog lässt sich T durch Multiplikation von links bestimmen:

$$A_i \mapsto A''_i := A_{\text{ref}}^{-1} \cdot A_i \quad \text{und} \quad B_i \mapsto B''_i := B_{\text{ref}}^{-1} \cdot B_i$$

Dies führt dann zu:

$$A''_i \cdot T = T \cdot B''_i$$

Die Transformation der einen Zeitreihe in das Koordinatensystem der anderen erfolgt nun folgendermaßen. $B|_A$ ist dabei die Zeitreihe B, die in das Koordinatensystem von A transformiert wurde. Analog bezeichnet $A|_B$ die Zeitreihe A, transformiert in das Koordinatensystem von B:

$$B_i|_A = S \cdot B_i \cdot T^{-1} \tag{3.6}$$

$$A_i|_B = S^{-1} \cdot A_i \cdot T \tag{3.7}$$

Bei einer idealen Messung ohne Fehler und ohne Rauschen wäre dann $A_i = B_i|_A \, \forall \, i$ und $B_i = A_i|_B \, \forall \, i$. Bei der Wahl des Referenzposen-Paares gibt es verschiedene Möglichkeiten. Die einfachste Möglichkeit ist es, wenn man sich ein beliebiges Posenpaar $(A_{\text{ref}}, B_{\text{ref}}) := (A_k, B_k)$ mit $0 \leq k \leq n-1$ der Zeitreihen A_i und B_i heraussucht. Wären die Posen alle perfekt und ohne jegliches Rauschen gemessen, wäre die Lösung unabhängig von der Wahl des Referenzposen-Paares. Dies ist jedoch bei realen Messdaten nicht der Fall. Wenn das gewählte Referenzposen-Paar durch Rauschen oder Sensordrift fehlerhaft gemessen wurde, dann pflanzt sich dieser Fehler natürlich auch in den transformierten Zeitreihen A'_i und B'_i (bzw. A''_i und B''_i) fort. Wenn im schlimmsten Falle sogar ein Messausfall oder Messausreißer als Referenz gewählt wird, dann ist die komplette Rechnung unbrauchbar.

Festes Referenz-Posenpaar

Eine einfache Möglichkeit, dieses Problem zu umgehen ist es, die Rechnung mit **jedem Posenpaar** als Referenz durchzuführen. Damit lässt sich die beste Referenzpose bestimmen, allerdings wächst der Rechenzeit-Aufwand dabei linear mit der Länge der Zeitreihen, da das AX=XB-Lösungsverfahren für jedes Referenzposen-Paar $(A_{\text{ref}}, B_{\text{ref}}) \in \{(A_i, B_i) \, \forall \, i\}$ durchgeführt werden muss.

„Gemitteltes“ Referenz-Posenpaar

Eine andere Möglichkeit, eine Referenzpose zu wählen, wäre eine Art **„mittlere Pose“**. Während der Mittelwert für den translatorischen Anteil $\vec{t} \in \mathbb{R}^3$ mit

$$\bar{\vec{t}}_{k,k+p} := \frac{1}{p+1} \sum_{i=k}^{k+p} \vec{t}_i$$

wohldefiniert ist, muss man sich für die „mittlere“ Rotation eine sinnvolle Definition überlegen. Dies ist beispielsweise mit Hilfe von Quaternionen $q \in \mathbb{H}$ möglich. Auch hier lässt sich der „Mittelwert“ definieren:

$$\bar{q}_{k,k+p} := \tfrac{1}{p+1} \sum_{i=k}^{k+p} q_i$$

Man muss hierbei jedoch beachten, dass alle Quaternionen auf der gleichen Hemisphäre im $\mathbb{R}^4$ liegen. Durch die Antipodalität beschreiben q und $-q$ dieselbe Drehung, würden sich aber bei einer einfachen Mittelung gegenseitig aufheben. Man kann dieses Problem jedoch einfach wie folgt beheben:

Man betrachte dazu das Skalarprodukt zwischen zwei Quaternionen $q_i, q_j \in \mathbb{H}$:

$$\langle q_i | q_j \rangle := \sum_{\mu=0}^{3} q_{i,\mu} \cdot q_{j,\mu}$$

Möchte man nun erzwingen, dass die Quaternionen, über die gemittelt wird, auf der gleichen Hemisphäre liegen, dann muss man bei der Wahl des Vorzeichens (q_i oder $-q_i$) jeweils beachten, dass als Drehwinkel zwischen dem Quaternion q_i und einem beliebig ausgewählten Referenzquaternion $q_\mathrm{r} \in \{q_k, q_{k+1}, \ldots, q_{k+p}\}$ immer der kleinere gewählt wird. Mit anderen Worten, muss das Skalarprodukt $\langle q_\mathrm{r} | q_i \rangle$ stets ≥ 0 sein. Damit ergibt sich für die Wahl des Vorzeichens von q_i folgende Vorschrift (Für eine anschauliche Darstellung siehe Abb. 3.3.):

$$q_i \mapsto \begin{cases} -q_i & \text{wenn} \quad \langle q_\mathrm{r} | q_i \rangle < 0 \\ q_i & \text{wenn} \quad \langle q_\mathrm{r} | q_i \rangle \geq 0 \end{cases} \qquad k \leq i \leq k+p$$

Damit lässt sich als Referenzpose eine „mittlere“ Pose $A_\mathrm{ref} = \bar{A}_{k,k+p} := (\bar{q}_{k,k+p}, \bar{\vec{t}}_{k,k+p})$ bestimmen.

Hierzu muss jedoch ein Ausschnitt der Zeitreihen gewählt werden, an dem die Bewegung sehr langsam erfolgt. Idealerweise müsste sogar ein Stillstand in der Bewegung vorhanden sein. Dies kann z.B. ganz am Anfang einer Bewegungsmessung erfolgen. Voraussetzung ist dabei allerdings, dass die Messung eine solche Phase aufweist, was eine starke Einschränkung darstellt. Man müsste z.B. die Messung kalibrieren, indem man am Anfang für ein paar Sekunden eine Ruheposition einnimmt. Da dies im Allgemeinen nicht der Fall ist werden wir diesen Ansatz nicht weiter verfolgen.

Variables Referenz-Posenpaar

Eine weitere Möglichkeit ist es, jeweils die (h-ten) **nächsten Nachbar-Posen** als Referenzposen-Paar zu nehmen: $(A_{\mathrm{ref},i}, B_{\mathrm{ref},i}) := (A_{i+h}, B_{i+h})$, $0 \leq i \leq n-h-1$. Damit ist keine Abhängigkeit von der Wahl einer festen Referenzpose mehr gegeben. Durch die Multiplikation

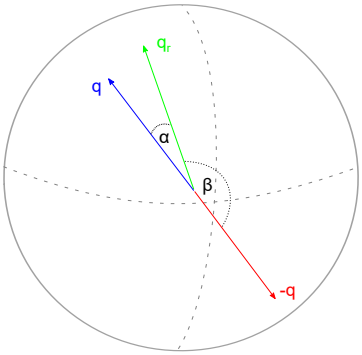


Abbildung 3.3: Antipodalität der Quaternionen (Darstellung auf der Einheitssphäre im $\mathbb{R}^4$): q und $-q$ beschreiben dieselbe Drehung. Durch die Wahl des Vorzeichens über das Skalarprodukt $\langle q|q_r\rangle \geq 0$ wird das Quaternion ausgewählt, welches den kleineren Winkel mit dem Referenzquaternion q_r einschließt. In diesem Beispiel wäre dies das Quaternion q mit positivem Vorzeichen.

mit der Inversen der jeweiligen Nachbarpose erhält man die jeweilige inkrementelle Rotation, die die Orientierung in einem Zeitpunkt zur Orientierung im h-ten nächsten Zeitpunkt überführt. Gleiches gilt auch für den translatorischen Anteil. Dieses Verfahren ist zwar unabhängig von der Wahl eines festen Referenzposen-Paares, hat dafür jedoch einen anderen, entscheidenden Nachteil. Wenn die Bewegung im Vergleich zur Messfrequenz relativ langsam verläuft, ist die inkrementelle Zeitreihe sehr „klein“. Dies bedeutet, dass die Elemente $A'_i \approx \mathbb{1}^{4\times4}$, sowie $B'_i \approx \mathbb{1}^{4\times4}$ sind. Damit entartet das Problem $A' \cdot X = X \cdot B'$ zu $X \approx X$, welches für jedes beliebige $X \in \mathbb{R}^{4\times4}$ trivial erfüllt ist. Deshalb können S und T in diesem Falle nicht mehr korrekt berechnet werden. Aus diesem Grunde muss der Parameter h hinreichend groß gewählt werden, damit dieser Fall nicht eintritt. Dadurch werden jedoch die Zeitreihen um h Elemente verkürzt. Um den optimalen Wert für h zu bestimmen, muss also wieder jeder Wert von $1 \leq h \leq h_{\max}$ untersucht werden. Dazu sollte $h_{\max}$ als fester prozentualer Anteil der Zeitreihen-Länge gewählt werden. Damit wächst die benötigte Rechenzeit allerdings auch hier wieder linear mit der Länge der Zeitreihen.

3.3 Lösungsansatz via Quaternion-AX=YB

Will schlägt in seiner Dissertation [43] ein weiteres Verfahren vor, welches AX=YB direkt löst. Es handelt sich hierbei um eine modifizierte Variante des ursprünglich von Zhuang et al. in [44] vorgeschlagenen Algorithmus, welcher allerdings die Antipodalität der Quaternionen mit berücksichtigt. Auch hier wird die Nebenbedingung eingehalten, dass $X, Y \in SE(3)$, die Transformationen X und Y also jeweils **starre Transformationen** sind. Dabei wird das folgende Funktional in zwei Schritten gelöst:

$$\eta = l \sum_{i=0}^{n-1} (\varphi(\Theta_{A,i}\Theta_X, \Theta_Y\Theta_{B,i}))^2 + \sum_{i=0}^{n-1} \|\Theta_{A,i}\vec{t}_X + \vec{t}_{A,i} - (\Theta_Y\vec{t}_{B,i} + \vec{t}_Y)\|^2 \tag{3.8}$$

Die Funktion $\varphi : SO(3) \times SO(3) \mapsto [0,\pi] \subset \mathbb{R}, (\Theta_1, \Theta_2) \mapsto \arccos\left(\frac{\mathrm{Tr}(\Theta_1^\top \Theta_2)-1}{2}\right)$ im ersten Summanden beschreibt dabei den relativen Drehwinkel zwischen den durch Θ_1 und Θ_2 repräsentierten Rotationen. Der Ausdruck $\|\cdot\|^2$ im zweiten Summanden ist die gewöhnliche Euklidische L_2^2-Norm auf $\mathbb{R}^3$. $l \in \mathbb{R}^+$ ist ein beliebig zu wählender Skalierungsfaktor zwischen rotatorischem und translatorischem Anteil.

Es handelt sich auch hier um ein *separierbares Verfahren*, welches zuerst den rotatorischen Anteil löst, indem der erste Summand des obigen Funktionals (3.8) minimiert wird. Die gefundene Lösung (Θ_Y) wird anschließend verwendet, um damit den translatorischen Anteil zu lösen, indem der zweite Summand minimiert wird. Aus diesem Grunde spielt auch der Skalierungsfaktor l für die Optimierung keine Rolle, da die Minimums-Lösung des rotatorischen Anteils unabhängig von einem Vorfaktor ist. Da der translatorische Anteil anschließend getrennt gelöst wird, ist dieser ebenfalls unabhängig von l.

Um den rotatorischen Anteil zu lösen, wird der Ausdruck

$$\Theta_{A,i}\Theta_X = \Theta_Y\Theta_{B,i} \Leftrightarrow \Theta_{B,i} = \Theta_Y^\top\Theta_{A,i}\Theta_X \quad \text{mit} \quad \Theta_{A,i}, \Theta_{B,i}, \Theta_X, \Theta_Y \in SO(3) \tag{3.9}$$

wie folgt mit Hilfe von Quaternionen ausgedrückt:

$$q_{B,i} = q_Y^* * q_{A,i} * q_X \quad \text{mit} \quad q_{A,i}, q_{B,i}, q_X, q_Y \in \mathbb{H}_1 \tag{3.10}$$

Wenn man nun die Quaternionen-Produkte ($*$) auf der rechten Seite von Gleichung (3.10) ausrechnet und die Komponenten entsprechend umsortiert, so lässt sich dieser Ausdruck wie folgt darstellen [43]:

$$b_i = M \cdot a_i \quad \text{mit} \quad a_i, b_i \in \mathbb{R}^4, M \in SO(4) \tag{3.11}$$

Hierbei sind a_i und b_i die Quaternionen $q_{A,i}$ bzw. $q_{B,i}$ aus Gleichung (3.10), interpretiert als Vektoren im $\mathbb{R}^4$. Für die aus den Komponenten von q_X und q_Y konstruierte Matrix M gilt: $M^\top \cdot M = M \cdot M^\top = \mathbb{1}$, sowie $\det(M) = +1$. Es handelt sich hierbei also um eine 4×4-Rotationsmatrix $M \in SO(4)$, die die Quaternionen aus der Zeitreihe A auf die Quaternionen aus der Zeitreihe B rotiert [43].

Eine Least-Squares-Lösung für die Matrix M aus Gleichung (3.11) lässt sich nun mit Hilfe einer Singulärwert-Zerlegung der Matrix $C = \sum_{i=0}^{n-1} a_i \cdot b_i^\top$ bestimmen und damit auch die Quaternionen q_X und q_Y aus Gleichung (3.10), aus deren Komponenten die Matrix M besteht:

$$q_X = \frac{M_X}{\|M_X\|} \quad \text{mit} \quad M_X := \begin{pmatrix} M_{00} + M_{11} + M_{22} + M_{33} \\ (M_{32} - M_{23}) + (M_{01} - M_{10}) \\ (M_{13} - M_{31}) + (M_{02} - M_{20}) \\ (M_{21} - M_{12}) + (M_{03} - M_{30}) \end{pmatrix} \in \mathbb{R}^4$$

$$q_Y = \frac{M_Y}{\|M_Y\|} \quad \text{mit} \quad M_Y := \begin{pmatrix} M_{00}+M_{11}+M_{22}+M_{33} \\ (M_{32}-M_{23})-(M_{01}-M_{10}) \\ (M_{13}-M_{31})-(M_{02}-M_{20}) \\ (M_{21}-M_{12})-(M_{03}-M_{30}) \end{pmatrix} \in \mathbb{R}^4$$

Hierbei gibt es jedoch folgenden Sonderfall zu beachten: Ist die Spur der Matrix M, $\mathrm{Tr}(M) = 0$, so können q_X und q_Y nicht direkt aus M bestimmt werden. Man kann in diesem Falle jedoch eine Matrix P finden, welche aus den Komponenten zweier Basis-Quaternionen $q_u, q_v \in \{(1,0,0,0)^\top, (0,1,0,0)^\top, (0,0,1,0)^\top, (0,0,0,1)^\top\}$ konstruiert wird, mit $\mathrm{Tr}(P \cdot M) \neq 0$. Aus der Matrix $P \cdot M$ lassen sich dann die Quaternionen $q'_X = q_u * q_X$ und $q'_Y = q_v * q_Y$ und daraus schließlich $q_X = q_u^* * q'_X$ und $q_Y = q_v^* * q'_Y$ rekonstruieren. Für weitere Details siehe [43].

Da dieser Algorithmus – geometrisch interpretiert – die Rotations-Quaternionen $q_{A,i}$ aus der Zeitreihe A (interpretiert als Vektoren im $\mathbb{R}^4$) auf die Rotations-Quaternionen $q_{B,i}$ der Zeitreihe B (ebenfalls interpretiert als Vektoren im $\mathbb{R}^4$) rotiert, spielt es eine entscheidende Rolle, welches gegenseitige Vorzeichen $q_{A,i}$ und $q_{B,i}$ jeweils haben. Aufgrund der **Antipodalität** können die Rotations-Quaternionen der Zeitreihen A und B jedoch beliebig geflippt sein, da zwei Quaternionen q und $-q$ jeweils dieselbe Rotation beschreiben.

Um für zwei Zeitreihen der Länge n nicht alle 2^{n-1} relevanten[(3)] Flipping-Konfigurationen ausprobieren zu müssen, schlägt Will folgenden RANSAC-Ansatz [45] vor. Darin wird für 4 zufällig ausgewählte Quaternionen-Paare eine (vorläufige) Matrix M jeweils für jede der 8 relevanten Flipping-Konfigurationen dieser 4 zufällig gewählten Quaternionen-Paare bestimmt. Erfüllt eine davon das Kriterium $|\cos(\alpha_i)| = |b_i^\top M \cdot a_i| > c$ für ein geeignet zu wählendes $c \in (0,1)$, $\forall i$ (α_i ist hierbei der Winkel zwischen b_i und $M \cdot a_i$), so wird mit Hilfe dieser vorläufigen Rotationsmatrix M die Flipping-Konfiguration der Quaternionen aus den Zeitreihen A und B bestimmt. (Sollte das Kriterium für keine der 8 M-Matrizen erfüllt sein, werden 4 neue Quaternionen-Paare zufällig ausgewählt und das Verfahren mit diesen wiederholt.) Mit einer gefundenen Matrix M, die das Kriterium erfüllt, wird nun wie folgt die Flipping-Konfiguration bestimmt:

$$a_i \mapsto \begin{cases} -a_i & \text{wenn} \quad \cos(\alpha_i) = b_i^\top M \cdot a_i < -c \\ a_i & \text{sonst} \end{cases} \qquad 0 \le i \le n-1$$

Mit Hilfe dieser Flipping-Konfiguration wird dann die „exakte“ Matrix M unter Berücksichtigung aller a_i und b_i mit $0 \le i \le n-1$ bestimmt und damit die Quaternionen q_X und q_Y berechnet (siehe oben).

(3) Bei n Quaternionen-Paaren $q_{A,i}$ und $q_{B,i}$ gibt es insgesamt 4^n Flipping-Konfigurationen. Da jedoch nur das *relative* Vorzeichen zwischen jeweils zwei Quaternionen von Interesse ist, gibt es für jedes Quaternionen-Paar nur 2 statt 4 *relevante* Flipping-Konfigurationen. Alle $q_{A,i}$ oder alle $q_{B,i}$ zu flippen kann jedoch auch durch eine Rotation im $\mathbb{R}^4$ kompensiert werden, da $(-\mathbb{1}) \in SO(4)$. Deswegen gibt es bei n Quaternionen-Paaren nur 2^{n-1} *relevante* Flipping-Konfigurationen.

Mittels der gefundenen Quaternionen q_X und q_Y lassen sich die Rotations-Matrizen Θ_X und Θ_Y aus Gleichung (3.9) bestimmen. Damit ist der rotatorische Anteil des Problems gelöst.

Der translatorische Anteil des Problems wird anschließend gelöst, indem der Ausdruck

$$\eta_{\text{trans}} := \left\| \begin{pmatrix} \Theta_{A,0} & -\mathbb{1} \\ \Theta_{A,1} & -\mathbb{1} \\ \vdots & \vdots \\ \Theta_{A,n-1} & -\mathbb{1} \end{pmatrix} \cdot \begin{pmatrix} \vec{t}_X \\ \vec{t}_Y \end{pmatrix} - \begin{pmatrix} \Theta_Y \vec{t}_{B,0} - \vec{t}_{A,0} \\ \Theta_Y \vec{t}_{B,1} - \vec{t}_{A,1} \\ \vdots \\ \Theta_Y \vec{t}_{B,n-1} - \vec{t}_{A,n-1} \end{pmatrix} \right\|^2$$

bezüglich der Translations-Vektoren $\vec{t}_X$ und $\vec{t}_Y$ minimiert wird. Dies ist ein aus der Numerik wohlbekanntes Problem, welches sich mittels QR- oder Singulärwert-Zerlegung der Matrix auf der linken Seite lösen lässt (siehe z.B. [46]).

Die vorangehend beschriebenen Verfahren wurden auf sehr vielfältigen Daten, die sehr unterschiedlichen Anwendungs-Szenarien entsprechen, getestet und analysiert (siehe kommender Abschnitt). Dabei wurde eine relativ rauscharme Messreihe, die mit zwei unterschiedlichen Sensortypen aufgezeichnet wurde und im Kontrast dazu verschiedene Messreihen einer Motorbewegung, die sehr großes Rauschen aufweisen, untersucht. Daneben wurden noch künstlich erzeugte Datensätze verwendet, bei denen der Rausch-Anteil künstlich generiert wird und dadurch frei wählbar ist. Damit lässt sich die Robustheit der oben beschriebenen Sensordaten-Fusions-Verfahren gegenüber Sensor-Rauschen genau studieren.

3.4 Ergebnisse

3.4.1 Ergebnisse: künstliche Datensätze

Um die beiden vorangehend beschriebenen Algorithmen miteinander vergleichen zu können, wurden diese auf künstlich erzeugten Datensätzen getestet. Hierzu wurden alle 2003 Shapes des JGU_Numbers-Datensatzes, sowie 5613[(4)] Shapes des 6DMG-Datensatzes (siehe Kapitel 1) verwendet, um auf folgende Weise künstliche AX=YB-Daten daraus zu erzeugen. Bei diesen beiden Datensätzen liegen die Bewegungen als vollständige 6-DoF-Posen vor, auch wenn bei der Klassifikation im vorangehenden Kapitel lediglich die Rotations-Anteile der Messdaten verwendet wurden. Für jedes Shape wurden zufällig zwei Transformationen $S, T \in SE(3)$ generiert. Dazu wurde eine im Raum zufällig orientierte, normierte Rotations-Achse $\vec{r} \in \mathbb{R}^3, \|\vec{r}\| = 1$ gewählt und ein zufällig uniform im Intervall von $\varphi \in [-\pi, +\pi]$ gewählter Winkel verwendet, um damit ein Rotations-Quaternion $q = (\cos(\frac{\varphi}{2}), \sin(\frac{\varphi}{2}) \cdot \vec{r}))^\top$ zu konstruieren, aus welchem dann eine Rotations-Matrix Θ berechnet wurde. Der translatorische Anteil $\vec{t} = (t_x, t_y, t_z)^\top \in \mathbb{R}^3$ wurde erzeugt, indem jede Komponente t_i zufällig uniform aus dem Intervall $[-100\text{mm}, +100\text{mm}]$ gewählt wurde. Aus der Rotations-Matrix Θ und dem Translations-Vektor $\vec{t}$ wurde dann schließlich eine homogene 4×4-Matrix konstruiert. Die auf diese Weise zufällig erzeugten Transformationen S und T wurden dann verwendet, um aus den Posen der JGU_Numbers- bzw. 6DMG-Zeitreihen A_i, die transformierten Zeitreihen $B_i := S^{-1} \cdot A_i \cdot T$ zu erzeugen. Anschließend wurden jeweils beide Zeitreihen A_i und B_i mit zufällig erzeugtem Rauschen – sowohl im rotatorischen, als auch im translatorischen Anteil – versehen. Dazu wurden für jeden einzelnen Zeitpunkt wieder je zwei zufällige starre Transformationen $Q_i, R_i \in SE(3)$ erzeugt und die Zeitreihen damit „gestört“: $A'_i := Q_i \cdot A_i$ und $B'_i := R_i \cdot B_i$. Das *translatorische* Rauschen spielte hierbei keine allzu große Rolle und wurde für jede der drei Raum-Komponenten auf das Intervall $[-10\text{mm}, +10\text{mm}]$ beschränkt. Dies entspricht ca. 5% der mittleren räumlichen Ausdehnung der JGU_Numbers-Shapes. Bei den 6DMG-Shapes entspricht dies in etwa 2% der mittleren räumlichen Ausdehnung. Um die Auswirkungen des *rotatorischen* Rauschens zu untersuchen, wurde der maximale Störungs-Winkel des rotatorischen Anteils im Bereich von $\varphi_{\text{noise,max}} \in [0°, 20°]$ in Schritten von jeweils 1° durchgefahren. Dies bedeutet, dass der Winkel des rotatorischen Rauschens φ_{noise} jeweils zufällig uniform aus dem Intervall von $\varphi_{\text{noise}} \in [-\varphi_{\text{noise,max}}, +\varphi_{\text{noise,max}}]$ gewählt wurde.

Um die Ergebnisse miteinander vergleichen zu können, benötigt man ein Abstandsmaß zwischen den beiden Zeitreihen. Da beide vorgestellten Verfahren den rotatorischen und den translatorischen Anteil getrennt voneinander lösen und es des Weiteren auch physika-

(4) Bei 2 der insgesamt 5615 Shapes des 6DMG-Datensatzes konnte bei der Lösung via AX=XB die Wurzel aus der Matrix M in Gleichung (3.5) nicht ausgewertet werden werden. Dieser Fall tritt auf wenn die Matrix, deren Wurzel berechnet werden soll, entweder nicht invertierbar ist oder reelle negative Eigenwerte besitzt (siehe https://eigen.tuxfamily.org/dox/unsupported/group__MatrixFunctions__Module.html#matrixbase_sqrt). Diese beiden Shapes wurden deshalb nicht verwendet.

lisch nicht sinnvoll ist, eine Summe aus einer Rotation und einer Translation (so etwas wie: mm + °) zu definieren, werden im Folgenden rotatorischer und translatorischer Anteil auch getrennt betrachtet. Hierzu werden der mittlere Rotations-Fehler und der mittlere Translations-Fehler wie folgt definiert:

Definition 27: mittlerer Rotations-Fehler und mittlerer Translations-Fehler

Seien $A = \{A_i\}$ mit $A_i = (\Theta_{A,i}, \vec{t}_{A,i})$ und $B = \{B_i\}$ mit $B_i = (\Theta_{B,i}, \vec{t}_{B,i})$ zwei gleich lange Zeitreihen der Länge $|A| = |B| = n$ bestehend aus starren Transformationen $(A_i, B_i \in SE(3))$. Seien ferner $S = (\Theta_S, \vec{t}_S)$ und $T = (\Theta_T, \vec{t}_T)$ zwei starre Transformationen $(S, T \in SE(3))$, sodass $A_i \cdot T \approx S \cdot B_i \ \forall i$. Hierbei bezeichnet jeweils $\Theta \in SO(3)$ Rotations-Matrizen und $\vec{t} \in \mathbb{R}^3$ Translations-Vektoren.

Der mittlere*) *Rotations*-Fehler der Transformationen S und T ist nun definiert als:

$$\varepsilon_{\text{rot}}(A, B, S, T) := \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (\varphi(\Theta_{A,i}\Theta_T, \Theta_S\Theta_{B,i}))^2}$$

mit

$$\varphi : SO(3) \times SO(3) \mapsto [0, \pi], (\Theta_1, \Theta_2) \mapsto \arccos\left(\frac{\text{Tr}\,(\Theta_1^\top \Theta_2) - 1}{2}\right)$$

und der mittlere*) *Translations*-Fehler als:

$$\varepsilon_{\text{trans}}(A, B, S, T) := \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} \|\Theta_{A,i}\vec{t}_T + \vec{t}_{A,i} - (\Theta_S\vec{t}_{B,i} + \vec{t}_S)\|^2}$$

*) Es handelt sich hierbei um den Mittelwert des jeweiligen Fehlers über alle Posenpaare (A_i, B_i) der beiden Zeitreihen.

Es wurden die folgenden 3 Verfahren[(5)] miteinander verglichen:

- AX=XB mit festem Referenz-Posenpaar
- AX=XB mit variablem Referenz-Posenpaar
- Quaternion-AX=YB

Die ersten beiden entsprechen dem von Park und Martin [38] vorgeschlagenen Verfahren. Im ersten Falle wurde jeweils ein Posenpaar als feste Referenz für alle Zeitpunkte gewählt. Dazu wurden alle Paare $(A_{\text{ref}}, B_{\text{ref}}) := (A_k, B_k), 0 \leq k \leq n-1$ getestet und das ausgewählt, welches das beste Ergebnis (im Sinne des kleinsten Abstandes) lieferte. Im zweiten Falle wurde das jeweils h-te nächste Nachbar-Posenpaar $(A_{\text{ref,i}}, B_{\text{ref,i}}) := (A_{i+h}, B_{i+h}), 0 \leq h \leq \frac{n}{3}$

(5) Genauer: 2 Varianten des Verfahrens via AX=XB und das Verfahren via Quaternion-AX=YB

verwendet. Auch hier wurden alle Werte von $0 \leq h \leq \frac{n}{3}$ getestet und das ausgewählt, welches den kleinsten Abstand lieferte. Als Abstands-Maß zur Bewertung der Ergebnisse bei der jeweiligen Parameterwahl diente hier (ausnahmsweise) einfach die Summe des mittleren Rotations-Fehlers und des mittleren Translations-Fehlers gemäß Definition 27. Das dritte Verfahren, welches wir als „Quaternion-AX=YB“ bezeichnen wollen, ist der von Will [43] vorgeschlagene Ansatz.

Die Ergebnisse sind in Abb. 3.4 und 3.5 dargestellt. Es ist deutlich zu erkennen, dass das Quaternion-AX=YB-Verfahren dem AX=XB-Verfahren überlegen ist. Die Abhängigkeit von einem Referenz-Posenpaar – sowohl von einem festen, als auch von einem variablen – verschlechtert die Ergebnisse deutlich. Im Idealfall „perfekt“ (ohne Rauschen, Sensor-Drift oder andere Störungen) gemessener Daten wäre das Ergebnis unabhängig von der Wahl des Referenz-Posenpaares. Da reale Messdaten jedoch immer von Rauschen betroffen sind, tritt dieser Fall in der Realität nicht ein. Dies wurde durch die „Störung“ der künstlich generierten Zeitreihen mit einer zufälligen Rotation und Translation an jedem Zeitpunkt simuliert. Da beide Verfahren zuerst das rotatorische Problem lösen und mit der dort gefundenen Lösung den translatorischen Anteil berechnen, ist letzterer von der Fehler-Fortpflanzung betroffen: Ein Fehler in der zuvor bestimmten Rotation wirkt sich anschließend auf die Bestimmung der Translation aus. Aus diesem Grund wirkt sich ein Rauschen des translatorischen Anteils der Posen nicht so gravierend auf die Ergebnisse aus, wie ein Rauschen des rotatorischen Anteils. Je stärker das Rauschen der gesamten Zeitreihen, desto stärker wirkt sich dies natürlich auch auf die gewählten Referenz-Posen aus. Da das Quaternion-AX=YB-Verfahren unabhängig von einem Referenz-Posenpaar arbeitet, ist es auch deutlich robuster gegen Sensor-Rauschen. Aus diesem Grund sind die durchschnittlichen rotatorischen und translatorischen Fehler der untersuchten Verfahren ohne Rauschen im rotatorischen Anteil (maximaler Rotations-Winkel der „Störung“ $\varphi_{\text{noise,max}} = 0°$) auch fast identisch und driften dann für $\varphi_{\text{noise,max}} > 0°$ zunehmend auseinander (siehe Abb. 3.4 und 3.5). Ein weiterer Vorteil des Quaternionen-AX=YB-Verfahrens ist seine numerische Stabilität. Das AX=XB-Verfahren verwendet den Matrix-Logarithmus $\log(\Theta)$ (siehe Definition 26), welcher für Matrizen $\Theta \in SO(3)$ mit $\text{Tr}(\Theta) = -1$ nicht definiert ist. Des Weiteren wird in Gleichung (3.5) die Wurzel aus der (aus den logarithmierten Rotationen konstruierten) Matrix M gezogen, was voraussetzt, dass diese invertierbar ist und keine negativen reellen Eigenwerte besitzt. Diese Einschränkungen führen dazu, dass das Lösungsverfahren in einzelnen Fällen fehlschlagen kann. Aus diesem Grund mussten zwei der 5615 Shapes des 6DMG-Datensatzes beim Vergleich der Lösungsverfahren ausgelassen werden. Ein weiterer Nachteil des AX=XB Verfahrens ist, dass – für die Variante mit festem Referenz-Posenpaar – zur Bestimmung des besten Referenz-Posenpaares die Gesamtlaufzeit um einen Faktor n erhöht wird (n bezeichnet hierbei die Länge der beiden Zeitreihen: $n = |A| = |B|$). Bei der Variante mit variablem Referenz-Posenpaar erhöht sich die Gesamtlaufzeit um einen Faktor $f \propto n$ (in unserem Fall $f = \frac{n}{3}$) zur Bestimmung des optimalen Wertes für den Parameter h. Beim

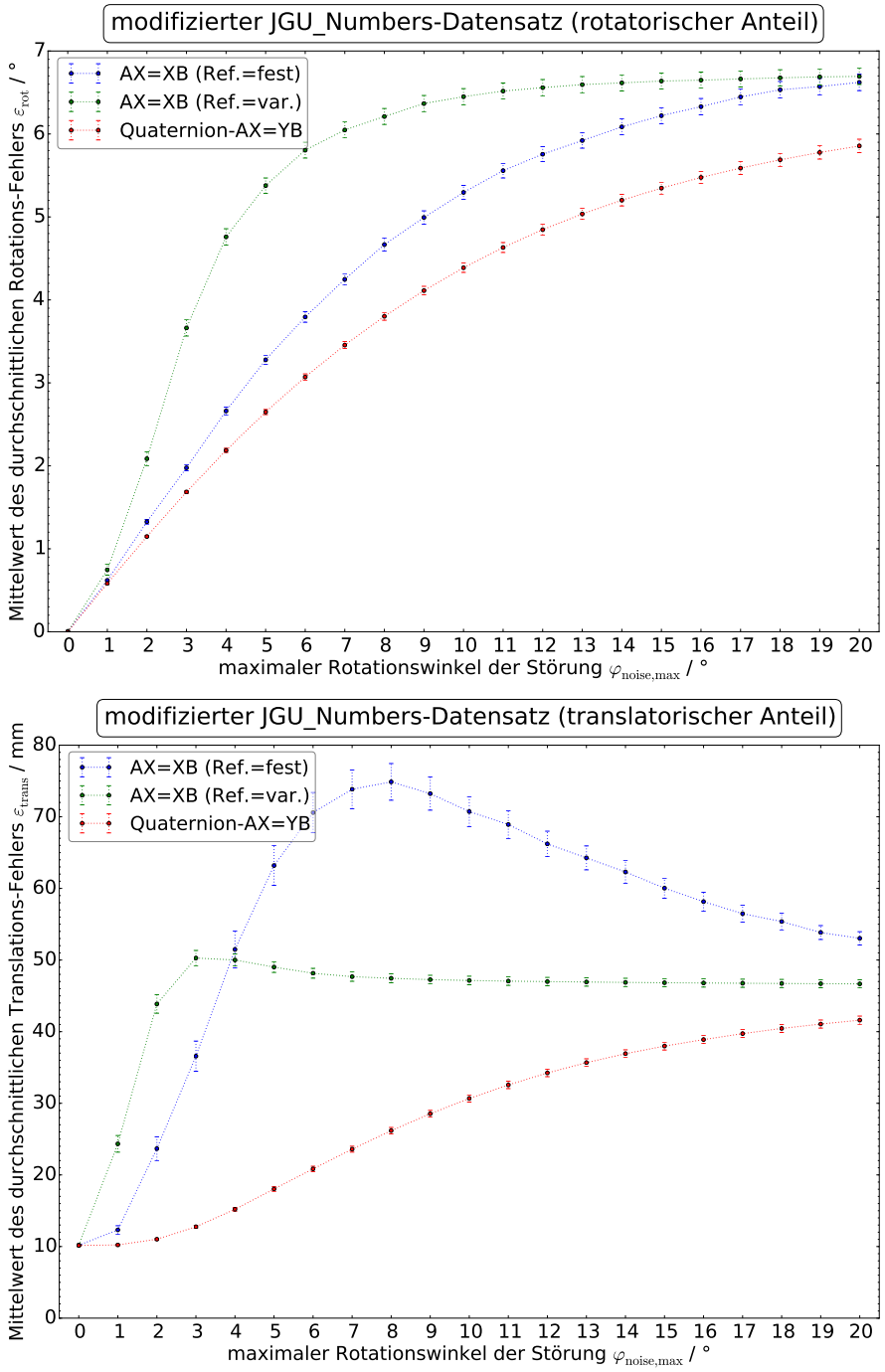


Abbildung 3.4: Rotations- und Translations-Fehler der mit den unterschiedlichen Verfahren bestimmten Transformationen S und T auf den künstlich transformierten Shapes des JGU_Numbers-Datensatzes. Dargestellt sind Mittelwert und Standard-Fehler (als Fehler-Balken) des durchschnittlichen Rotations-Fehlers pro Zeitpunkt **[oben]** und des durchschnittlichen Translations-Fehlers pro Zeitpunkt **[unten]** über alle 2003 Shapes des JGU_Numbers-Datensatzes – jeweils gegen den maximalen Rotationswinkel der Störung $\varphi_{\mathrm{noise,max}}$ (siehe Text).

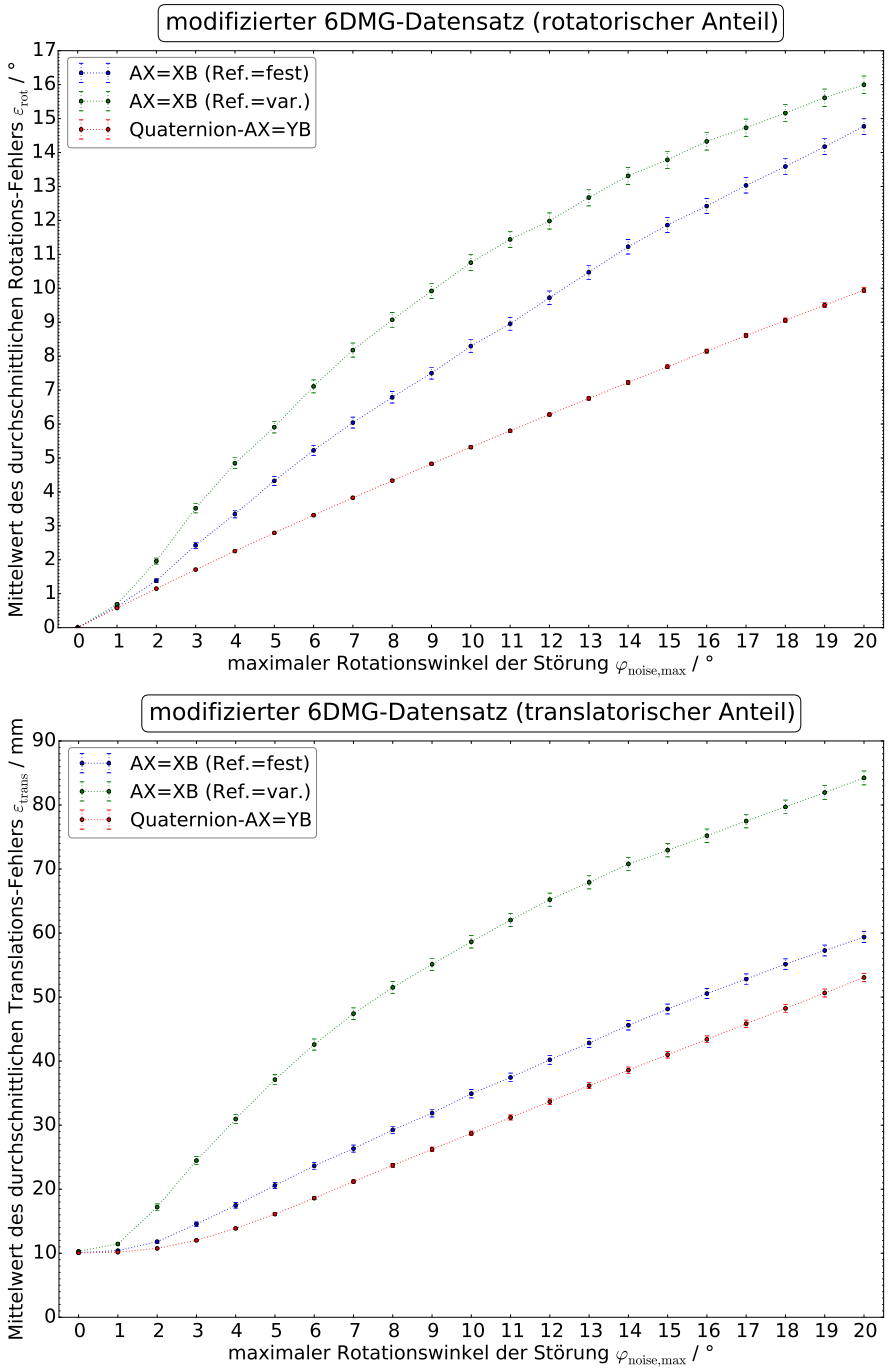


Abbildung 3.5: Rotations- und Translations-Fehler der mit den unterschiedlichen Verfahren bestimmten Transformationen S und T auf den künstlich transformierten Shapes des 6DMG-Datensatzes. Dargestellt sind Mittelwert und Standard-Fehler (als Fehler-Balken) des durchschnittlichen Rotations-Fehlers pro Zeitpunkt **[oben]** und des durchschnittlichen Translations-Fehlers pro Zeitpunkt **[unten]** über 5613 Shapes des 6DMG-Datensatzes – jeweils gegen den maximalen Rotationswinkel der Störung $\varphi_{\text{noise,max}}$ (siehe Text).

direkten Vergleich der beiden AX=XB-Verfahren ist das Verfahren mit fester Referenz-Pose in der Regel besser als das Verfahren mit variabler Referenz-Pose. Lediglich der Translations-Fehler beim JGU_Numbers-Datensatz ist für Werte von $\varphi_{\text{noise,max}} \geq 4°$ beim Verfahren mit variabler Referenz-Pose geringer (siehe Abb. 3.4 [unten]).

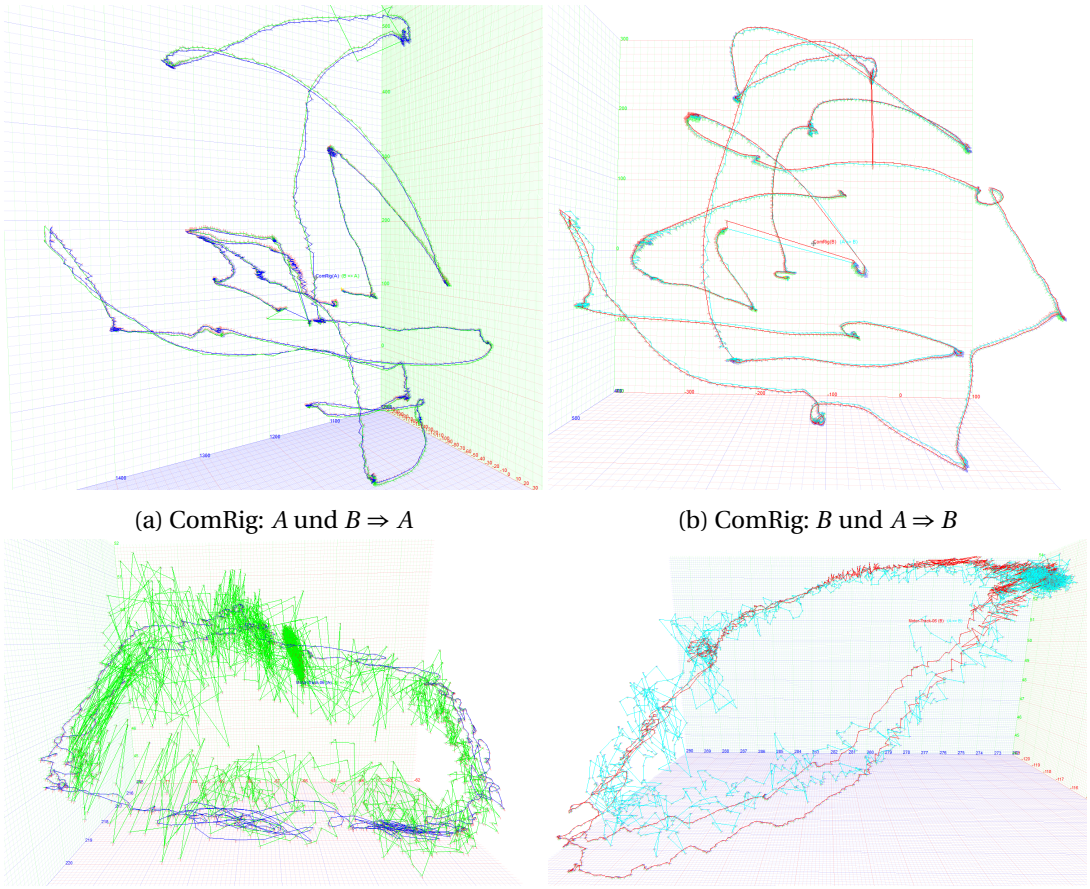
3.4.2 Ergebnisse: reale Datensätze

Neben den zuvor untersuchten künstlich erzeugten Datensätzen wurden auch einige reale Messreihen untersucht. Eine Messreihe („ComRig“) wurde erzeugt, indem ein Ultraschall-Marker und ein IR-Marker, die an einer Stange fest miteinander verbunden waren, durch den Raum bewegt wurden. Beide Marker wurden dabei von jeweils einem Sensor (Ultraschall-Mikrofon-Array, bzw. IR-Kamera) mit 50 Hz (weitestgehend) zeitsynchron getrackt. Diese Messreihe besteht aus 3825 Posen-Paaren und weist relativ geringes Rauschen auf (siehe Abb. 3.6 (a), (b)). 13 weitere Messreihen („Motor-Track-xx“) wurden auf folgende Weise erzeugt [47]: Zwei IR-Kameras wurden fest an der Karosserie eines PKWs befestigt und filmten jeweils eine Anordnung aus IR-LEDs (Marker), welche fest am Motor des Fahrzeugs angebracht waren. Auf diese Weise kann man die Vibrationen des Motors während des Betriebs des Fahrzeugs messen. Die Messwerte wurden hierbei wieder (weitestgehend) zeitsynchron mit ca. 120 Hz aufgenommen. Durch die sehr schnellen und starken Vibrationen des Motors bei gleichzeitig relativ geringer Gesamtauslenkung sind diese Messreihen sehr stark mit Rauschen behaftet (siehe Abb. 3.6 (c), (d)).

Die Ergebnisse sind in Abb. 3.7 dargestellt. Die Rotations-Fehler (siehe Abb. 3.7 [oben]) sind bei allen drei untersuchten Sensordaten-Fusions-Verfahren für alle Messreihen noch sehr ähnlich – auch wenn das Quaternion-AX=YB-Verfahren auch hier bereits stets etwas bessere Ergebnisse liefert. Beim Translations-Fehler (siehe Abb. 3.7 [unten]) wird jedoch deutlich, dass das Quaternion-AX=YB-Verfahren dem AX=XB-Verfahren klar überlegen ist. Bei der „ComRig“-Messreihe, welche relativ wenig Rauschen aufweist, sind die Ergebnisse der drei Verfahren noch fast identisch. Bei den sehr stark verrauschten „Motor-Tracks“ erkennt man jedoch deutlich, dass das Quaternion-AX=YB-Verfahren erheblich besser funktioniert. Hierin zeigt sich deutlich, dass die Abhängigkeit von einem Referenz-Posenpaar gerade bei stark verrauschten Daten ein großer Nachteil ist, was sich insbesondere im von der Fehler-Fortpflanzung betroffenen translatorischen Anteil der Lösungen widerspiegelt.

Betrachtet man die sehr stark verrauschten Motor-Track-Messreihen (siehe Abb. 3.6 (c) (d)) so lässt sich hier erkennen, dass das Quaternion-AX=YB-Verfahren trotz des sehr starken Sensor-Rauschens dennoch in der Lage ist, ein gutes Alignment der beiden Messreihen zu finden: Die originale Kurve und die transformierte Kurve liegen jeweils sichtbar übereinander. Damit ist erkennbar, dass die globalen (für jeden Punkt der Zeitreihen gleichen) starren Transformationen S und T korrekt bestimmt werden konnten. Das Sensor-Rauschen verursacht bei der für die Motor-Track-Daten verwendeten Messmethode Messfehler im Mil-

limeter-Bereich [47]. Zieht man in Betracht, dass die komplette Bewegung bei den Motor-Track-Daten eine Gesamt-Auslenkung von nur ca. drei bis vier Zentimetern aufweist, wird deutlich, wie stark das Sensor-Rauschen diese Bewegungen dominiert. Die Tatsache, dass das Quaternion-AX=YB-Verfahren bei allen Motor-Track-Daten sehr geringe Translations- und Rotations-Fehler erreicht (siehe Abb. 3.7) beweist dessen Robustheit gegenüber Sensor-Rauschen.



(a) ComRig: A und $B \Rightarrow A$

(b) ComRig: B und $A \Rightarrow B$

(c) Motor-Track-06: A und $B \Rightarrow A$

(d) Motor-Track-06: B und $A \Rightarrow B$

Abbildung 3.6: ComRig-Messreihe [(a), (b)] und Motor-Track-06-Messreihe [(c), (d)]. Dargestellt sind jeweils die gemessenen Zeitreihen A (blau) und B (rot), sowie die jeweils mit der besten gefundenen Transformation transformierten Zeitreihen $B \Rightarrow A$ (grün) und $A \Rightarrow B$ (cyan). Diese besten Transformations-Ergebnisse lieferte dabei das Quaternion-AX=YB-Verfahren. Man erkennt deutlich, dass die Motor-Track-Messreihen (hier exemplarisch: Motor-Track-06) sehr stark rauschbehaftet sind, während die ComRig-Messreihe relativ wenig Rauschen aufweist. Dennoch wird ein relativ gutes Alignment auch bei den Motor-Track-Daten erreicht.

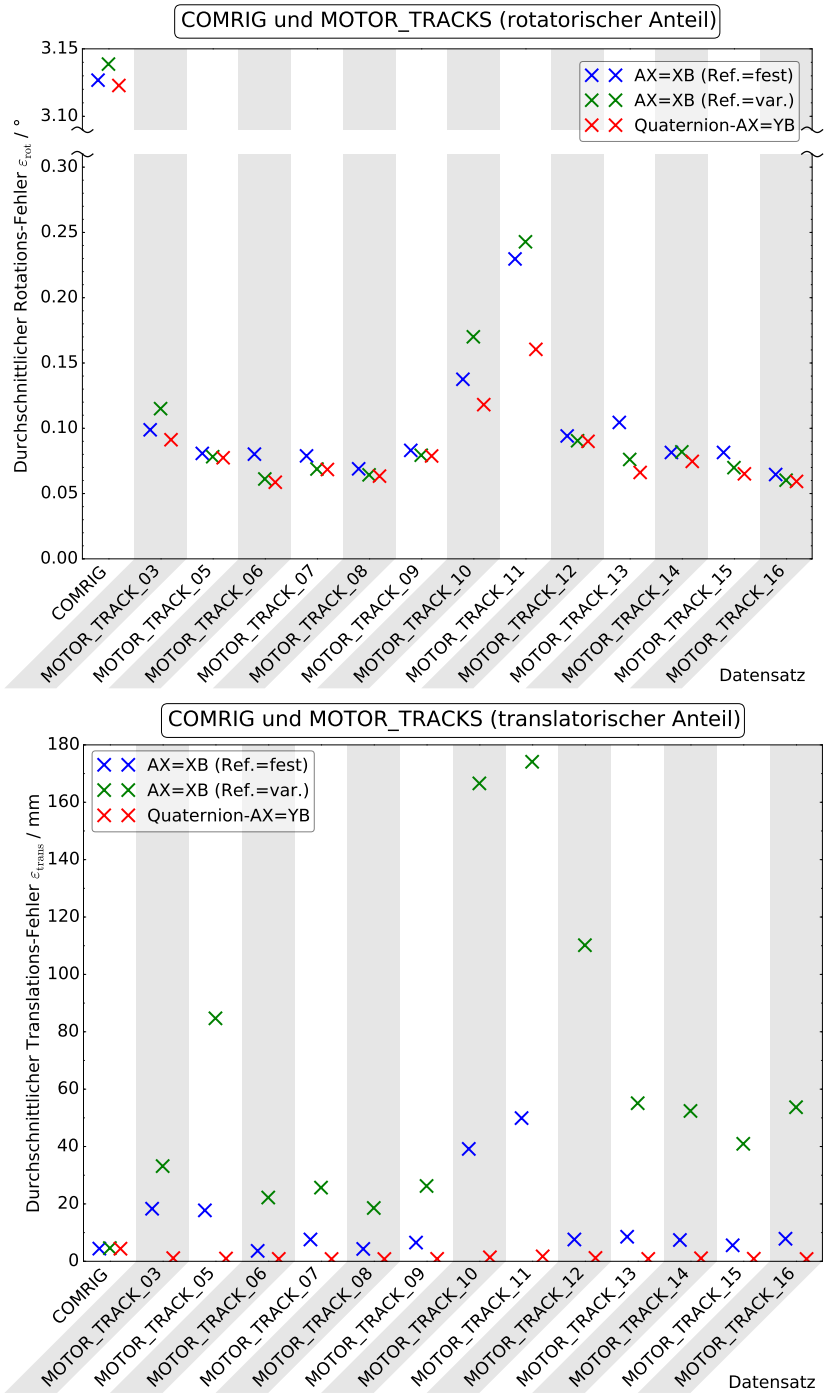


Abbildung 3.7: Rotations- und Translations-Fehler der mit den unterschiedlichen Verfahren bestimmten Transformationen S und T auf den realen Messreihen. Dargestellt ist der durchschnittliche Rotations-Fehler pro Zeitpunkt **[oben]** und der durchschnittliche Translations-Fehler pro Zeitpunkt **[unten]**.

3.5 Synchronisation

Bei den bisherigen Untersuchungen wurde angenommen, dass die beiden Zeitreihen A und B zeitlich exakt synchronisiert sind. Dies ist allerdings bei realen Messungen nicht immer der Fall. Arbeiten zwei Messsysteme wie z.B. eine IR-Kamera und ein Ultraschall-Sensor oder zwei verschiedene Kamera-Modelle mit unterschiedlichen Messfrequenzen (Frame-Raten), so müssen die Messreihen zuerst synchronisiert werden. Ein anderes Problem ist der Startzeitpunkt. Selbst wenn beide Messsysteme die gleiche Frame-Rate besitzen, kann durch unterschiedliche Startzeiten der Messungen ein zeitlicher Offset zwischen den beiden Messungen entstehen. Ohne eine gemeinsame Zeitbasis der Messsysteme ist dieser Offset jedoch nicht a priori bekannt. Während ein solcher Offset noch durch eine einfache Verschiebung einer Zeitreihe gegen die andere kompensiert werden kann, ist eine Synchronisation der beiden Zeitreihen im Falle unterschiedlicher Messraten oder gar bei Messausfällen nicht so einfach durchführbar.

Das Grund-Problem ist nun Folgendes: Um zwei Zeitreihen A und B via AX=YB in ein gemeinsames Koordinatensystem zu bringen, müssen diese bereits zeitlich synchronisiert sein. Um jedoch zwei Zeitreihen zu synchronisieren, müssen die Daten bereits in einem gemeinsamen Koordinatensystem vorliegen. Um dieses Problem zu umgehen, machen wir nun folgenden Ansatz. Dazu rufen wir uns noch einmal in Erinnerung, dass die **inkrementellen Rotationen** einer Bewegungs-Zeitreihe **unabhängig vom Bezugssystem** sind (siehe Beweis 2 im vorangehenden Kapitel).

Dabei ergibt sich jedoch folgendes Problem: Unterscheiden sich zwei Rotations-Zeitreihen lediglich durch eine Rotation, welche **von einer Seite** wirkt, sodass gilt:

$$\Theta_{A,i} = \Theta_{B,i} \cdot \Theta_X$$

dann gilt der in Beweis 2 nachgewiesene Zusammenhang: Die inkrementellen Rotationen $\Theta'_{B,i} := \Theta_{B,i+h} \cdot \Theta_{B,i}^{-1}$ sind dabei unabhängig von Θ_X. Hier haben wir es allerdings mit einer Wirkung **von links und von rechts** zu tun:

$$\Theta_{A,i} = \Theta_Y \cdot \Theta_{B,i} \cdot \Theta_X^{-1}$$

Für diesen Zusammenhang gilt im Allgemeinen nicht, dass die inkrementellen Rotationen unabhängig von Θ_X und Θ_Y sind. Allerdings kann man folgende Überlegung anstellen: Da die Rotationen (unabhängig von ihrer konkreten Darstellung als Rotations-Quaternionen oder als Rotations-Matrizen) Gruppen-Elemente der $SO(3)$ sind, lässt sich für jeden Zeitpunkt i eine Rotation $\Theta_{Z,i}$ finden, für die Folgendes gilt:

$$\Theta_Y \cdot \Theta_{B,i} \cdot \Theta_X^{-1} =: \Theta_{B,i} \cdot \Theta_{Z,i}$$

$\Theta_{Z,i}$ ist jedoch nicht konstant, da es nicht nur von den Konstanten Θ_X und Θ_Y, sondern auch von der Rotation $\Theta_{B,i}$ abhängt. Allerdings können wir für *langsame* Bewegungen –

also für Bewegungen, die pro Zeitpunkt jeweils nur geringe Änderungen der Orientierung aufweisen – annehmen, dass $\Theta_{Z,i} \approx \Theta_{Z,i+h}$. Deshalb sollte die Schrittweite h der inkrementellen Rotation für dieses Verfahren so klein wie möglich gewählt werden. Nehmen wir dies an, so lässt sich analog zu Beweis 2 zeigen, dass die inkrementellen Rotationen *näherungsweise* unabhängig von der Koordinaten-Transformation sind:[6]

Beweis 4

Seien $\Theta_1, \Theta_2 \in SO(3)$ zwei Rotationen. Dann ist die inkrementelle Rotation $\Delta\Theta$, die einen Körper aus der Lage Θ_1 in die Lage Θ_2 überführt, gegeben durch:

$$\Delta\Theta = \Theta_2 \cdot \Theta_1^{-1}$$

Seien nun $\tilde{\Theta}_1, \tilde{\Theta}_2, \Theta_X, \Theta_Y \in SO(3)$ mit

$$\tilde{\Theta}_1 := \Theta_Y \cdot \Theta_1 \cdot \Theta_X^{-1} =: \Theta_1 \cdot \Theta_{Z,1}$$

und

$$\tilde{\Theta}_2 := \Theta_Y \cdot \Theta_2 \cdot \Theta_X^{-1} =: \Theta_2 \cdot \Theta_{Z,2}$$

die selben Rotationen, jedoch gemessen in einem anderen, durch die Transformationen Θ_X und Θ_Y bestimmten Bezugssystem. Der Unterschied der Orientierungen Θ_1 und Θ_2 bzw. $\tilde{\Theta}_1$ und $\tilde{\Theta}_2$ sei hierbei klein genug, sodass gilt: $\Theta_{Z,1} \approx \Theta_{Z,2}$. Die inkrementelle Rotation $\Delta\tilde{\Theta}$ ist dann gegeben als

$$\begin{aligned}
\Delta\tilde{\Theta} = \tilde{\Theta}_2 \cdot \tilde{\Theta}_1^{-1} &= (\Theta_2 \cdot \Theta_{Z,2}) \cdot (\Theta_1 \cdot \Theta_{Z,1})^{-1} \\
= (\Theta_2 \cdot \Theta_{Z,2}) \cdot (\Theta_{Z,1}^{-1} \cdot \Theta_1^{-1}) &= \Theta_2 \cdot (\Theta_{Z,2} \cdot \Theta_{Z,1}^{-1}) \cdot \Theta_1^{-1} \\
\approx \Theta_2 \cdot \mathbb{1} \cdot \Theta_1^{-1} &= \Theta_2 \cdot \Theta_1^{-1} \\
\approx \Delta\Theta & \qquad \square
\end{aligned}$$

$(\cdot)$ ist hierbei das gewöhnliche Matrizen-Produkt auf $\mathbb{R}^{3\times3}$.

Damit lässt sich eine zeitliche Synchronisation der 6-DoF-Zeitreihen realisieren, indem ein Alignment per Dynamic Time Warping (siehe vorangehendes Kapitel) auf den *inkrementellen Rotationen* der beiden Zeitreihen bestimmt wird. Mit dem auf diese Weise bestimmten Warping-Pfad werden dann beide 6-DoF-Zeitreihen aligniert. Die somit synchronisierten Zeitreihen können im Anschluss per AX=YB in ein gemeinsames Koordinatensystem transformiert werden. Damit können wir das Vorgehen nun wie folgt beschreiben:

Gegeben: zwei zeitlich noch nicht synchronisierte Zeitreihen $A = \{A_i\}$ und $B = \{B_i\}$ mit $A_i, B_i \in SE(3)$. Gesucht: zwei Transformationen $S, T \in SE(3)$, sodass $A_i T \approx S B_i \; \forall i$.

[6] Dieser Beweis wird in der Darstellung der Rotationen als Rotations-Matrizen $\Theta \in SO(3)$ geführt. In der Darstellung als Rotations-Quaternionen $q \in \mathbb{H}_1$ (wie in Beweis 2) gilt dieser jedoch analog.

- Bilde aus den Rotations-Anteilen der Zeitreihen A und B die Zeitreihen der inkrementellen Rotationen q'_A und q'_B (gemäß Definition 25 aus dem vorangehenden Kapitel).
- Finde ein optimales zeitliches Alignment (einen optimalen Warping-Pfad γ_{opt}) auf den Zeitreihen der inkrementellen Rotationen q'_A und q'_B mit Hilfe von DTW.
- Verwende das gefundene Alignment γ_{opt} zur Bestimmung der synchronisierten 6-DoF-Zeitreihen A_{sync} und B_{sync}.
- Finde S und T via AX=YB auf den synchronisierten Zeitreihen A_{sync} und B_{sync}.

Die in Beweis 4 geforderte langsame Änderung des rotatorischen Anteils, die eine zwingende Voraussetzung für die Anwendbarkeit des Synchronisations-Verfahrens darstellt, wurde für alle hier untersuchten Datensätze – sowohl die realen, als auch die künstlichen – überprüft. Selbst bei den im translatorischen Anteil sehr stark verrauschten Motor-Track-Daten ist der rotatorische Anteil stabil genug, um diese Voraussetzung zu erfüllen. Damit ist das Synchronisations-Verfahren für alle untersuchten Datensätze anwendbar.

Die Ergebnisse für die bereits zuvor in diesem Abschnitt untersuchten Messreihen („ComRig“ und „Motor-Track-xx“) sind in Abb. 3.8 dargestellt. Die Fehler-Differenz

$$\Delta\varepsilon_{\text{rot}} := \varepsilon_{\text{rot,orig}} - \varepsilon_{\text{rot,sync}} \quad \text{bzw.} \quad \Delta\varepsilon_{\text{trans}} := \varepsilon_{\text{trans,orig}} - \varepsilon_{\text{trans,sync}}$$

gibt dabei an, welches Verfahren bessere Ergebnisse liefert. $\Delta\varepsilon > 0$ bedeutet, dass der (Rotations- bzw. Translations-)-Fehler bei den zuvor synchronisierten Zeitreihen geringer ist – die Synchronisation also zu **besseren Ergebnissen** führt. Entsprechend bedeutet $\Delta\varepsilon < 0$, dass die Synchronisation zu **schlechteren Ergebnissen** führt.

Bei der schwach verrauschten ComRig-Messreihe führt die Synchronisation zu einer deutlichen Verbesserung im rotatorischen Anteil (Abb. 3.8, [oben]), während der translatorische Anteil (Abb. 3.8, [unten]) fast unverändert bleibt. Bei den stark verrauschten Motor-Track-Messreihen ist keine einheitliche Verbesserung oder Verschlechterung erkennbar. Besonders beim AX=XB-Verfahren mit variablem Referenz-Posenpaar gibt es beim translatorischen Anteil zum Teil starke Verbesserungen (Motor-Tracks: 05, 12, und 16) aber auch zum Teil starke Verschlechterungen (Motor-Tracks: 10 und 13). Allerdings war der absolute Fehler bei diesem Verfahren im Vergleich zu den anderen ohnehin deutlich höher (vergleiche Abb. 3.7). Bei den anderen beiden Verfahren (AX=XB mit festem Referenz-Posenpaar und Quaternion-AX=YB) sind auf den Motor-Track-Messreihen keine großen Unterschiede zwischen der Verwendung der synchronisierten und der nicht synchronisierten Zeitreihen erkennbar.

Um den Einfluss von Rauschen auf die Qualität der Ergebnisse bezüglich der Verwendung

der synchronisierten und der nicht synchronisierten Zeitreihen näher zu untersuchen, wurden erneut künstliche Datensätze aus den JGU_Numbers- und 6DMG-Datensätzen – analog zu Abschnitt 3.4.1 – generiert. Auch hier wurde das Rauschen im translatorischen Anteil wieder auf $[-10\text{mm}, +10\text{mm}]$ beschränkt. Für das rotatorische Rauschen wurde wieder $\varphi_{\text{noise,max}} \in [0°, 20°]$ in Schritten von jeweils $1°$ durchgefahren. Um nun den Effekt der Synchronisation untersuchen zu können, wurden die Zeitreihen zusätzlich Index-weise um einen Shift $s \in \mathbb{Z}$ gegeneinander verschoben. Dieser Shift wurde für jedes generierte Zeitreihenpaar zufällig uniform im Intervall $s \in [-10, +10]$ gewählt. Das Vorzeichen von s drückt dabei aus, in welche Richtung die beiden Zeitreihen gegeneinander verschoben werden: $s > 0$ bedeutet, dass Zeitreihe A später beginnt und umgekehrt bedeutet $s < 0$, dass Zeitreihe B später beginnt.

Die Ergebnisse sind in Abb. 3.9 und 3.10 dargestellt[7]. Es werden wieder Mittelwert und Standard-Fehler der Transformations-Fehler-Differenzen $\Delta\varepsilon$ über alle verwendeten[7] Shapes des jeweiligen Datensatzes dargestellt. Man erkennt, dass eine Verbesserung der Transformations-Fehler für geringes Rauschen stattfindet, welche mit zunehmendem Rauschen immer kleiner wird. Im Allgemeinen ist das Vorgehen, die Zeitreihen vor der eigentlichen Transformations-Berechnung auf die hier beschriebene Art und Weise zu synchronisieren, also durchaus geeignet, um die Ergebnisse der Transformation zu verbessern. Die hierbei künstlich generierten zeitlichen Verschiebungen sind jedoch nur der einfachste Fall von asynchron aufgenommenen Daten. Möglicherweise vorhandene unterschiedliche Messfrequenzen oder sogar zeitliche Variationen der Messfrequenzen können jedoch – durch die Verwendung von Dynamic Time Warping im Synchronisations-Schritt – ebenfalls kompensiert werden.

[7] Die Betrachtung des AX=XB-Verfahrens mit variablem Referenz-Posenpaar wurde hierbei bewusst nicht einbezogen, da dieses recht instabil war und es vermehrt zu Fehlern bei der Berechnung der Matrix-Wurzel (Gleichung (3.5)), sowie der Matrix-Logarithmen (Definition 26) mit $\text{Tr}(\Theta) = -1$ kam. Bei der Verwendung des AX=XB-Verfahrens mit festem Referenz-Posenpaar mussten aus diesem Grund jedoch ebenfalls einige Shapes ausgelassen werden.

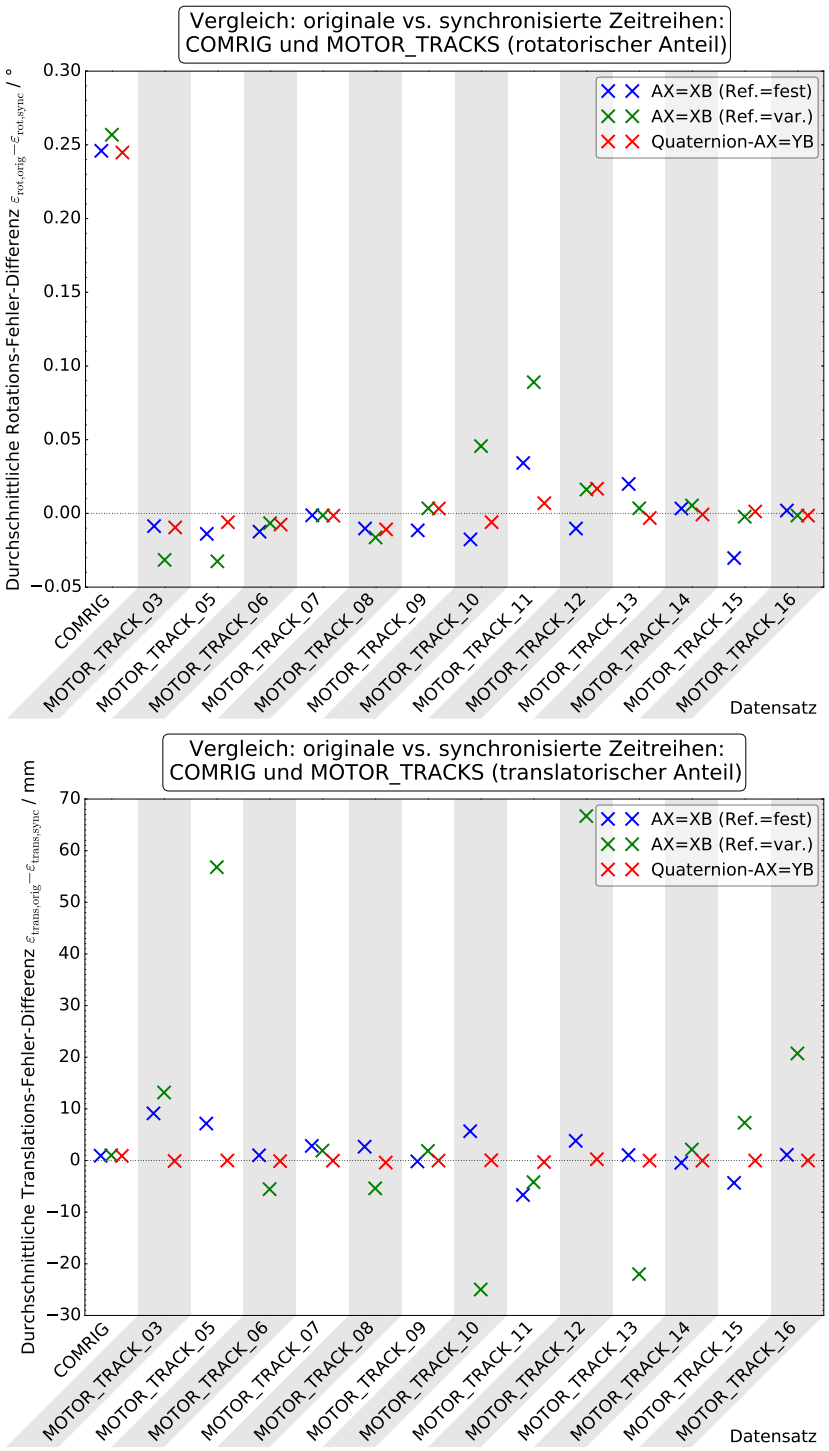


Abbildung 3.8: Rotations- und Translations-Fehler-Differenz zwischen originalen und synchronisierten Zeitreihen der mit den unterschiedlichen Verfahren bestimmten Transformationen S und T auf den realen Messreihen. Dargestellt ist die durchschnittliche Rotations-Fehler-Differenz pro Zeitpunkt **[oben]** und die durchschnittliche Translations-Fehler-Differenz pro Zeitpunkt **[unten]**.

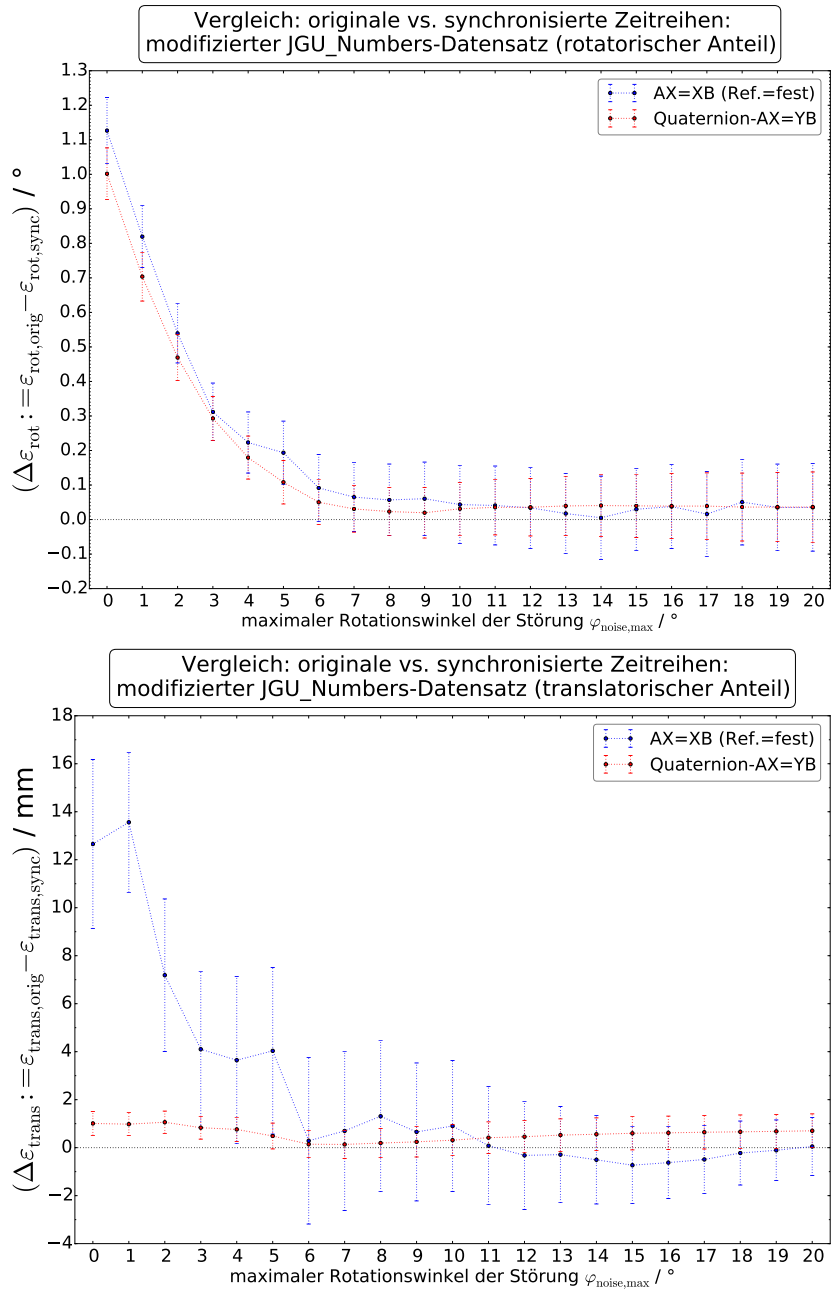


Abbildung 3.9: Rotations- und Translations-Fehler-Differenz der mit den unterschiedlichen Verfahren bestimmten Transformationen S und T auf den künstlich transformierten Shapes des JGU_Numbers-Datensatzes. Dargestellt sind Mittelwert und Standard-Fehler (als Fehler-Balken) der Rotations-Fehler-Differenz **[oben]** und der Translations-Fehler-Differenz **[unten]** über alle 2003 Shapes des JGU_Numbers-Datensatzes jeweils gegen den maximalen Rotationswinkel der Störung $\varphi_{\mathrm{noise,max}}$ (siehe Text).

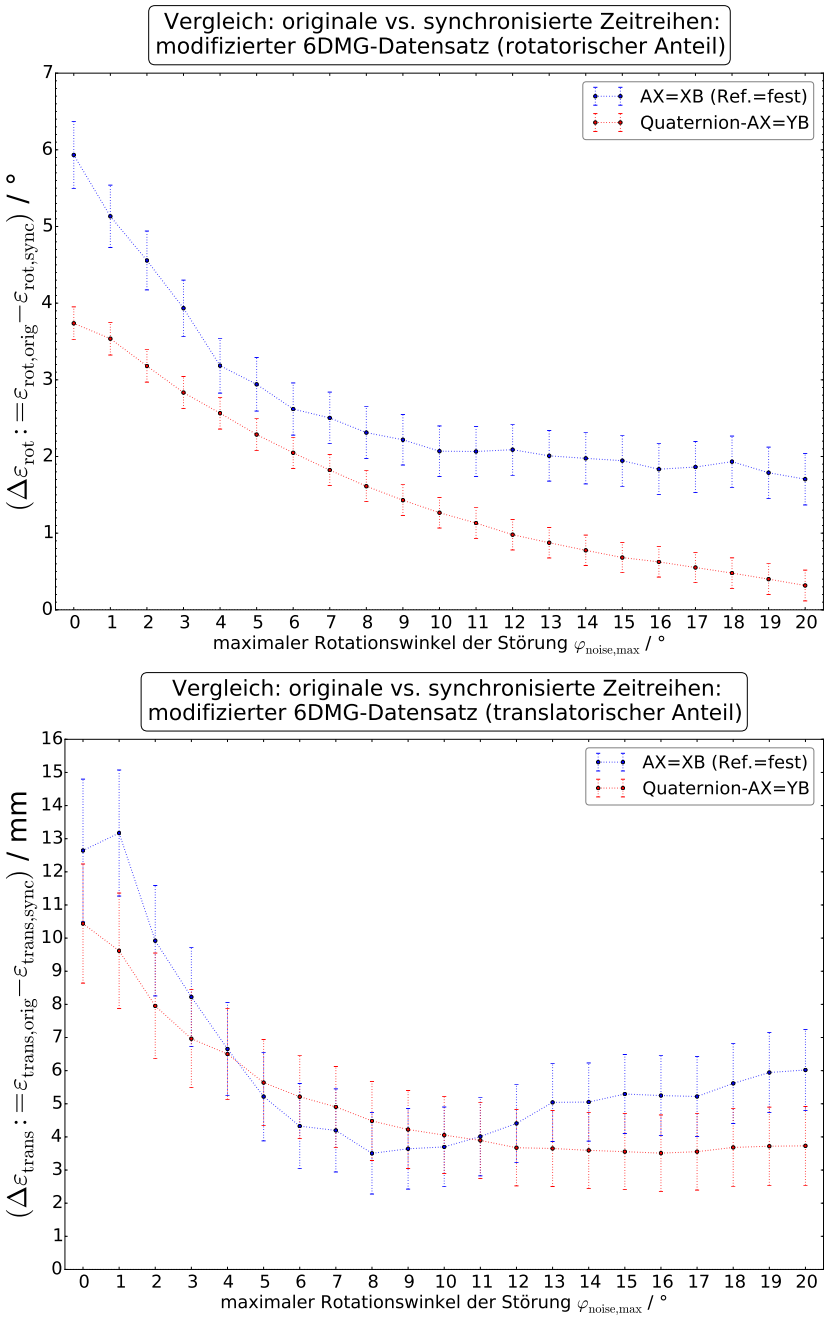


Abbildung 3.10: Rotations- und Translations-Fehler-Differenz der mit den unterschiedlichen Verfahren bestimmten Transformationen S und T auf den künstlich transformierten Shapes des 6DMG-Datensatzes. Dargestellt sind Mittelwert und Standard-Fehler (als Fehler-Balken) der Rotations-Fehler-Differenz **[oben]** und der Translations-Fehler-Differenz **[unten]** über 5574 Shapes des 6DMG-Datensatzes jeweils gegen den maximalen Rotationswinkel der Störung $\varphi_{\mathrm{noise,max}}$ (siehe Text).

3.6 Zusammenfassung

In diesem Kapitel wurden drei verschiedene[8] Verfahren zur Lösung des Sensordaten-Fusions-Problems der Form AX=YB miteinander verglichen. Als Test-Daten wurden dabei Messreihen aus sehr unterschiedlichen Szenarien verwendet, wobei sowohl reale Messdaten, als auch künstlich generierte Daten untersucht wurden. Es stellte sich dabei heraus, dass die Abhängigkeit der beiden Varianten des AX=XB-Lösungsansatzes von einem Referenz-Posenpaar – insbesondere bei stark verrauschten Daten – zu erheblich schlechteren Ergebnissen führt im Vergleich zum Quaternion-AX=YB-Verfahren, welches kein Referenz-Posenpaar benötigt. Dieses konnte sogar die extrem stark verrauschten Motor-Track-Daten sehr gut und mit geringen Fehlern im translatorischen und rotatorischen Anteil alignieren. Des Weiteren ist das Quaternion-AX=YB-Verfahren auch numerisch stabiler als die untersuchten AX=XB-Varianten, da es ohne Matrix-Logarithmus und Matrix-Wurzel auskommt, welche bei den AX=XB-Verfahren in manchen Fällen dazu führen, dass keine gültige Lösung bestimmt werden kann. Außerdem ist die Laufzeit der AX=XB-Verfahren deutlich höher, da diese alleine für die Bestimmung des optimalen Wertes des freien Parameters um einen Faktor der Zeitreihenlänge mehr Rechenzeit benötigt als das parameterfreie Quaternion-AX=YB Verfahren.

Da reale Messreihen nicht immer perfekt synchronisiert sind, wurde ein Preprocessing-Verfahren zur zeitlichen Synchronisierung zweier Zeitreihen auf Basis von Dynamic Time Warping vorgestellt. Dieses ist in der Lage, zwei Zeitreihen zu synchronisieren, die nicht in einem gemeinsamen Koordinatensystem gemessen wurden. Dies führt – insbesondere bei nicht allzu stark verrauschten Daten – zu einer deutlichen Verbesserung der Transformations-Ergebnisse. Voraussetzung für die Anwendbarkeit dieses Verfahrens ist allerdings, dass die aufgezeichneten Bewegungen im Vergleich zur Messfrequenz nicht zu schnell ablaufen.

In Tabelle 3.1 sind zum Abschluss noch einmal die Transformations-Fehler der realen Messreihen – getrennt für den rotatorischen und den translatorischen Anteil – aufgeführt. Die obere Tabelle zeigt hierbei die Ergebnisse für die originalen (nicht synchronisierten) Zeitreihen, während die untere Tabelle die Ergebnisse für die zuvor synchronisierten Zeitreihen enthält. Die jeweils besseren Ergebnisse im direkten Vergleich zwischen synchronisierten und nicht synchronisierten Zeitreihen sind in grün dargestellt. Die zusätzlich hellgrün hinterlegten Werte sind die besten Ergebnisse im Vergleich der drei verwendeten Sensordaten-Fusions-Verfahren.

[8] Quaternion-AX=YB, sowie zwei Varianten von AX=XB

Datensatz	AX=XB (Ref.=fest)		AX=XB (Ref.=var.)		Quaternion-AX=YB	
(original)	ε_{rot} / °	$\varepsilon_{\text{trans}}$ / mm	ε_{rot} / °	$\varepsilon_{\text{trans}}$ / mm	ε_{rot} / °	$\varepsilon_{\text{trans}}$ / mm
COMRIG	3.127	4.424	3.139	4.610	3.123	4.390
MOTOR_TRACK_03	0.099	18.326	0.115	33.181	0.091	1.112
MOTOR_TRACK_05	0.081	17.758	0.078	84.692	0.077	0.971
MOTOR_TRACK_06	0.080	3.634	0.061	22.226	0.059	0.834
MOTOR_TRACK_07	0.079	7.604	0.069	25.682	0.068	0.802
MOTOR_TRACK_08	0.069	4.313	0.064	18.581	0.063	0.790
MOTOR_TRACK_09	0.083	6.525	0.079	26.252	0.079	0.851
MOTOR_TRACK_10	0.138	39.174	0.170	166.583	0.118	1.393
MOTOR_TRACK_11	0.230	49.916	0.243	174.113	0.160	1.720
MOTOR_TRACK_12	0.094	7.605	0.090	110.183	0.090	1.167
MOTOR_TRACK_13	0.104	8.554	0.076	55.106	0.066	0.832
MOTOR_TRACK_14	0.081	7.423	0.082	52.405	0.075	1.056
MOTOR_TRACK_15	0.081	5.579	0.070	40.949	0.065	0.888
MOTOR_TRACK_16	0.065	7.860	0.060	53.710	0.059	0.802

Datensatz	AX=XB (Ref.=fest)		AX=XB (Ref.=var.)		Quaternion-AX=YB	
(synchronisiert)	ε_{rot} / °	$\varepsilon_{\text{trans}}$ / mm	ε_{rot} / °	$\varepsilon_{\text{trans}}$ / mm	ε_{rot} / °	$\varepsilon_{\text{trans}}$ / mm
COMRIG	2.881	3.501	2.882	3.590	2.878	3.496
MOTOR_TRACK_03	0.107	9.195	0.147	20.010	0.101	1.201
MOTOR_TRACK_05	0.095	10.597	0.111	27.887	0.083	0.968
MOTOR_TRACK_06	0.093	2.620	0.068	27.770	0.066	0.959
MOTOR_TRACK_07	0.080	4.774	0.070	23.767	0.070	0.831
MOTOR_TRACK_08	0.079	1.633	0.081	23.968	0.074	1.189
MOTOR_TRACK_09	0.095	6.699	0.076	24.393	0.075	0.871
MOTOR_TRACK_10	0.155	33.497	0.124	191.559	0.124	1.337
MOTOR_TRACK_11	0.195	56.586	0.154	178.295	0.154	2.025
MOTOR_TRACK_12	0.104	3.790	0.074	43.477	0.073	0.912
MOTOR_TRACK_13	0.085	7.494	0.073	77.102	0.069	0.844
MOTOR_TRACK_14	0.078	7.870	0.077	50.278	0.075	1.077
MOTOR_TRACK_15	0.112	9.910	0.072	33.631	0.064	0.895
MOTOR_TRACK_16	0.063	6.757	0.062	32.973	0.061	0.793

Tabelle 3.1: Transformations-Ergebnisse AX=YB. Angegeben ist der Transformations-Fehler (gemäß Definition 27) – jeweils getrennt für den rotatorischen und den translatorischen Anteil. In grün dargestellt sind die besten Ergebnisse (also die niedrigsten Fehler) jeweils im Vergleich: originale Zeitreihen **[obere Tabelle]** vs. im Preprocessing synchronisierte Zeitreihen **[untere Tabelle]**. Zusätzlich hellgrün hinterlegt sind die besten Ergebnisse für die jeweilige Messreihe insgesamt von allen Transformations-Verfahren (AX=XB mit festem Referenz-Posenpaar, AX=XB mit variablem Referenz-Posenpaar und Quaternion-AX=YB).

KAPITEL 4

ZUSAMMENFASSUNG

Zusammenfassung

Diese Arbeit beschäftigte sich im Wesentlichen mit der Klassifikation von Bewegungszeitreihen. Außerdem wurde ein Verfahren zur Stream-Segmentierung mehrdimensionaler Zeitreihen vorgestellt, sowie die Sensordaten-Fusion von Bewegungszeitreihen behandelt. Im Folgenden werden die Ergebnisse der einzelnen Kapitel noch einmal zusammengefasst.

In Kapitel 1 wurden nach einer kurzen Behandlung der theoretischen Grundlagen die verwendeten Datensätze vorgestellt. Neben öffentlich zugänglichen Datensätzen wurde im Rahmen dieser Arbeit ein weiterer Bewegungs-Datensatz erzeugt, bei dem das Zeichnen von Ziffern in 6 Freiheitsgraden aufgezeichnet wurde.

In Kapitel 2 wurde zuerst ein Verfahren vorgestellt, welches einen vorhandenen Bewegungs-Datenstrom (semi-)automatisiert in Teilsequenzen segmentieren kann. Dieses auf Continuous-Wavelet-Transformation basierende Verfahren ist in der Lage, eine grobe Vorsegmentierung des Datenstromes vorzunehmen. Eine anschließende manuelle Überprüfung und gegebenenfalls Korrektur der Segmentierung ist im Allgemeinen jedoch erforderlich, da die Ergebnisse stark von den verwendeten Parametern abhängig sind.
Im Anschluss wurde eine modifizierte Variante der von Kim et al. in [24] eingeführten Lower Bound – welche in konstanter Zeit ausgewertet werden kann – auf Quaternionen-wertige Zeitreihen erweitert. Zusätzlich wurde eine von Keogh et al. in [23] eingeführte, schärfere Lower Bound – welche in linearer Zeit mit der Länge der Zeitreihe auswertbar ist – ebenfalls auf Quaternionen-wertige Zeitreihen erweitert. Diese erreichte auf den Test-Datensätzen Pruning-Raten von über 80%. Zusätzlich wurde eine schnelle Approximation an diese Lower Bound – basierend auf Intervall-Arithmetik – entwickelt, die um etwa einen Faktor 70 geringere Laufzeiten benötigt, bei einer im Mittel über alle getesteten Datensätze weniger als 3% geringeren Pruning-Rate im Vergleich zur exakten Lower Bound. Zusätzlich dazu wurde eine Lower-Bound-Kaskade der beiden erweiterten Lower Bounds von Kim und Keogh untersucht, mit der die Gesamt-Pruning-Rate um weitere bis zu 4% erhöht werden konnte.
Im Hauptteil dieses Kapitels wurden zunächst drei Zeitreihen-Abstandsmaße (Lockstep, (Sakoe-Chiba) constrained DTW und subsequence DTW) miteinander vergleichen. Dazu wurde deren Klassifikations-Fehler auf den im ersten Kapitel vorgestellten Test-Datensätzen bestimmt. Dabei stellte sich das elastische constrained DTW mit seiner Eigenschaft des globalen Alignments als das am besten für die Shape-basierte One-Nearest Neighbor-(1NN-)Klassifikation geeignete Zeitreihen-Abstandsmaß heraus. Ebenso wurden drei punktweise Abstandsmaße (geodätischer Abstand auf Quaternionen, Euklidische Norm auf Quaternionen und Frobenius-Norm auf Rotationsmatrizen) miteinander verglichen, wobei sich das geodätische Abstandsmaß (zumindest in den meisten Fällen) als das beste erwies. Dieses ist auch das physikalisch sinnvollste Abstandsmaß zwischen zwei Rotationen, da es exakt den Winkel zwischen zwei Orientierungen berechnet.

Im Anschluss daran wurde ein neuer, bisher nicht bekannter Ansatz zur Klassifikation anhand von inkrementellen Rotationen betrachtet. Dieser lieferte in den meisten Fällen bessere Klassifikations-Ergebnisse. Lediglich beim Opportunity-Datensatz wurden die Ergebnisse mit diesem Ansatz deutlich schlechter. Allerdings erwies sich der Opportunity-Datensatz im Allgemeinen als sehr schwierig bezüglich der hier untersuchten 1NN-Klassifikation, da die einzelnen Instanzen jeder Bewegungsklasse sehr unterschiedlich in ihrer Durchführung und damit generell nicht gut per 1NN klassifizierbar waren. Der künstlich erzeugte qCBF-Datensatz zeigte besonders die Stärken der Klassifikation anhand der inkrementellen Rotationen: Durch die vom gewählten Koordinatensystem unabhängige Darstellung der inkrementellen Rotationen ist eine deutlich bessere Klassifikation möglich.

In Kapitel 3 wurden anschließend drei Verfahren zur Sensordaten-Fusion miteinander verglichen. Die Aufgabe bestand darin, zwei gegebene Zeitreihen aus 6-DoF-Posen in ein gemeinsames Koordinatensystem zu überführen. Untersucht wurden zwei Varianten eines AX=XB-Lösungsverfahrens [38] und ein auf der Rotation von Quaternionen basierendes AX=YB-Lösungsverfahren [43]. Diese wurden auf sehr unterschiedlichen Datensätzen mit sehr unterschiedlich stark ausgeprägtem Rauschen getestet. Es stellte sich dabei heraus, dass die Abhängigkeit der AX=XB-Varianten von einem Referenz-Posenpaar zu deutlich schlechteren Ergebnissen führte – insbesondere bei stark verrauschten Daten. Des Weiteren war das AX=XB-Verfahren numerisch instabil und lieferte bei bestimmten Eingabe-Daten ungültige Ergebnisse. Das untersuchte Quaternion-AX=YB-Verfahren wies keine solche numerische Instabilität auf und war zudem auch in der Lage, stark verrauschte Zeitreihen in ein gemeinsames Koordinatensystem zu überführen. Außerdem haben die AX=XB-Varianten eine um einen Faktor der Zeitreihenlänge größere Laufzeit, da der optimale Wert eines freien (diskreten) Parameters per Grid-Search, also dem Austesten aller möglichen Parameterwerte, bestimmt werden muss. Da das Quaternion-AX=YB-Verfahren parameterfrei ist, entfällt dieser Schritt dort.
Abschließend wurde ein neuartiges Verfahren vorgestellt, welches in der Lage ist, zwei nicht synchronisierte Zeitreihen, die in unterschiedlichen Koordinatensystemen gemessen wurden, vor der eigentlichen Koordinatensystem-Transformation zu synchronisieren. Durch die Darstellung der Zeitreihen als inkrementelle Rotationen kann ein zeitliches Alignment der beiden Zeitreihen mit Hilfe von Dynamic Time Warping bestimmt werden. Damit lassen sich die beiden Zeitreihen zuerst synchronisieren und anschließend in ein gemeinsames Koordinatensystem transformieren.

Ausblick

In dieser Arbeit wurde gezeigt, wie sich Bewegungs-Zeitreihen allein anhand ihrer Rotations-Anteile per One-Nearest-Neighbor-Verfahren klassifizieren lassen. Eine Weiterentwicklung bestünde nun z.B. darin, eine Klassifikation auf Zeitreihen, bestehend aus kompletten 6-DoF-Posen durchzuführen. Dafür benötigt man allerdings Datensätze, die Posen

eines Sensors mit vollen 6 Freiheitsgraden aufweisen. Bei den verwendeten Datensätzen trifft dies nur auf den 6DMG-, sowie den selbst erstellten JGU_Numbers-Datensatz zu. Datensätze, die ausschließlich mit Inertial-Sensoren aufgezeichnet werden, weisen in der Regel nur die drei Rotations-Freiheitsgrade auf. Eine driftfreie Bestimmung der räumlichen Position des Sensors ist dabei selbst mit Hilfe komplexerer Bewegungsmodelle (z.B. erweiterte Kalman-Filter) nur schwer realisierbar. Aus diesem Grunde benötigt man für die Aufzeichnung voller 6-DoF-Posen andere Aufzeichnungssysteme wie z.B. Kamera-basierte Tracker. Bei der Klassifikation in 6 Freiheitsgraden treten wieder neue Fragestellungen auf. Es muss z.B. eine physikalisch sinnvolle Kopplung von rotatorischem und translatorischem Anteil der Bewegungen erfolgen, um ein punktweises Abstandsmaß für 6-DoF-Posen definieren zu können. Dies könnte beispielsweise mit Hilfe der Screw-Motion-Theorie, bei der die Posen als duale Quaternionen dargestellt werden, erfolgen [40]. Auch eine Darstellung als homogene 4×4-Matrizen ist denkbar, wobei auch dann wieder ein sinnvolles Abstandsmaß zwischen solchen Matrizen verwendet werden muss. Zur Laufzeit-Verbesserung müssten dann auch die Lower Bounds sinnvoll auf 6-DoF-Punkte erweitert werden.

Ein weiterer Punkt ist die Möglichkeit, die für Dynamic Time Warping benötigte Relaxierung der Penalty-Matrix geeignet zu parallelisieren. Für eindimensionale Zeitreihen hat Hundt in [20] bereits eine effiziente CUDA-Parallelisierung von DTW implementiert. Diese könnte entsprechend auf mehrdimensionale Zeitreihen erweitert werden. Da durch geeignete Parallelisierung die Laufzeiten teilweise um ein bis zwei Größenordnungen verringert werden können, wäre es damit möglich, ein Anfrage-Shape mit einer wesentlich größeren Menge an bekannten Bewegungsmustern zu vergleichen, wodurch die Klassifikations-Genauigkeit enorm steigen würde.

Danksagungen

Allen voran möchte ich mich bei meinem Doktorvater bedanken. Er hat mich stets tatkräftig unterstützt und war jederzeit für mich da, wenn ich Hilfe oder Rat brauchte. Ebenfalls möchte ich dem Zweitgutachter dieser Dissertation für seinen Einsatz danken. Zudem möchte ich auch den weiteren Mitarbeitern vom Institut für Informatik der Johannes Gutenberg-Universität Mainz für die gute und stets fruchtbare Zusammenarbeit, die eine wesentliche Grundlage für die Entstehung dieser Arbeit war, danken. Den Mitarbeiterinnen vom Instituts-Sekretariat möchte ich danken, dass sie sich dafür eingesetzt haben, dass vertragliche Schwierigkeiten in der letzten Promotions-Phase erfolgreich überwunden werden konnten. Mein weiterer Dank gilt den Mitarbeitern vom Zentrum für Datenverarbeitung für die zur Verfügung gestellte Rechenzeit auf dem Supercomputer MOGON der Universität Mainz, auf dem ein großer Teil der in dieser Arbeit verwendeten Rechnungen durchgeführt wurde. Ebenfalls danken möchte den Mitarbeitern der Technik-Gruppe für ihre Unterstützung bei technischen Problemen mit der Hardware. Außerdem möchte ich unseren Kooperationspartnern aus Wiesbaden und Rüsselsheim danken, für die gelungene Zusammenarbeit im Rahmen mehrerer Kooperationsprojekte, auf deren Forschung diese Arbeit im Wesentlichen aufbaut. Zum Schluss möchte ich meiner Familie und meinen Freunden danken, die mich immer wieder ermutigt haben, meine Ziele im Blick zu halten und immer für mich da waren.

ANHANG

FORMALE DEFINITIONEN

Verwendete Mengensymbole

- Sei $\mathbb{R}$ die Menge der reellen Zahlen.
- Sei $\mathbb{R}^+$ die Menge der positiven reellen Zahlen – exklusive der Null – :
  $\mathbb{R}^+ := \{x \in \mathbb{R} \mid x > 0\}$.
- Sei $\mathbb{R}_0^+$ die Menge der positiven reellen Zahlen – inklusive der Null – :
  $\mathbb{R}_0^+ := \{x \in \mathbb{R} \mid x \geq 0\}$.
- Sei $\mathbb{R}^N$ der N-dimensionale Vektorraum über der Menge der reellen Zahlen.
- Sei $\mathbb{R}^{N \times M}$ der $N \times M$-dimensionale Vektorraum der Matrizen über der Menge der reellen Zahlen.
- Sei $\mathbb{C}$ die Menge der komplexen Zahlen.
- Sei $\mathbb{N}$ die Menge der natürlichen Zahlen – exklusive der Null.
- Sei $\mathbb{N}_0$ die Menge der natürlichen Zahlen – inklusive der Null.
- Sei $\mathbb{H}$ die Menge der Quaternionen. Mit dem Quaternionen-Produkt
  $(*) : \mathbb{H} \times \mathbb{H} \mapsto \mathbb{H}$ bilden die Quaternionen eine multiplikative Gruppe.
- Sei $\mathbb{H}_1$ die Menge der normierten (Rotations-)Quaternionen:
  $\mathbb{H}_1 := \{q \in \mathbb{H} \mid \|q\| = 1\}$

Weitere Definitionen

- Sei $SO(3)$ die Menge der Rotations-Matrizen im 3-dim. Raum (*Drehgruppe*):
  $SO(3) := \{\Theta \in \mathbb{R}^{3\times 3} \mid \Theta \cdot \Theta^{\top} = \Theta^{\top} \cdot \Theta = \mathbb{1}, \det(\Theta) = +1\}$.
- Sei $SE(3)$ die Menge der starren Transformationen im dreidimensionalen Raum:
  $SE(3) := \{(\Theta, \vec{t}) \mid \Theta \in SO(3), \vec{t} \in \mathbb{R}^3\}$.
- Sei $[A, B]$ mit $A, B \in \mathbb{R}$ ein reellwertiges, geschlossenes Intervall:
  $[A, B] := \{x \in \mathbb{R} \mid A \leq x \leq B\} \subseteq \mathbb{R}$.
- Sei (A, B) mit $A, B \in \mathbb{R}$ ein reellwertiges, offenes Intervall:
  $(A, B) := \{x \in \mathbb{R} \mid A < x < B\} \subseteq \mathbb{R}$.

ANHANG

B

VERZEICHNISSE

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LITERATURVERZEICHNIS

[1] H. Tjaden, U. Schwanecke, F. A. Stein, und E. Schömer, „High-Speed and Robust Monocular Tracking," in *Proceedings of the 10th International Conference on Computer Vision Theory and Applications (VISIGRAPP)*, 2015, S. 462–471.

[2] soft2tec Rüsselsheim, „nexonar - next generation tracking technology," http://www.nexonar.com/de/produkte/listener/3dlistener/#data, 2017, [Online; abgerufen: 13. Juni 2017].

[3] M. W. Kadous, „Learning Comprehensible Descriptions of Multivariate Time Series," in *Proceedings of the Sixteenth International Conference on Machine Learning*, Serie ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, S. 454–463.

[4] K.-H. Goldhorn, H.-P. Heinz, und M. Kraus, *Moderne mathematische Methoden der Physik. Band 2: Operator- und Spektraltheorie - Gruppen und Darstellungen*. Springer Verlag, Berlin, 2010, S. 147–171.

[5] W. Hamilton, *Lectures on Quaternions*. Hodges and Smith, Dublin, 1853.

[6] K. Shoemake, „Animating Rotation with Quaternion Curves," *SIGGRAPH Comput. Graph.*, Vol. 19, Nr. 3, S. 245–254, Jul. 1985.

[7] E. B. Dam, M. Koch, und M. Lillholm, „Quaternions, interpolation and animation," Department of Computer Science, University of Copenhagen, Tech. Rep., 1998.

[8] H. Goldstein, C. Poole, und J. Safko, *Classical Mechanics, Third Edition*. Addison Wesley, New York u.a., 2002.

[9] R. M. Murray, S. S. Sastry, und L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*, 1. Aufl. Boca Raton, FL, USA: CRC Press, Inc., 1994.

[10] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, G. Tröster, P. Lukowicz, G. Pirkl, D. Bannach, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, und J. d. R. Millán, „Collecting complex activity data sets in highly rich networked sensor environments," in *Proceedings of the Seventh International Conference*

on Networked Sensing Systems (INSS), Kassel, Germany. IEEE Computer Society Press, Jun. 2010.

[11] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, und D. Roggen, „The Opportunity Challenge: A Benchmark Database for On-body Sensor-based Activity Recognition," *Pattern Recognition Letters*, Vol. 34, Nr. 15, S. 2033–2042, Nov. 2013. [Online]. Abrufbar unter: http://dx.doi.org/10.1016/j.patrec.2012.12.014

[12] S. Alavi, D. Arsenault, und A. Whitehead, „Quaternion-Based Gesture Recognition Using Wireless Wearable Motion Capture Sensors," *Sensors*, Vol. 16, Nr. 5, S. 605, 2016. [Online]. Abrufbar unter: http://www.mdpi.com/1424-8220/16/5/605

[13] M. Chen, G. AlRegib, und B.-H. Juang, „6DMG: A New 6D Motion Gesture Database," in *Proceedings of the 3rd Multimedia Systems Conference*, Serie MMSys '12. New York, NY, USA: ACM, 2012, S. 83–88. [Online]. Abrufbar unter: http://doi.acm.org/10.1145/2155555.2155569

[14] K. Henriksen, J. Sporring, und K. Hornbæk, „Virtual Trackballs Revisited," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 10, Nr. 2, S. 206–216, Mrz. 2004.

[15] J. Macqueen, „Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, Nr. 14. Oakland, CA, USA., 1967, S. 281–297.

[16] [verschiedene Autoren], „Online Documentation – OpenCV: Smoothing Images," http://docs.opencv.org/3.2.0/d4/d13/tutorial_py_filtering.html, 2016, [Online; abgerufen: 03. März 2017].

[17] H. Ryan, „Ricker, Ormsby, Klauder, Butterworth – A Choice of Wavelets," *CSEG Recorder*, S. 8–9, 1994.

[18] H. Sakoe und S. Chiba, „Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 26, S. 43–49, 1978.

[19] D. J. Berndt und J. Clifford, „Using Dynamic Time Warping to Find Patterns in Time Series," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, Serie AAAIWS'94. AAAI Press, 1994, S. 359–370. [Online]. Abrufbar unter: http://dl.acm.org/citation.cfm?id=3000850.3000887

[20] C. Hundt, „Efficient Subsequence Alignment of Time Series - A massively parallel approach," Doktorarbeit, Johannes Gutenberg-Universität Mainz, 2015.

[21] D. S. Hirschberg, „A Linear Space Algorithm for Computing Maximal Common Subsequences," *Commun. ACM*, Vol. 18, Nr. 6, S. 341–343, Jun. 1975. [Online]. Abrufbar unter: http://doi.acm.org/10.1145/360825.360861

[22] C. A. Ratanamahatana und E. Keogh, „Making Time-series Classification More Accurate Using Learned Constraints," in *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 2004, S. 11–22.

[23] E. Keogh, „Exact Indexing of Dynamic Time Warping," in *Proceedings of the 28th International Conference on Very Large Data Bases*, Serie VLDB '02. VLDB Endowment, 2002, S. 406–417.

[24] S.-W. Kim, S. Park, und W. W. Chu, „An index-based approach for similarity search supporting time warping in large sequence databases," in *Proceedings of the 17th International Conference on Data Engineering*, 2001, S. 607–614.

[25] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, und E. Keogh, „Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, Serie KDD '12. New York, NY, USA: ACM, 2012, S. 262–270.

[26] D. Lemire, „Streaming Maximum-minimum Filter Using No More Than Three Comparisons Per Element," *Nordic Journal of Computing*, Vol. 13, Nr. 4, S. 328–339, Dez. 2006. [Online]. Abrufbar unter: http://dl.acm.org/citation.cfm?id=1324123.1324129

[27] Johannes Gutenberg-Universität Mainz, „Systeme - High Performance Computing," https://hpc.uni-mainz.de/high-performance-computing/mogonbild/, 2016, [Online; abgerufen: 16. März 2017].

[28] I. Steinwart und A. Christmann, *Support Vector Machines*. Springer New York, 2008, S. 285–329.

[29] R. Kruse, C. Borgelt, C. Braune, F. Klawonn, C. Moewes, und M. Steinbrecher, *Computational Intelligence*, 2. Aufl. Springer Vieweg, 2015.

[30] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, und D. Steinberg, „Top 10 algorithms in data mining," *Knowledge and Information Systems*, Vol. 14, Nr. 1, S. 1–37, 2008. [Online]. Abrufbar unter: http://dx.doi.org/10.1007/s10115-007-0114-2

[31] L. Breiman, „Random Forests," *Machine Learning*, Vol. 45, Nr. 1, S. 5–32, 2001. [Online]. Abrufbar unter: http://dx.doi.org/10.1023/A:1010933404324

[32] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, und E. Keogh, „Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, Vol. 26, Nr. 2, S. 275–309, 2013. [Online]. Abrufbar unter: http://dx.doi.org/10.1007/s10618-012-0250-5

[33] P.-N. Tan, M. Steinbach, und V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[34] M. Shah, R. D. Eastman, und T. Hong, „An overview of robot-sensor calibration methods for evaluation of perception systems," in *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*. ACM New York, NY, USA, 2012, S. 15–20.

[35] K. Daniilidis, „Hand-Eye Calibration Using Dual Quaternions," *The International Journal of Robotics Research*, S. 286–298, 1999.

[36] Y. Shiu und S. Ahmad, „Calibration of Wrist-Mounted Robotic Sensors by Solving Homogeneous Transform Equations of the Form AX = XB," *IEEE Transactions on Robotics and Automation*, S. 16–29, 1989.

[37] R. Tsai und R. Lenz, „A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration," *IEEE Transactions on Robotics and Automation*, S. 345–358, 1989.

[38] F. C. Park und B. J. Martin, „Robot sensor calibration: solving AX=XB on the Euclidean group," *IEEE Transactions on Robotics and Automation*, S. 717–721, 1994.

[39] H. H. Chen, „A screw motion approach to uniqueness analysis of head-eye geometry," in *IEEE Proceedings of Computer Vision and Pattern Recognition*, 1991, S. 145–151.

[40] K. Daniilidis und E. Bayro-Corrochano, „The dual quaternion approach to hand-eye calibration," in *Proceedings of the 13th International Conference on Pattern Recognition*, 1996, S. 318–322.

[41] H. Zhuang und Y. C. Shiu, „A Noise-Tolerant Algorithm for Robotic Hand-Eye Calibration with or without Sensor Orientation Measurement," *IEEE Transactions on Systems, Man, and Cybernetics*, S. 1168–1175, 1993.

[42] A. Nádas, „Least squares and maximum likelihood estimation of rigid motion," *IBM Research Report RC6945*, 1978.

[43] D. Will, „Fast Approximative Data Structures for Applications in the Automotive Industry," Doktorarbeit, Johannes Gutenberg-Universität Mainz, 2016.

[44] H. Zhuang, Z. S. Roth, und R. Sudhakar, „Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form AX=YB," *IEEE Transactions on Robotics and Automation*, Vol. 10, Nr. 4, S. 549–554, Aug. 1994.

[45] M. A. Fischler und R. C. Bolles, „Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, Vol. 24, Nr. 6, S. 381–395, Jun. 1981. [Online]. Abrufbar unter: http://doi.acm.org/10.1145/358669.358692

[46] M. Hanke-Bourgeois, *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, 3. Aufl. Springer, 2009.

[47] D. Franzen, „Optisches Tracking mit hoher räumlicher und zeitlicher Auflösung," Bachelorarbeit, Johannes Gutenberg-Universität Mainz, 2015.

ANHANG C

LEBENSLAUF

Lebenslauf

Zur Person

■ Name	Frédéric Alexander Stein
■ Geburtsdatum	[ausgelassen in elektronischer Version der Dissertation]
■ Geburtsort	[ausgelassen in elektronischer Version der Dissertation]
■ Staatsangehörigkeit	deutsch
■ Anschrift	[ausgelassen in elektronischer Version der Dissertation]

Schule und Studium

■ 08/1993 – 03/2002	Abitur, staatliches Gymnasium Nieder-Olm
■ 04/2003 – 12/2010	Diplom Physik, Johannes Gutenberg-Universität Mainz
■ seit 2012	Doktorand am Institut für Informatik, Johannes Gutenberg-Universität Mainz

Berufserfahrung

■ 04/2002 – 02/2003	Zivildienst, Seniorenresidenz Nieder-Olm
■ seit 2011	wissenschaftlicher Mitarbeiter an der Johannes Gutenberg-Universität Mainz

Sonstiges

■ Fremdsprachen	Englisch (B2), Französisch (B1)

Mainz, 28. Juni 2017