Albert-Ludwigs-Universität Freiburg

Master Thesis

Multiple Interval-based Curve Alignment (MICA)

Thesis in Erfüllung der Anforderungen
zum Erlangen des Grades Master of Science
im

Lehrstuhl für Bioinformatik
Institut für Informatik

Erstellt von:
B. Sc. Matthias Beck

Betreuer:
Dr. Martin Mann

Gutachter:
Prof. Dr. Rolf Backofen
Prof. Dr. Heinrich Spiecker

Februar 2014

Zusammenfassung

Zu Beginn dieser Arbeit wird ein Problem dargestellt, welches verdeutlicht, warum eine Alignierungsmethode erforderlich ist. Dabei wird auf Arbeiten eingegangen, welche sich mit ähnlichen Problemen befassen.

Im Anschluss daran werden die Grundlagen erarbeitet, welche für die Umsetzung der Alignierung relevant sind. Auf diesen Grundlagen baut die Konzeption einer Alignierungsmethode für multiple Kurven auf, welche im Einzelnen in dieser Arbeit beschrieben wird. Dazu zählt auch die Entwicklung von verschiedenen Strategien dieser Alignierungsmethode.

Teil dieser Arbeit ist sowohl die Konzeption einer multiplen Kurven-Alignierungsmethode als auch deren Implementation. Dazu zählt auch die Beschreibung der Implementation und deren grafischer Benutzeroberfläche. Die grafische Benutzeroberfläche soll dabei den Benutzer bei der Durchführung der Alignierung unterstüzen.

Abschließend wird eine Evaluation der in dieser Arbeit entstandenen Anwendung mit einer vergleichbaren existierenden Anwendung zur multiplen Kurven-Alignierung durchgeführt. Dabei wird bei der Evaluation ein besonderer Augenmerk auf die Qualität der resultierenden Alignments gelegt. Zudem wird in der Evaluation analyisert, in welchem Ausmaß eine umgesetzte Parallelisierung in der Implementation einen Geschwindigkeitsvorteil bei der Durchführung der Alignierung bietet.

Zum Schluss wird das Ergebnis dieser Arbeit zusammengefasst dargestellt.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während der Erstellung dieser Masterarbeit unterstützt haben.

Ganz besonderen Dank gilt meinem Betreuer Martin Mann für seine umfangreiche Unterstützung. Nicht nur für seine wertvollen Hinweise möchte ich mich recht herzlich bedanken, sondern auch für seine Bereitschaft für kurzfristige Besprechungen in seinem Büro.

Auch will ich mich bei Matthias Eisenmann für das Korrekturlesen bedanken.

Zudem gilt meiner Freundin Nicole Leonhard ein Dank für ihre Motivation und emotionale Unterstützung.

Nicht zuletzt möchte ich mich bei meinen Eltern Maria Beck und Alfred Beck bedanken, welche mich das gesamte Studium über sowohl finanziell als auch bei all meinen Entscheidungen unterstützt haben.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Dieses Kapitel beginnt mit einem Beispiel, welches verdeutlicht, warum eine intervallbasierte Alignierungsmethode erforderlich sein kann. Zudem wird auf Arbeiten eingegangen, welche in Zusammenhang mit dieser Arbeit stehen. Zum Schluss dieses Kapitels wird das Ziel und der Aufbau der Arbeit definiert.

1.1 Motivation

Hat man eine Menge von Daten zur Verfügung, lässt sich für die Menge als Ganzes eine Aussage treffen. Diese Aussage repräsentiert die Eigenschaften aller in der Menge enthaltenen Daten. Im nun folgenden Beispiel ist ein Datum der Menge eine Kurve. Mehrere Kurven zusammengefasst, ergeben die Menge der Kurven. Um nun für die Menge an Kurven eine Aussage zu treffen, kann man über Mittelwertsbildung eine Konsensuskurve bilden. Mit Hilfe der Aussage der Konsensuskurve sind zum Beispiel Abweichungen in der Menge der Kurven auszugleichen. Dabei sind Messfehler eine mögliche Ursache für Abweichungen der einzelnen Kurven.

Im Beispiel betrachten wir eine Menge aus genau zwei Kurven P und P'. Beide Kurven sind in Abbildung 1.1 grafisch dargestellt. Die Wertetabellen zu beiden Kurven befindet sich im Anhang A.1. Um nun für die Menge eine Aussage zu treffen wird im Folgenden der Mittelwert aus beiden Kurven P und P' berechnet. Dabei wird für jeden Punkt der Konsensuskurve $\hat{P}$ die Werte beider korrespondierenden Punkte aus P und P' addiert und durch zwei dividiert. Die Konsensuskurve $\hat{P}$ ist

in Abbildung 1.2 zu sehen. Bei der Berechnung von $\hat{P}$ handelt es sich um keine Alignierung, sondern um eine direkte Mittelwertberechnung.

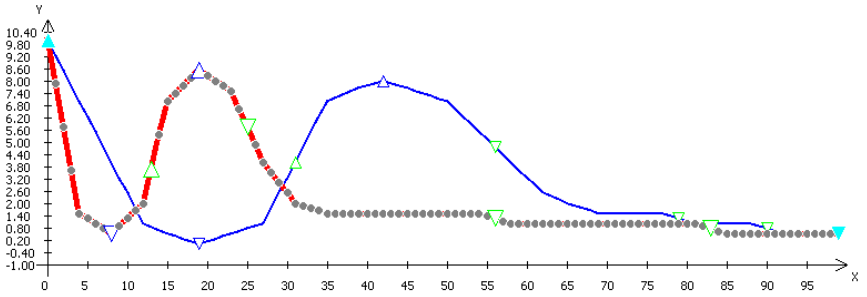

Abbildung 1.1: Eingabe: Kurve P (blau), Kurve P' (rot)

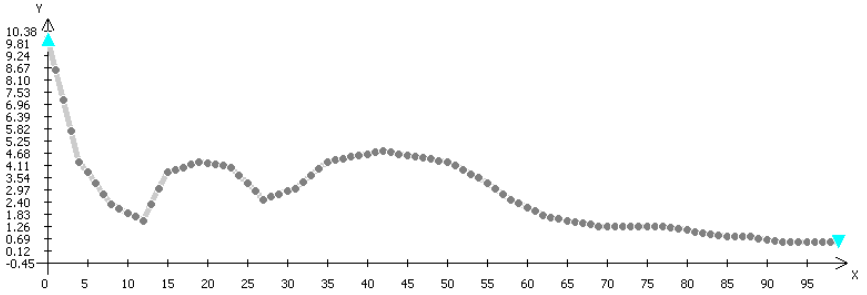

Abbildung 1.2: Ergebnis: Kurve $\hat{P}$ (grau) aus naiver Mittelwertberechnung von P und P'

Bei einem Vergleich der Gestalt von P und P' ist zu sehen, dass auf einen Tiefpunkt zum Beginn ein Hochpunkt in beiden Profilen folgt. Nach dem Hochpunkt verläuft die Kurve beider Profile langsam gegen null. Vergleicht man nun die Gestalt des Profils $\hat{P}$, welchen den naiven Mittelwert beider Profile darstellt, mit den Gestalten von P, P', so sind deutlich Unterschiede festzustellen. Es existieren nun zwei Tiefpunkte und zwei Hochpunkte. Somit hat die Gestalt des Profils, welches mit dem naiven Mittelwert berechnet wurde, nichts mit den Gestalten der zu Grunde liegenden Profile zu tun.

In der folgenden Abbildung 1.4 ist das Ergebnis einer intervallbasierten Alignierungsmethode zu sehen. Dabei repräsentiert die Kurve $\tilde{P}$ die Gestalt der beiden Ausgangskurven P und P'. In Abbildung 1.3 ist eine Verzerrung der beiden Ausgangskurven durch die intervallbasierte Alignierungsmethode zu sehen. Diese Verzerrung ist notwendig, um das Ergebnis $\tilde{P}$, mit seiner gestalterhaltenden Form, zu ermitteln.

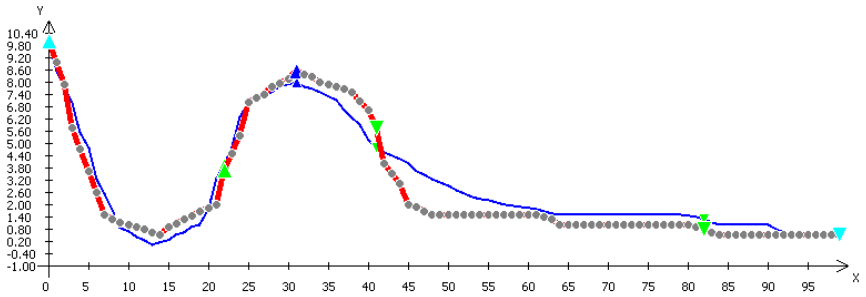

Abbildung 1.3: Ergebnis: Verzerrte Kurve P und P'

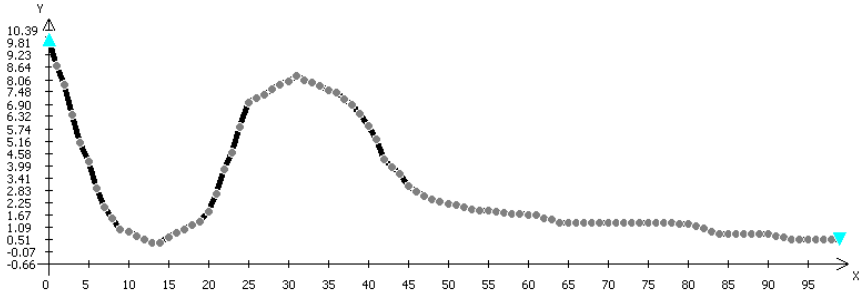

Abbildung 1.4: Ergebnis: Kurve $\tilde{P}$ (schwarz) durch intervallbasierte Alignierung von P und P'

Dieses Beispiel zeigt deutlich, dass der naive Ansatz zur Berechnung des Mittelwerts einer Menge von Daten nicht immer aussagekräftig ist. An dieser Stelle bietet sich eine intervallbasierte Alignierungsmethode an. Solch eine Methode ermöglicht eine bessere Repräsentation der Gemeinsamkeiten von Kurvenmengen. Diese Arbeit befasst sich im Folgenden mit der Umsetzung einer mehrfachen intervallbasierten Kurven-Alignierungsmethode (MICA, Multiple Interval-based Curve Alignment).

Verwendung findet die Kurven-Alignierungsmethode zum Beispiel in der Spracherkennnung. Dort werden Signalkurven mit einer Musterkurve verglichen, um einzelne Wörter zu bestimmen. Das Problem an dieser Stelle ist, dass dasselbe Wort unterschiedlich schnell gesprochen oder betont werden kann. Dies führt zu einer Abweichung in der Phase und Amplitude der Signalkurve. An dieser Stelle setzt die Alignierung an und erstellt ein Mapping von Kurvenbereichen auf das Muster. Somit lässt sich bestimmen, wie ähnlich eine Kurve einem Muster ist. Dabei wird ein DTW (dynamic time warping) Verfahren verwendet, wie es in [Wang and Gasser, 1997], [Petitjean and Gançarski, 2012] und [Petitjean et al., 2011] beschrieben ist.

Ein anderes Beispiel für die multiple Alignierung von Kurven stellt die Erstellung einer repräsentativen Kurve (Konsensus) aus einer Menge von Ausgangskurven dar. Der Konsensus besitzt dabei die charakteristischen Eigenschaften aller Ausgangskurven. Somit kann man durch den Konsensus Aussagen für die ganze Menge der Ausgangskurven treffen. Zudem ist es möglich eine Separierung von Kurven mit gleichen Eigenschaften durchzuführen. Dabei handelt es sich um Clustering-Verfahren, wie sie in [Liu and Yang, 2009] und [Sangalli et al., 2008] beschrieben sind. Dadurch lassen sich aus einer Menge von Kurven jene finden, welche sich am ähnlichsten sind. Diese werden dann in einem Cluster gruppiert.

1.2 In Zusammenhang stehende Arbeiten

Für die Alignierung ist es wichtig zu wissen, welche Kurvenbereiche am besten zueinander passen. Dazu ist es zunächst erforderlich, die Charakteristik einer Kurve zu analysieren. Zu den Charakteristika zählen Hoch-, Tief- und Wendepunkte. Die Methode zur Identifizierung solcher Charakteristika von Kurven ist in [Gasser and Kneip, 1995] beschrieben und baut auf zwei Ansätzen auf. Zunächst wird die Identifizierung von charakteristischen Punkten der einzelnen Kurven durch Regression und Differenzierung durchgeführt. Anschließend wird die relative Häufigkeit der Charakteristika entlang der horizontalen Achse bestimmt. Diese Häufigkeit entlang der horizontalen Achse spiegelt die Struktur einer Kurve wieder.

Das von [Srivastava et al., 2011] und [Kurtek et al., 2011] beschriebene Verfahren zur Registrierung von Funktionsdaten betrachtet die Phasen- und Amplitudeninformation von Funktionen getrennt. Dabei wird eine Distanzmetrik auf dem Quotienten von einer Ausgangsfunktion zu deren Verzerrung gebildet. Mit Hilfe dieser Distanzmetriken lässt sich ein Durchschnitt bilden, nach welchem die Ausgangsfunktionen durch Verzerren ausgerichtet werden. Das Verzerren wird dabei als Werkzeug betrachtet, welches eine Funktion horizontal aligniert. Dies verringert somit die Abweichung vom Durchschnitt. Das Verfahren folgt dabei folgenden Schritten: Berechnen der Distanzmetrik aller Funktionen; Erzeugen des Durchschnitts aus den Distanzmetriken; Bestimmen der minimalen Distanz einer Funktion mit Verzerrung zum Durchschnitt, anschließender Alignierung und resultierender Verzerrungsfunktion.

[Ramsay and Li, 1998] beschäftigt sich mit einem, zu Kapitel 1.1, analogen Problem. Dabei repräsentiert eine Durchschnittskurve keine der Ausgangskurven. Die Kurvendaten (Wachstums-, Temperaturkurven) sind dabei entlang einer Zeitachse erfasst. Ein Vergleich auf der Grundlage der zeitlichen Information liefert somit unzureichende Ergebnisse. In [Ramsay and Li, 1998] wird ein Registrierungsansatz für Kurven beschrieben, welcher das Problem löst. Ein ähnlicher Registrierungsansatz findet in der Umsetzung dieser Arbeit Verwendung. Der Registrierungsansatz basiert auf der Identifizierung von hervorstechenden Merkmalen der Kurven, welche zeitlich so ausgerichtet werden, dass die Merkmale zur gleichen Zeit auftreten. Solche Merkmale sind zum Beispiel Extrempunkte. Durch ein stochastisches Transformationsmodell und der Schätzung einer Verzerrungsfunktion wird die Kurvenregistrierung durchgeführt.

Liefert eine Alignierung von verschiedensten Kurven nur schlechte Ergebnisse, hilft eine Gruppierung der Kurven in einzelne Cluster. Ein Clustering-Verfahren für die Alignierung von Kurven wird in [Sangalli et al., 2008] durch einen k-Means-Algorithmus durchgeführt. Dazu wird ein Ähnlichkeitsmaß zwischen zwei Clustern mit Kurven definiert. Dieses Ähnlichkeitsmaß wird verwendet, um k Cluster von ähnlichen Kurven zu bilden. Für verschiedene k werden die Kurven in den Clustern aligniert. Durch das Bilden von Alignments auf der Basis von Clustern erhält man Kurven, welche aussagekräftiger in der Repräsentierung der Eigenschaften der Ausgangskurven, und vor allem der Kurven des Clusters, sind.

Ein weiterer Clustering Ansatz für Funktionsdaten ist in [Liu and Yang, 2009] beschrieben. Dabei wird das Clustering auf der Grundlage von zeitlicher Verzerrung der Funktionsdaten durchgeführt. Zeitliche Verzerrung ist dabei auf die horizontale Achse des Koordinatensystems bezogen. Zeitgleich mit dem Clustering findet die Alignierung in diesem Verfahren statt.

[Kneip and Ramsay, 2008] versucht Kurven durch das Finden einer Menge von Verzerrungsfunktionen gemäß ihrer Eigenschaften zu alignieren/registrieren. Es wird gezeigt, dass eine Registrierung von Haupteigenschaften von Funktionen, wie zum Beispiel die Amplitude, für den Registrierungsvorgang nützlich sind. Zudem wird ein Algorithmus vorgestellt, welcher iterativ eine Zielfunktion, auf der Grundlage von Verzerrungsfunktionen und der Bewertung von Funktionshaupteigenschaften, liefert. Die Zielfunktion stellt dabei den Konsensus der Ausgangskurven dar.

In [Kneip and Gasser, 1992] wird, ausgehend von Beispielkurven, eine Durchschnittskurve gesucht, welche die typische Struktur jeder der Beispielkurven wiedergibt. Eine direkte Erstellung einer Durchschnittskurve liefert, analog zum Beispiel in Kapitel 1.1, schlechte Ergebnisse. Aufgrund von Verschiebungen der Beispielkurven können Ausprägungen in der Durchschnittskurve ausgelöscht werden. Der in der Arbeit beschriebene Ansatz synchronisiert/verschiebt die einzelnen Kurven bevor die Durchschnittskurve gebildet wird. Das daraus erstellte Ergebnis spiegelt die Gemeinsamkeiten der Beispielkurven sowohl in Struktur als auch in der Stärke wider.

[Wang and Gasser, 1997] befasst sich mit dem Alignieren von Kurven auf der Grundlage von dynamischer Zeitverzerrung, wobei auf einigen Ideen aus [Kneip and Gasser, 1992] aufgebaut wird. Jedoch wird das in [Kneip and Gasser, 1992] beschriebene Verfahren mit den einzelnen Verschiebungsfunktionen für die Alignierung der Kurven als zu empfindlich und zeitaufwändig erachtet. Deshalb wird in [Wang and Gasser, 1997] ein dynamischer Zeitverzerrungsansatz (dynamic time warping, DTW) für das Alignieren von zwei Signalen verwendet, wie er in [Rabiner and Schmidt, 1980] beschrieben ist. DTW wird bei der Spracherkennung eingesetzt. Da ein Wort unterschiedlich lang ausgesprochen werden kann und für die Erkennung ein fixes Muster existiert, führt der DTW eine Zeitnormierung durch. Bei DTW handelt es sich um ein Verfahren, welches durch die Dynamische Programmierung, mit einer Komplexität von $O(n^2)$ Zeit und Speicher, gelöst wird. Eine Optimierung stellt FastDTW[1] dar, welche annähernd mit einer Komplexität von $O(n)$ für Zeit und Speicher auskommt. Weitere Optimierungen der DTW-Ansätze sind in [Al-Naymat et al., 2012] und [Lemire, 2009] beschrieben. Ein Problem beim DTW-Ansatz ist, dass unter Umständen durch die Rekursion mehrere Punkte einer Kurve auf einen Punkt der anderen Kurve gemappt werden. Dies resultiert in einem Verlust von Intervallen. Im Vergleich dazu kann auch bei der in dieser Arbeit umgesetzten MICA-Methode ein Verlust von Datenpunkten auftreten, falls ein langes Intervall auf ein sehr viel kürzeres Intervall verzerrt wird. DWT löst dabei das gleiche Problem, welches in dieser Arbeit durch das Finden der optimalen Intervallpaarzerlegung umgesetzt wurde.

[Petitjean and Gançarski, 2012] verwendet DTW, um ein multiples Alignment

[1] `https://code.google.com/p/fastdtw/`, Dynamic Time Warping (DTW) with a linear time and memory complexity

von gegebenen Sequenzen zu erstellen. Dabei wird DWT für die Ähnlichkeitsbestimmung zwischen Sequenzen verwendet. Implizit ist durch die Bestimmung der Ähnlichkeit das korrespondierende Mapping zwischen beiden Sequenzen gegeben. Ein weiterer Anwendungsfall für DTW ist in [Petitjean et al., 2011] beschrieben. Dabei wird die Ähnlichkeitsbestimmung verwendet, um Cluster zu bilden.

In [Rice and Silverman, 1991] wird eine Methode beschrieben, welche auf der Grundlage einer Menge von Ausgangskurven eine Mittelwertkurve abschätzt. Dabei wird versucht jene Mittelwertkurve zu finden, welche die Distanz zu allen Ausgangskurven minimiert. Für die Distanzbewertung wird dabei die Methode der kleinsten Quadrate angewandt (generalized least squares, vgl. [Takeaki and Hiroshi, 2004]).

[Tang and Müller, 2008] zeigt einen Ansatz, welcher paarweise Verzerrungsfunktionen auf der Grundlage aller Kombinationen der Ausgangskurven bestimmt. Verzerren synchronisiert dabei einzelne Kurvenverläufe auf einen globalen Kurvenverlauf. Anschließend werden die paarweisen Verzerrungsfunktionen genutzt, um eine Abschätzung für den nächsten Schritt zu treffen. Dabei wird ein Durchschnitt aus den individuellen paarweisen Verzerrungen zu einer Kurve gebildet. Diese durchschnittliche Verzerrung liefert die globale Verzerrung für eine Kurve. Mit Hilfe der globalen Verzerrungen lassen sich die Kurven synchronisieren.

Diese hier umgesetzte Arbeit basiert grundlegend auf dem bereits vorgestellten Ansatz aus [Bender et al., 2012]. Wie auch bei den zuvor beschriebenen Arbeiten wird hier auf einer Alignierung durch die Ausrichtung von prägnanten Kurveneigenschaften aufgebaut. Dazu zählen Extrempunkte und Wendepunkte. Das dort beschriebene und hier umgesetzte Verfahren basiert auf einer progressiven iterativen Vorgehensweise, bei der schrittweise Ergebnisse aus paarweisen Alignierungsvorgängen berechnet werden. Dabei basiert das Verfahren aus [Bender et al., 2012] auf einer statischen Vorgehensweise, bei der bereits zu Beginn, durch paarweise Distanzberechnung, festgelegt wird, in welcher Reihenfolge die progressiven Alignierungsverfahrensschritte durchgeführt werden. Der in dieser Arbeit gewählte Ansatz sieht eine dynamische Anpassung der Distanzen vor. Dadurch kann sich die Reihenfolge der progressiven Verfahrensschritte zur Laufzeit ändern, wovon man sich ein besseres Ergebnis verspricht.

1.3 Ziel und Aufbau der Arbeit

Der Fokus dieser Arbeit richtet sich auf die Methoden zur Umsetzung der multiplen Alignierung von Kurven. Zu Beginn wird ein kleines Beispiel eingeführt, welches sowohl die Problematik als auch den Nutzen der Alignierung beschreibt. Bevor die Methoden für die Alignierung erläutert werden, wird auf die dazu nötigen Grundlagen eingegangen. Anschließend wird die Umsetzung im Einzelnen beschrieben und zum Schluss der Arbeit eine Zusammenfassung gegeben.

Ziel dieser Arbeit ist die Entwicklung einer interaktiven grafischen Benutzeroberfläche, die es ermöglicht multiple Kurvenalignments durchzuführen. Dabei wird ein dynamischer progressiver Ansatz für die multiple Kurvenalignierung umgesetzt. Zudem zählt die Evaluierung des neuen dynamischen Ansatzes im Vergleich mit dem statischen Ansatz aus [Bender et al., 2012] zum Ziel der Arbeit.

Kapitel 2

Grundlagen

Dieses Kapitel befasst sich mit theoretischen Grundlagen und Methoden. Beides ist wichtig für die Umsetzung der Alignierungsmethoden. Die Alignierungsmethoden sind nicht Teil dieses Kapitels und werden gesondert im nachfolgenden Kapitel 3 behandelt. Zunächst wird das Profil als zentrale Struktur genauer beschrieben. Anschließend wird auf Filtermechanismen für wichtige Eigenschaften eines Profils eingegangen. Am Ende dieses Kapitels wird die Intervallzerlegung eingeführt.

2.1 Profil

Ein Profil P ist gegeben durch eine Sequenz von Punkten (x_i, y_i) mit $i \in [1, n]$ und $x_i < x_{i+1}$. Dabei gilt folgende, in Gleichung 2.1 definierte, Annahme:

$$x_i = i \tag{2.1}$$

Profil P steht für eine Datenpunktmenge. Die Datenpunktmenge $\mathbb{X}$ über fortlaufende n Datenpunkte ist mit $\mathbb{X} = \{x_i | x_i < x_{i+1}\}$ mit $i \in [1, n]$ definiert. P wird um eine approximative Funktion $y : [x_1, x_n] \to \mathbb{R}$ erweitert. $\mathbb{R}$ steht dabei in diesem Fall für die Menge der reellen Zahlen. Die Definition für Funktion y ist in Gleichung 2.2 beschrieben.

$$y(x) = \begin{cases} y_i & \text{, falls } x = x_i \in \mathbb{X} \\ y_i + \frac{y_j - y_i}{x_j - y_i}(x - x_i) & \text{, sonst lineare Interpolation} \\ & \text{mit } x_i = \lfloor x \rfloor \in \mathbb{X}, x_j = \lceil x \rceil \in \mathbb{X} \end{cases} \tag{2.2}$$

Im Verlauf der Arbeit werden Steigungswerte der Profile für Ähnlichkeitsbestimmungen benötigt. Auf Grund der diskreten Daten eines Profils ist es erforderlich die Steigungswerte zu approximieren. Dabei werden zuerst Steigungswerte s_i für jeden Datenpunkt (x_i, y_i) mit Hilfe einer linearen Regression über ein gleitendes Fenster geschätzt.

Die Approximation geschieht mit Hilfe einer linearen Regression über ein gleitendes Fenster der Länge $2c + 1$. Hierbei ist c der Radius des gleitenden Fensters und es gilt $c \ll n$. Für jeden Punkt x_i werden je c Punkte links und rechts innerhalb des Fensters für die lineare Regression betrachtet. Im Folgenden wird $c = 2$ verwendet, soweit nicht anders angegeben.

Für das gleitende Fenster können zwei Grenzfälle auftreten. Beide führen zu Zugriffen welche nicht in $\mathbb{X}$ liegen. Beide Grenzfälle sind im Folgenden aufgeführt:

- $i \leq c$
- $i > n - c$

Für den Fall $i \leq c$ existieren links von i nicht genügend Datenpunkte für ein vollständiges gleitendes Fenster mit $2c + 1$ Datenpunkten. Für den Fall $i > n - c$ existieren rechts von i nicht genügend Datenpunkte. Dadurch verkleinert sich das gleitende Fenster beim Erreichen der Grenzfälle. Die Länge des gleitenden Fensters ist somit durch $(x_{\max(1,i-c)}, \ldots, x_{\min(n,i+c)})$ Datenpunkte für ein beliebiges $i \in [1, n]$ definiert.

Für eine Approximation der Steigungswerte steht im besten Fall ein gleitendes Fenster der Länge $2c+1$ zur Verfügung. In den Grenzfällen jedoch mindestens ein gleitendes Fenster der Länge $c + 1$.

An dieser Stelle wird die Definition eines arithmetischen Mittels eingeführt. Die Indizes $a, b \in [1, n]$ grenzen einen Bereich ein. Ausschließlich dieser Bereich wird für die Berechnung des arithmetischen Mittels in Betracht gezogen. Die Grenzen des Bereichs sind über folgende Gleichung 2.3 definiert.

$$1 \leq a < b \leq n \tag{2.3}$$

Das arithmetische Mittel für x Werte im Bereich $[a, b]$ ist über Gleichung 2.4 definiert, wobei ein Laufindex $h \in [a, b]$ innerhalb des definierten Bereichs existiert.

$$\bar{x}_{[a,b]} = \frac{\sum_{h=a}^{b} x_h}{b - a + 1} \tag{2.4}$$

Das arithmetische Mittel $\bar{y}_{[a,b]}$ für y Werte verhält sich analog zur Gleichung 2.4.

Die Punkte $((x_a, y_a), \ldots, (x_b, y_b))$ mit $a = \max(1, i-c), b = \max(1, i-c)$ im gleitenden Fenster liefern die Zielpunkte für eine Ausgleichsgerade. Die Ausgleichsgerade kann durch lineare Regression ermittelt werden. An dieser Stelle ist weniger die Ausgleichsgerade das Ziel sondern deren Steigung. Die Anstiegsapproximation s_i für x_i ist durch den Anstieg der Ausgleichsgeraden über die Gleichung 2.5 mit $a = \max(1, i - c), b = \max(1, i - c)$ definiert.

$$s_i = \frac{\sum_{h=a}^{b}(x_h - \bar{x}_{[a,b]})(y_h - \bar{y}_{[a,b]})}{\sum_{h=a}^{b}(x_h - \bar{x}_{[a,b]})^2} \tag{2.5}$$

Die erste Ableitung y' der diskreten Funktion y wird im Folgenden durch die Funktion $s : [x_1, x_n] \to \mathbb{R}$ approximiert und ist in folgender Gleichung 2.6 definiert.

$$y'(x \in \mathbb{X}) \approx s(x \in \mathbb{X}) \tag{2.6}$$

Über Interpolation wird Funktion $s(x)$ für $x \in [x_1, x_n]$ analog zu $y(x)$ in der Gleichung 2.2 definiert.

2.2 Referenzpunkte und Filtering

Bei einem Referenzpunkt handelt es sich um einen wichtigen Punkt innerhalb des Definitionsbereichs eines Profils. Ein Referenzpunkt kann einen von sechs Typen annehmen: Startpunkt, Endpunkt, Hochpunkt, Tiefpunkt, Wendepunkt steigend und Wendepunkt fallend. Alle Referenzpunkte sind durch Menge $\mathbb{R} \subseteq \mathbb{X}$ definiert. Die verschiedenen Typen werden nun im Einzelnen erläutert.

Der **Startpunkt** $\in (\mathbb{R}_S \subseteq \mathbb{R})$ markiert den ersten Punkt eines Profils $\mathbb{R}_S = \{x_1\}$. Somit hat jedes Profil genau einen Startpunkt.

Der **Endpunkt** $\in (\mathbb{R}_E \subseteq \mathbb{R})$) verhält sich analog zu $\mathbb{R}_S$, jedoch mit dem Unterschied, dass der Endpunkt den letzten Punkt eines Profils markiert $\mathbb{R}_E = \{x_n\}$. Jedes Profil besitzt genau einen Endpunkt.

Der **Hochpunkt** $\in (\mathbb{R}_H \subseteq \mathbb{R})$ kennzeichnet ein Maximum in den Profildaten. Dabei wird zunächst geprüft, ob der y_i-Wert eines Punkts an Stelle i echt größer als der direkte Vorgänger und echt größer als der direkte Nachfolger ist. In diesem Fall liegt der Hochpunkt an Stelle i. Im anderen Fall wird geprüft, ob es sich um ein Plateau handelt. Das bedeutet, alle Nachfolger bis $j > i$ haben identische Werte $y_j = y_i$ und es gilt $y_{j+1} < y_i$. Angenommen das Plateau endet an Stelle j, dann kann die Länge des Plateaus bestimmt werde. Die Länge definiert sich über $j + 1 - i$. Wenn das Plateau endet und der nächste Wert $y_j > y_{j+1}$ ist, wird der Hochpunkt an Stelle $i + \frac{j-i}{2}$ gesetzt.

Der Pseudocode in Listing 2.1 zeigt, wie ein Punkt $x_i \in \mathbb{R}_H$ ermittelt wird.

```
for i in n
  if y(i-1) < y(i) && y(i) >= y(i+1)
    if y(i) > y(i+1)
      // i is maximum
      // continue with i in the outer loop
    else
      for j = i in n
        if y(i) == y(j)
          // enlarge the plateau
          // continue with j in the inner loop
        else
          if y(i) > y(j)
            // i + ((j-i)/2) is maximum
            // and continue with i = j+1 in the outer loop
          else
            // ignore saddle point
            // and continue with i = j+1 in the outer loop
```

Listing 2.1: Bestimmung von $x_i \in \mathbb{R}_H$

Der **Tiefpunkt** $\in (\mathbb{R}_T \subseteq \mathbb{R})$ kennzeichnet ein Minimum in den Profildaten. Ein Punkt $x_i \in \mathbb{X}$ wird analog zum Pseudocode aus Listing 2.1 bestimmt, jedoch mit anderen Vergleichsoperatoren. Für die Tiefpunktbestimmung wird „<“ durch „>“ und „>“ durch „<“ ersetzt.

Der **Wendepunkt (steigend)** $\in (\mathbb{R}_{W_s} \subseteq \mathbb{R})$ kennzeichnet einen Wendepunkt mit positiver Steigung in den Profildaten. Anders als bei der Bestimmung von $\mathbb{R}_H$ und $\mathbb{R}_T$ werden bei $\mathbb{R}_{W_s}$ die Steigungswerte s an Stelle der Datenwerte y für die Wendepunktbestimmung verwendet. Ein Wendepunktpunkt $x_i \in \mathbb{R}_{W_s}$ wird

analog zur Hochpunktbestimmung ermittelt. Dabei handelt es sich um den selben Pseudocode wie in Listing 2.1, jedoch mit dem Unterschied, dass nun auf die Steigungswerte s zugegriffen wird. Somit ist $\mathbb{R}_{W_s}$ ein Maximum in Bezug auf die Steigungsdaten.

Der **Wendepunkt (fallend)** $\in (\mathbb{R}_{W_f} \subseteq \mathbb{R})$ kennzeichnet einen Wendepunkt mit negativer Steigung in den Profildaten. Die Bestimmung von $\mathbb{R}_{W_f}$ verhält sich analog zur Bestimmung von $\mathbb{R}_{W_s}$, jedoch mit umgekehrten Vergleichsoperatoren. Somit ist $\mathbb{R}_{W_f}$ ein Minimum in Bezug auf die Steigungsdaten eines Profils. Auch hier ist der Pseudocode für die Bestimmung ähnlich zu Listing 2.1.

Ein Profil kann relativ viele Referenzpunkte in $\mathbb{R}$ besitzen. Möglicherweise sind aber nur einige wenige von Wichtigkeit. Dabei hat der Endbenutzer zu entscheiden, welche Referenzpunkte für die Alignierung als wichtig erachtet werden. Für solch einen Fall existiert ein Mechanismus um Referenzpunkte zu verwerfen. Dieser Mechanismus ist ein Filter. Ausgenommen sind Startpunkt $\mathbb{R}_S$ und Endpunkt $\mathbb{R}_E$, die nicht verworfen werden dürfen. Im Folgenden werden zwei Filterarten beschrieben. Der Extrempunktfilter in Kapitel 2.2.1 entscheidet über die Einschränkung der Mengen R_H und R_T und liefert die eingeschränkte Extrempunktmenge $\mathbb{E}' \subset (\mathbb{R}_H \cup \mathbb{R}_T)$. Der Wendepunktfilter in Kapitel 2.2.2 entscheidet über die Einschränkung der Mengen R_{Ws} und R_{Wf} und liefert die eingeschränkte Wendepunktmenge $\mathbb{W}' \subset (\mathbb{R}_{W_s} \cup \mathbb{R}_{W_f})$.

Nach dem Anwenden beider Filtermechanismen steht die eingeschränkte Referenzpunktmenge $\mathbb{R}'$ zur Verfügung. Dabei setzt sich die eingeschränkte Referenzpunktmenge wie folgt zusammen: $\mathbb{R}' = \mathbb{R}_S \cup \mathbb{E}' \cup \mathbb{W}' \cup \mathbb{R}_E$.

2.2.1 Extrempunktfilter

Beim Extrempunktfilter handelt es sich um ein Rauschunterdrückungsverfahren. Rauschen ist eine Störungsinformation in den Daten. Ziel des Verfahrens ist es die Störung zu beseitigen oder vermindern. Für Extrempunkte wäre ein Störung die Identifizierung von Hoch- und Tiefpunkten mit sehr geringer Ausprägung. Das Rauschunterdrückungsverfahren bestimmt dabei ein Verhältnis zwischen globalem Hoch- und Tiefpunkt und ein Verhältnis zwischen lokalen Hoch- und Tiefpunkten. Ein Vergleich der Verhältnisse gibt Auskunft darüber wie stark die Störungsinformation vorliegt.

Folgende Abbildung 2.1 zeigt den direkten Vergleich eines Datensatzes mit Rauschen und das Ergebnis der Anwendung des Extremfilters mit 61% auf den Datensatz. Dabei sind die Extrempunkte mit den stärksten Ausprägungen erhalten geblieben. Die Extrempunkte mit geringen Ausprägungen wurden durch den Filter entfernt. Dazu zählen vor allem die Extrempunkte, welche im Band von $y \in [4, 12]$ lagen.

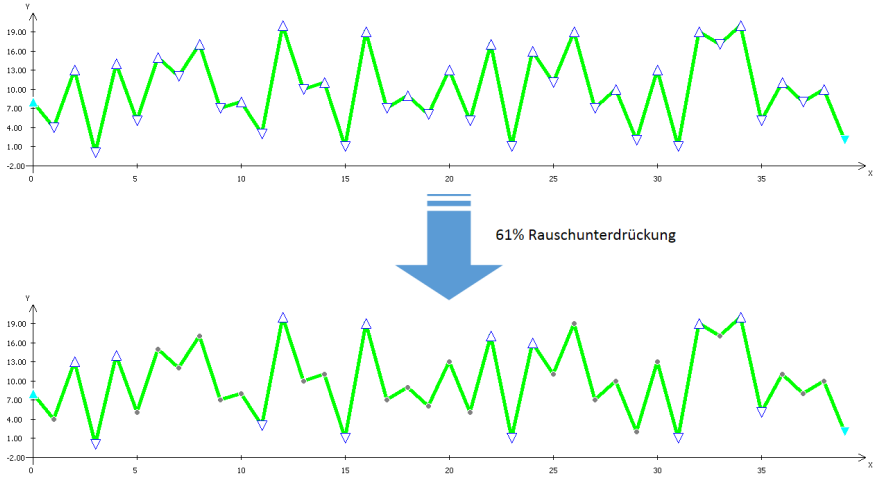

ABBILDUNG 2.1: Darstellung eines Profils mit vielen Extrempunkten (blaue Dreiecke) und hohem Rauschen. Nach Filteranwendung Unterdrückung des Rauschens und weniger Extrempunkte. Signifikante Ausprägungen bleiben erhalten.

Der Extrempunktfilter betrachtet nur Referenzpunkte $e \in \mathbb{R}_H \cup \mathbb{R}_T$. Die Menge $\mathbb{E}$ ist die Menge aller verfügbaren Hoch- und Tiefpunkte, welche sich über folgende Sequenz 2.7 definiert. Dabei ist der Startpunkt $\in \mathbb{R}_S$ und Endpunkt $\in \mathbb{R}_E$ auch in die Menge $\mathbb{E}$ mit aufgenommen. Der Grund dafür liegt darin, dass dadurch die äußersten Extrempunkte einen linken und rechten Nachbarn erhalten. Somit hat jeder Extrempunkt $e \in \mathbb{R}_H \cup \mathbb{R}_T$ genau zwei Nachbarn. Dies wiederum ist wichtig für die Berechnung des Verhältnisses eines jeden Extrempunkts $e \in \mathbb{R}_H \cup \mathbb{R}_T$ zu seinen Nachbarn in Menge $\mathbb{E}$ (vgl. Gleichung 2.13).

$$\mathbb{E} = \mathbb{R}_S \cup \mathbb{R}_H \cup \mathbb{R}_T \cup \mathbb{R}_E = \{e_1, \ldots, e_k\} \text{ mit } e_q < e_{q+1} \text{ und } q \in [1, k-1] \quad (2.7)$$

Dabei gilt für die Sequenz an Extrempunkten Gleichung 2.8, welche ausdrückt, dass das Vorkommen von Hoch- und Tiefpunkten in der Sequenz alterniert. Zudem gilt Bedingung 2.9 für die Menge $\mathbb{E}$, welche ausdrückt, dass der Start- und Endpunkt in die Menge aufgenommen werden. Wie bereits erwähnt, ist dies für

die später folgende Berechnung wichtig.

$$e_q \in \mathbb{R}_H \Leftrightarrow e_{q+1} \in \mathbb{R}_T \tag{2.8}$$

$$e_1 \in \mathbb{R}_S \wedge e_k \in \mathbb{R}_E \tag{2.9}$$

Der Filterwert f_E kann einen Prozentwert aus einem definierten Bereich annehmen (siehe Gleichung 2.10).

$$f_E \in [0, 100] \tag{2.10}$$

Wie zu Beginn dieses Kapitels erwähnt, existiert eine Relation zu einem globalen Hoch- und Tiefpunkt. Diese Relation liefert zusammen mit dem Filterwert einen Vergleichsoperator zur Bestimmung einer Extrempunkteinschränkung. Der globale Hochpunkt ist in Gleichung 2.11 definiert. Der globale Tiefpunkt ist in Gleichung 2.12 definiert. Für das globale Maximum und Minimum werden auch Start- und Endpunkt berücksichtigt.

$$e_{max} = \arg\max_{e \in \mathbb{E}} \{y(e)\} \tag{2.11}$$

$$e_{min} = \arg\min_{e \in \mathbb{E}} \{y(e)\} \tag{2.12}$$

Die Berechnung der eingeschränkten Extrempunktmenge $\mathbb{E}'$ ist in Gleichung 2.13 definiert und ist analog zum Filtern von Extrempunkten in [Bender et al., 2012]. Dabei müssen die beiden zuvor getroffenen Annahmen aus Gleichung 2.9 und Gleichung 2.13 gelten.

$$\begin{aligned} \mathbb{E}' = \{e_i \in \mathbb{E} | \forall_{i \in [2,k-1]} : \max\{|y(e_i) - y(e_{i-1})|, |y(e_i) - y(e_{i+1})|\} \\ > \frac{f_E}{100} \cdot (y(e_{max}) - y(e_{min}))\} \end{aligned} \tag{2.13}$$

Unter Beachtung von Gleichung 2.9 ist zu erkennen, dass in Gleichung 2.13 sowohl Start- und Endpunkt für die Berechnung des Verhältnisses verwendet werden. Dies ist vor allem von Bedeutung, wenn ein beliebiger Extrempunkt $e \in \mathbb{R}_H \cup \mathbb{R}_T$ keinen oder nur einen direkt angrenzenden Hoch- oder Tiefpunkt hat. Durch die Zuhilfenahme von Start- oder Endpunkt kann für jeden Extrempunkt $e \in \mathbb{R}_H \cup \mathbb{R}_T$ der Extrempunktfilter angewandt werden. Für den Start- und Endpunkt wird auf Grund von $i \in [2, k-1]$ keine Einschränkung angewandt. Beide Punkte werden somit nicht in die eingeschränkte Extrempunktmenge $\mathbb{E}'$ aufgenommen. Zudem ist

Gleichung 2.8 von Bedeutung, denn durch die Alternierung der Extrempunkttypen ist es möglich in Gleichung 2.13 die direkten Nachbarn zu verwenden. Dies ist essentiell, da für dieses Rauschunterdrückungsverfahren die Verhältnisse von Hoch- zu Tiefpunkt gemessen werden müssen. Außerdem lässt Gleichung 2.13 erkennen, dass der rechte Teil der Gleichung den Bereich $[0, y(e_{max}) - y(e_{min})]$ abdeckt. Der linke Teil der Gleichung 2.13 kann maximal $y(e_{max}) - y(e_{min})$ erreichen. Eine langsame Erhöhung von f_E hat, bei gleichbleibendem linken Teil der Gleichung 2.13, zur Folge, dass zunächst jene Extrempunkte eingeschränkt werden, welche den geringsten Wert aufweisen. Dadurch werden Extrempunkte mit der geringsten Ausprägung zuerst eingeschränkt und das Rauschen der Extrempunkte verringert.

Bei einem Filterwert $f_E = 0$ wird der rechte Teil von Gleichung 2.13 0. Dadurch werden keine Extrempunkte eingeschränkt und das Ergebnis nach der Filteroperation ist $\mathbb{E}' = \mathbb{E} \setminus (\mathbb{R}_S \cup \mathbb{R}_E)$. Im Gegensatz dazu hat ein Filterwert $f_E = 100$ zur Folge, dass alle Extrempunkte eingeschränkt werden. Als Ergebnis der Filterausführung ist die eingeschränkte Extrempunktmenge leer: $\mathbb{E}' = \emptyset$.

Folgendes Beispiel verdeutlicht die Anwendung des Extrempunktfilters. Das dabei verwendete Profil P ist durch Wertetabelle 2.1 definiert. Die grafische Darstellung ist in Abbildung 2.2 zu sehen. Dabei zeigt die Abbildung nur die relevanten Aspekte. Die Wendepunkte wurden aus Übersichtlichkeitsgründen entfernt.

x_i	$y(x_i)$	$\mathbb{R}$	x_i	$y(x_i)$	$\mathbb{R}$
1	7,0	$e_1 \in \mathbb{R}_S$	13	4,0	$e_4 \in \mathbb{R}_T$
2	6,75	-	14	5,2	-
3	6,5	-	15	6,4	-
4	6,25	-	16	7,6	-
5	6,0	$e_2 \in \mathbb{R}_T$	17	8,8	-
6	6,5	-	18	10,0	$e_5 \in \mathbb{R}_H$
7	7,0	-	19	9,714	-
8	7,5	-	20	9,429	-
9	8,0	$e_3 \in \mathbb{R}_H$	21	9,143	-
10	7,0	-	22	8,857	-
11	6,0	-	23	8,571	-
12	5,0	-	24	8,286	-
			25	8,0	$e_6 \in \mathbb{R}_E$

Tabelle 2.1: Wertetabelle zum Profil P aus dem Extrempunktfilterbeispiel zu Abbildung 2.2, mit Referenzpunkt $\mathbb{R}$ Relation

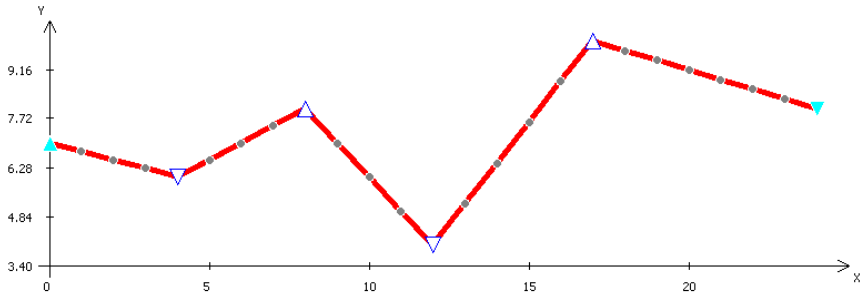

ABBILDUNG 2.2: Darstellung von Profil P aus Wertetabelle 2.1 mit vier Extrempunkten (blau), zwei davon Tiefpunkte (Dreieck nach unten) und zwei davon Hochpunkte (Dreieck nach oben), sowie Start- und Endpunkt (türkis), und Filterwert $f_E = 0$

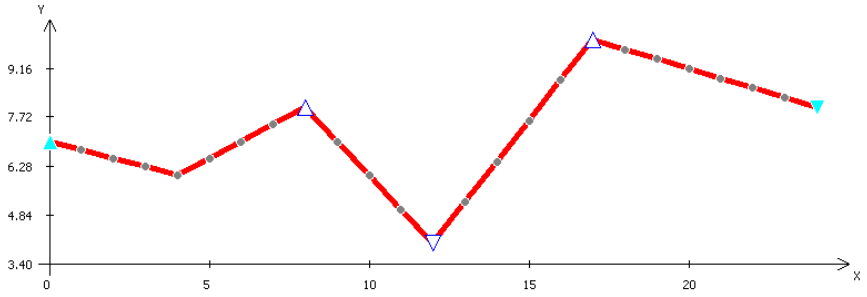

ABBILDUNG 2.3: Gleiches Profil P aus Abbildung 2.2, jedoch mit nur drei Extrempunkten und einem Filterwert $f_E = 33,5$

Wie Abbildung 2.2 zeigt, existieren genau vier Extrempunkte. Zusammen mit Start- und Endpunkt ergibt sich folgende Extrempunktmenge $\mathbb{E} = \{e_1, e_2, \ldots, e_6\}$, wobei $e_1 \in \mathbb{R}_S$ und $e_6 \in \mathbb{R}_E$. Die Hochpunkte sind $e_3, e_5 \in \mathbb{R}_H$ und die Tiefpunkte sind $e_2, e_4 \in \mathbb{R}_T$. Für die Filterung nach Gleichung 2.13 werden als globales Maximum $e_{max} = e_5$ und als globales Minimum $e_{min} = e_4$ ermittelt.

Im Folgenden wird für alle Extrempunkte $e \in \mathbb{E} \setminus (\mathbb{R}_S \cup \mathbb{R}_E)$ aus dem Beispiel der linke Teil der Gleichung 2.13 berechnet. Die Relation zwischen Datenpunkten und Extrempunkten kann der Wertetabelle 2.1 entnommen werden.

$$\begin{aligned} &\max\{|y(e_2) - y(e_1)|, |y(e_2) - y(e_3)|\} \\ &\qquad = \max\{|6-7|, |6-8|\} = \max\{1,2\} = 2 \qquad (2.14) \\ &\max\{|y(e_3) - y(e_2)|, |y(e_3) - y(e_4)|\} = \max\{2,4\} = 4 \qquad (2.15) \\ &\max\{|y(e_4) - y(e_3)|, |y(e_4) - y(e_5)|\} = \max\{4,6\} = 6 \qquad (2.16) \\ &\max\{|y(e_5) - y(e_4)|, |y(e_5) - y(e_6)|\} = \max\{6,2\} = 6 \qquad (2.17) \end{aligned}$$

In folgender Gleichung 2.18 wird der linke Teil der Gleichung 2.13 für $f_E = 0$ berechnet.

$$\frac{0}{100} \cdot (y(e_5) - y(e_4)) = 0 \cdot (10 - 4)) = 0 \cdot 6 = 0 \tag{2.18}$$

Vergleicht man alle Ergebnisse von Gleichungen 2.14, 2.15, 2.16 und 2.17 mit dem Ergebnis von Gleichung 2.18, ist in jedem Fall die „>“ Bedingung aus Gleichung 2.13 erfüllt. Deshalb wird für einen Filterwert $f_E = 0$ kein Extrempunkt eingeschränkt und das Resultat ist $\mathbb{E}' = \{e_2, e_3, e_4, e_5\}$.

Eine Anpassung des Filterwerts auf $f_E = 33,5$ hat nur eine Neuberechnung von Gleichung 2.18 zur Folge. Das Ergebnis ist durch Gleichung 2.19 dargestellt.

$$\frac{33,5}{100} \cdot 6 = 0,335 \cdot 6 = 2,01 \tag{2.19}$$

Ein Vergleich mit den zuvor ermittelten Ergebnissen mit Gleichung 2.19 zeigt, dass unter Berücksichtigung von Gleichung 2.13 die Gleichung 2.14 nicht erfüllt ist. Somit erhält man für einen Filterwert von $f_E = 33,5$ die eingeschränkte Extrempunktmenge $\mathbb{E}' = \{e_3, e_4, e_5\}$. Das Resultat dieses Filterwerts zeigt Abbildung 2.3. Sowohl die Abbildung als auch die berechneten Werte zeigen deutlich, dass zuerst der Extrempunkt mit der geringsten Ausprägung eingeschränkt wurde.

Wird der Filterwert sogar auf $f_E = 100$ erhöht, erhält man wie in Gleichung 2.20 zu sehen, 6 als Ergebnis des rechten Teils von Gleichung 2.13. In diesem Fall würden alle Extrempunkte inklusive dem globalen Maximum und Minimum eingeschränkt. Das Resultat ist eine leere eingeschränkte Extrempunktmenge $\mathbb{E}' = \emptyset$.

$$\frac{100}{100} \cdot 6 = 1 \cdot 6 = 6 \tag{2.20}$$

2.2.2 Wendepunktfilter

Der Wendepunktfilter kommt, anders als der Extrempunktfilter in Kapitel 2.2.1, ohne die Berechnung von Verhältnissen zu globalen Punkten in den Daten aus. Für das Einschränken von Punkten ist einzig der Steigungswert s eines Punktes und ein Schwellwert als Entscheidungskriterium notwendig.

Der Wendepunktfilter betrachtet nur Referenzpunkte $w \in \mathbb{R}_{W_s} \cup \mathbb{R}_{W_f}$. Dabei beinhaltet die Menge $\mathbb{W}$ alle Referenzpunkte vom Typ Wendepunkt, welche sich über

folgende Sequenz in Gleichung 2.21 definieren.

$$\mathbb{W} = \{w_1, \ldots, w_m\} = \mathbb{R}_{W_s} \cup \mathbb{R}_{W_f} \text{ mit } w_i < w_{i+1} \text{ und } i \in [1, m] \tag{2.21}$$

Der Schwellwert f_W des Wendepunktfilters, auch Filterwert genannt, wird mit den Steigungen der Wendepunkte $w \in \mathbb{R}_{W_s} \cup \mathbb{R}_{W_f}$ verglichen und kann folgende Werte annehmen:

$$f_W \geq 0 \tag{2.22}$$

Die Anwendung des Wendepunktfilters erzeugt als Ergebnis eine eingeschränkte Wendepunktmenge $\mathbb{W}'$. Diese ist in Gleichung 2.23 definiert und verhält sich analog zum Filtern von Wendepunkten in [Bender et al., 2012]. Dabei werden die Steigungswerte $s(w)$ eines jeden Wendepunkts mit dem Schwellwert verglichen. Ist der Steigungswert kleiner als der Schwellwert, so wird der Wendepunkt w eingeschränkt und ist nicht in $\mathbb{W}'$ enthalten.

$$\mathbb{W}' = \{w \in \mathbb{W} |\ |s(w)| \geq f_W\} \tag{2.23}$$

Folgendes Beispiel verdeutlicht, wie der Wendepunktfilter angewandt wird. Für das Beispiel wird das identische Profil verwendet, welches für das Extrempunktfilterbeispiel verwendet wurde (siehe Wertetabelle 2.1). Folgende Tabelle 2.2 ergänzt die Steigungswerte und die Wendepunkte des Profils P. Für die Berechnung der Steigungswerte wurde $c = 2$ als Radius für das gleitende Fenster verwendet.

In diesem Beispiel besitzt die Menge aller Wendepunkte genau drei Elemente $\mathbb{W} = \{w_1, w_2, w_3\}$. Die Wendepunkte sind entsprechend in Tabelle 2.2 aufgeführt. Auf Grund eines positiven Steigungswerts $s > 0$ werden $w_1, w_3 \in \mathbb{R}_{W_s}$ als steigende Wendepunkte identifiziert. Da $w_2 \in \mathbb{R}_{W_s}$ einen negativen Steigungswert $s < 0$ besitzt, handelt es sich um einen fallenden Wendepunkt. Die Wendepunkte sowie das Profil sind in Abbildung 2.4 mit einem Schwellwert von $f_W = 0$ dargestellt. In der Abbildung 2.4 wurden die Extrempunkte aus Übersichtlichkeitsgründen nicht dargestellt.

Ein Vergleich der Steigungswertbeträge $s(w \in \mathbb{W})$ aus Tabelle 2.2 mit dem Schwellwert $f_E = 0$ zeigt, dass alle drei Wendepunkte $w_1, w_2, w_3 \in \mathbb{W}$ die Bedingung,

x_i	$s(x_i)$	$\mathbb{R}$	x_i	$s(x_i)$	$\mathbb{R}$
1	-0,25	-	13	0,1	-
2	-0,25	-	14	0,76	-
3	-0,25	-	15	1,2	$w_3 \in \mathbb{R}_{W_s}$
4	-0,1	-	16	1,2	-
5	0,125	-	17	0,903	-
6	0,35	-	18	0,457	-
7	0,5	$w_1 \in \mathbb{R}_{W_s}$	19	0,011	-
8	0,2	-	20	-0,286	-
9	-0,25	-	21	-0,286	-
10	-0,7	-	22	-0,286	-
11	-1,0	$w_2 \in \mathbb{R}_{W_f}$	23	-0,286	-
12	-0,56	-	24	-0,286	-
			25	-0,286	-

TABELLE 2.2: Erweiterung der Wertetabelle um Steigungswerte zum Profil P aus dem Extrempunktfilterbeispiel zu Abbildung 2.2 und Abbildung 2.4, mit Referenzpunkt $\mathbb{R}$ Relation

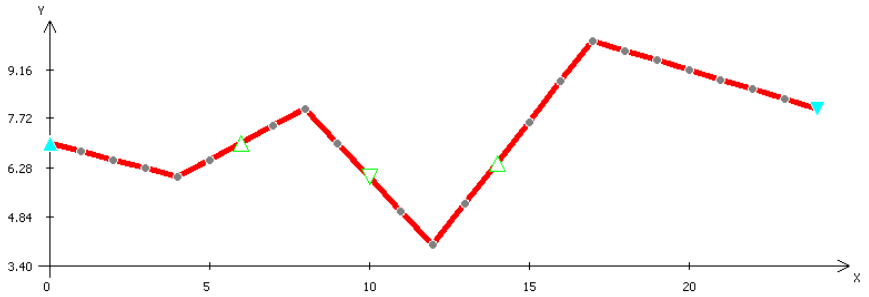

ABBILDUNG 2.4: Darstellung von Profil P aus Wertetabelle 2.1 mit drei Wendepunkten (grün), zwei davon steigend (Dreieck nach oben) und einer fallend (Dreieck nach unten), sowie Start- und Endpunkt (türkis), und Filterwert $f_W = 0$

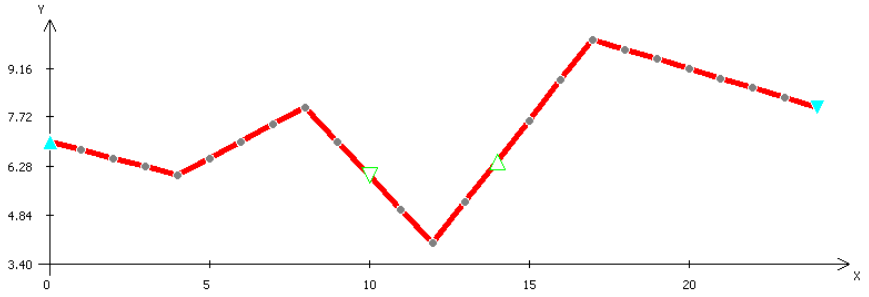

ABBILDUNG 2.5: Gleiches Profil P aus Abbildung 2.4, jedoch mit nur zwei Wendepunkten und einem Filterwert $f_W = 0,51$

welche in Gleichung 2.23 definiert ist, erfüllen. Die eingeschränkte Wendepunktmenge ist dabei identisch zur Wendepunktmenge $\mathbb{W}' = \mathbb{W}$. Dieses Szenario ist in Abbildung 2.4 dargestellt.

Bei einer Anpassung des Schwellwerts auf Wert $f_W = 0.51$ kommt es zu einer Einschränkung der Wendepunktmenge. Ein Vergleich mit Tabelle 2.2 zeigt, dass $|s(w_1)| = |0.5| = 0.5 \ngeq f_W$ die Bedingung nicht erfüllt. Im Gegensatz dazu führen die Berechnungen $|s(w_2)| = |-1| = 1 \geq f_W$ und $|s(w_3)| = 1,2 \geq f_W$ zu einer Erfüllung der Bedingung aus Gleichung 2.23. Somit erhält man für einen Schwellwert von $f_W = 0.51$ die eingeschränkte Wendepunktmenge $\mathbb{W}' = \mathbb{W} \setminus \{w_1\} = \{w_2, w_3\}$. Das Resultat dieses Schwellwerts ist in Abbildung 2.5 dargestellt.

Werden alle Wendepunkte eingeschränkt, ist das Ergebnis die leere Menge $\mathbb{W}' = \emptyset$. Um diesen Fall zu erzeugen werden die Beträge der Steigungswerte aus Tabelle 2.2 betrachtet. Durch $\arg\max_{w \in \mathbb{W}}\{|s(w)|\}$ erhält man mit $|s(w_3)| = 1.2$ den maximalen Steigungswert. Eine Anpassung des Schwellwerts auf $f_W = 1.3$ würde somit alle Wendepunkte einschränken.

2.3 Intervallzerlegung

Um auf die Intervallzerlegung eingehen zu können wird an dieser Stelle die Definition eines diskreten Intervalls I eingeführt. Gleichung 2.24 zeigt die Intervalldefinition. Dabei sind $a < b \in \mathbb{X}$ die Grenzen des Intervalls. Das bedeutet, dass die linke Intervallgrenze echt kleiner ist als die Rechte und dass die Intervallgrenzen innerhalb der durch das Profil gegebenen Datenpunkte liegen müssen. Zudem gibt $l \in \mathbb{N}^+$ die Anzahl der Punkte im Intervall an. Die Anzahl der Punkte im Intervall ist dabei immer größer als eins ($l > 1$).

$$I = [a, b]_l \text{ mit } a < b \in \mathbb{X} \text{ und } l \in \mathbb{N}^+ \tag{2.24}$$

Durch l besteht die Möglichkeit ein Intervall $[a, b]_l$ zu verzerren. Abhängig von der Anzahl an erforderlichen Punkten l im Intervall findet eine Streckung oder Stauchung statt. Für ein gegebenes Intervall $[a, b]_l$ werden die y-Werte definiert wie es Gleichung 2.25 darstellt. Abhängig von l werden, wie Gleichung 2.26 zeigt, die x-Werte zwischen den Intervallgrenzen a, b definiert. Dabei werden inklusive den Intervallgrenzen genau l Punkte für das Intervall zugänglich gemacht. Dadurch

ist es möglich das Intervall als kontinuierlichen Bereich zu betrachten. Analog zu Gleichung 2.25 können die Steigungswerte $s([a,b]_l)$ für ein Intervall bestimmt werden, indem in der Gleichung y durch s ersetzt wird.

$$y([a,b]_l) = (y(a), \ldots, y(u), \ldots, y(b)) \tag{2.25}$$

$$u = a + \frac{b-a+1}{l} \cdot t \text{ für } 1 < t < l \tag{2.26}$$

Ein Profil mit n Datenpunkten besitzt ein implizites Intervall (Gleichung 2.27). Im Folgenden wird hier vom initialen Intervall I_{init} eines Profils gesprochen. Dabei entspricht die linke Intervallgrenze dem ersten Punkt im Profil und die rechte Intervallgrenze dem letzten Punkt im Profil. Da das Profil n Datenpunkte hat, existieren zwischen beiden Grenzen $x_1, x_n \in \mathbb{X}$ genau $n-2$ Punkte im Intervall. Das initiale Intervall, inklusive der Grenzen, umfasst somit das gesamte Profil mit n Punkten.

$$I_{init} = [x_1, x_n]_n \tag{2.27}$$

Die Zerlegung $\mathbb{I}$ ist eine geordnete Menge von Intervallen, welche in Gleichung 2.28 definiert ist.

$$\mathbb{I} = \{I_1, \ldots, I_z\} \text{ für } I_i = [a_i, b_i]_{l_i} \text{ mit } b_i < a_{i+1} \tag{2.28}$$

Die Intervallzerlegung $\mathbb{I}$ sieht zudem vor, ein gegebenes Profil so aufzuteilen, dass für die Zerlegung die zwei folgenden Eigenschaften zutreffen:

- die Zerlegung ist disjunkt
- die Zerlegung ist vollständig

Die Zerlegung $\mathbb{I}$ ist disjunkt, falls folgende Bedingung 2.29 gilt.

$$\forall_{I \in \mathbb{I}} : \nexists_{I' \in \mathbb{I}} : (a < a' < b) \vee (a < b' < b) \text{ für } I = [a,b]_l, I' = [a',b']_{l'} \tag{2.29}$$

Eine Zerlegung $\mathbb{I}$ ist vollständig, genau dann, wenn Gleichung 2.30 gilt. Zudem ergibt die Konkatenation aller Intervalle das initiale Intervall I_{init}.

$$\begin{aligned} \exists_{I \in \mathbb{I}} : a = x_1 \\ \exists_{I \in \mathbb{I}} : b = x_n \\ \forall_{I \in \mathbb{I}} : (b = n) \vee \exists_{I' \in \mathbb{I}} : (b = a') \text{ für } I = [a,b]_l, I' = [a',b']_{l'} \end{aligned} \tag{2.30}$$

Wie die Intervallzerlegung bestimmt wird, ist in Kapitel 3.2.2 beschrieben.

Kapitel 3

Methoden

Dieses Kapitel befasst sich mit den verschiedenen Methoden zur Alignierung. Zunächst wird die paarweise Alignierung beschrieben. Diese Art der Alignierung liefert die Grundlage für die darauf folgenden multiplen Alignierungsmethoden. Am Ende dieses Kapitels wird auf die referenzbasierte Alignierung eingegangen. Zu allen drei Alignierungsmethoden wird zudem eine intervallbasierte Umsetzung beschrieben.

3.1 Paarweise Alignierung

Das Ziel der paarweisen Alignierung ist für zwei gegebene Profile eine intervallbasierte Verzerrung zu bestimmen. Dabei sollen ähnliche Profilteile im gleichen x-Bereich abgebildet werden. Auf Grundlage dieser Verzerrung wird das Konsensusprofil K abgeleitet. Dabei stellt K die gemeinsamen Charakteristika der Einzelprofile dar.

Für zwei Profile P und P' ist dabei ein Alignment: $\mathbb{A} = (\mathbb{I}, \mathbb{I}')$, wobei Intervallzerlegung $\mathbb{I}$ aus P und $\mathbb{I}'$ aus P' bestimmt ist (vgl. Abbildung 3.1 (1)). Beide Zerlegungen haben dabei dieselbe Länge $|\mathbb{I}| = |\mathbb{I}'| = z$. Die Bestimmung der Zerlegung ist nicht Teil des paarweisen Alignments, weshalb an dieser Stelle nicht näher darauf eingegangen wird. Jeweils ein Paar von Intervallen (I_i, I'_i) aus den Zerlegungen $\mathbb{I}, \mathbb{I}'$ wird für $1 \leq i \leq z$ aligniert (vgl. Abbildung 3.1 (2)). Für die Sequenz der alignierten Profilintervalle lässt sich im Anschluss ein mittleres Konsensusprofil K bestimmen. Bei Profil K handelt es sich um das Konsensusprofil des Alignments

$\mathbb{A}$ (vgl. Abbildung 3.1 (3) + (4)). Konsensus K soll bei der paarweisen Alignierung die Gemeinsamkeiten der Ausgangsprofile P, P' repräsentieren.

Im Folgenden wird definiert, wie die Daten des Konsensusprofils K berechnet werden. Gegeben sind zwei Intervalle $I = [a, b]_l$ und $I' = [a', b']_{l'}$. Ziel folgender Gleichung 3.3 ist die Berechnung der Werte im Konsensusintervall $\hat{I}$. Zunächst wird Gleichung 2.25 durch 3.1 erweitert. Dabei ist $i \in [1, l]$ die Iteration im Intervall. Gleichung 3.2 definiert die Berechnung der neuen Intervalllänge $\hat{l}$ für $\hat{I}$. Die Berechnung der jeweiligen y-Werte in Zerlegung $\hat{\mathbb{I}}$ für K erfolgt durch Gleichung 3.3. Analog zu Gleichung 3.3 werden die Steigungswerte $s([\hat{a}, \hat{b}]_{\hat{l}})_i$ berechnet.

$$y([a, b]_l)_i \text{ mit } i \in [1, l] \tag{3.1}$$

$$\hat{l} = \frac{l + l'}{2} \tag{3.2}$$

$$y([\hat{a}, \hat{b}]_{\hat{l}})_i = \frac{y([a, b]_{\hat{l}})_i + y([a', b']_{\hat{l}})_i}{2} \text{ mit } i \in [1, \hat{l}]$$
$$\text{und } \hat{a} = \frac{a + a'}{2}, \hat{b} = \frac{b + b'}{2} \tag{3.3}$$

Die Zerlegung $\hat{\mathbb{I}}$ des entsprechenden Konsensusprofils K ergibt sich aus den vollständigen und disjunkten Zerlegungen $\mathbb{I}$ und $\mathbb{I}'$ von P und P', wie in Gleichung 3.3 beschrieben.

Wie in Abbildung 3.1 zu sehen ist, basiert die paarweise Alignierung auf paarweisen Intervallalignments. Folgendes Beispiel verdeutlicht die Vorgehensweise eines paarweisen Intervallalignments. Gegeben sind zwei Profile P, P'. Die dazugehörigen Wertetabellen befinden sich im Anhang A.2. Beide Profile sind in Abbildung 3.3 dargestellt und haben $n = 25$ Datenpunkte. Für die paarweise Intervallalignierung $A = (I, I')$ sind zwei Intervalle notwendig. Im einfachsten Fall wird dazu für Profil P das initiale Intervall $I = I_{init}$ und für das Profil P' das initiale Intervall $I' = I'_{init}$ definiert.

Ein paarweises Intervallalignment beider initialer Intervalle ist in diesem Beispiel allerdings kein Alignment im eigentlichen Sinne. Da beide Intervalle dieselbe Anzahl an Punkten haben, findet im Folgenden keine Verzerrung der Intervalle statt. Das bedeutet alle Punkte der Profile bleiben für die Berechnung an denselben Positionen erhalten, da weder Punkte eingefügt noch entfernt werden. Um y-Werte des Konsensusprofils K für diesen Spezialfall ohne Verzerrung zu bestimmen, wird vereinfachte Gleichung 3.4 statt Gleichung 3.3 verwendet. Denn die y Werte können

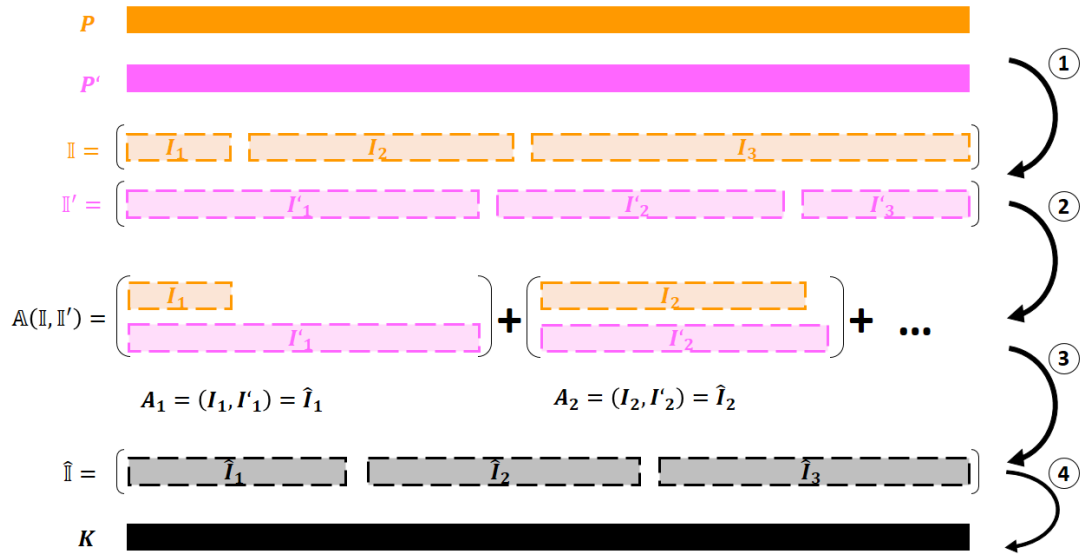

Abbildung 3.1: Paarweise Alignierung: (1) Bestimmung der Zerlegungen $\mathbb{I}, \mathbb{I}'$ aus P, P', (2) paarweises Intervallalignment, (3) Alignierung (Strecken/Zerren) der Intervallpaare (siehe Abbildung 3.2 (1), (4) Erzeugen des Konsensusprofils K über Mittelwerte der alignierten Profile (siehe Abbildung 3.2 (2))

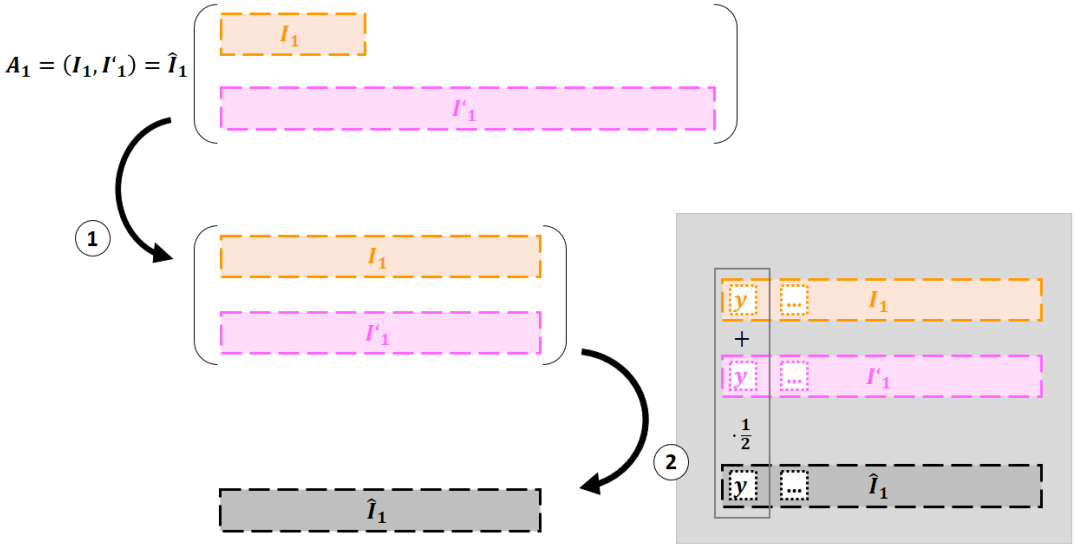

Abbildung 3.2: Paarweises Intervallalignment (Schritt (3) in Abbildung 3.1): Gegeben zwei Intervalle unterschiedlicher Anzahl an Datenpunkten. (1) Verzerren der Intervalle auf die mittlere Länge, (2) Bestimmung der Daten im Ergebnisintervall $\hat{I}_1$ im Konsensusprofil K

durch den nativen Mittelwert aus beiden Intervallen I_{init}, I'_{init} gebildet werden (vgl. Abbildung 3.2 ohne Schritt (1)).

$$\forall_{1 \leq i \leq n} : \hat{y}_i = \frac{y_i + y'_i}{2} \tag{3.4}$$

Das Resultat für die Berechnung des Beispiels mit Formel 3.4 und den initialen Intervallen I_{init}, I'_{init} zeigt Abbildung 3.4.

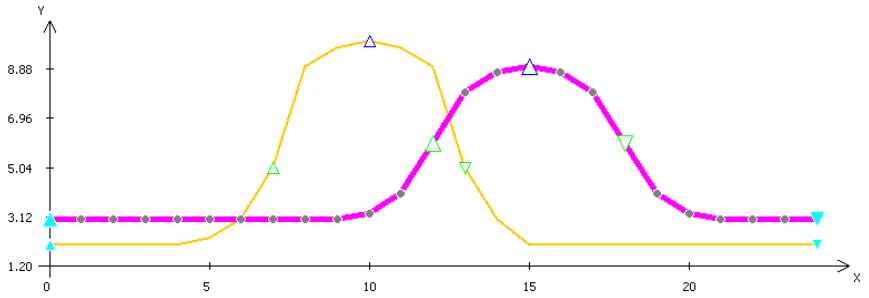

Abbildung 3.3: Profil P (orange) und P' (pink) aus Beispiel von Kapitel 3.1 mit sechs Referenzpunkten

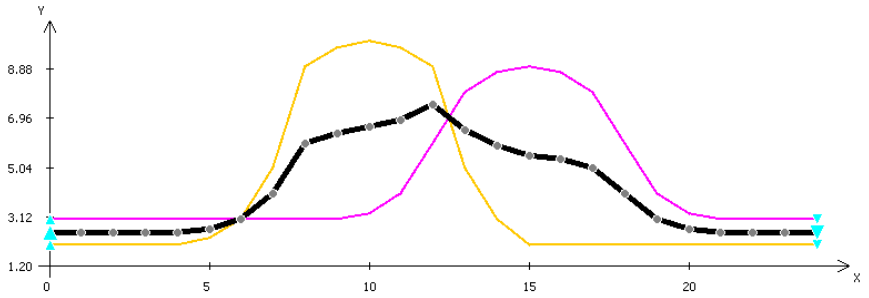

Abbildung 3.4: Profil P (orange) und P' (pink) aus Beispiel von Kapitel 3.1 mit $\mathbb{A} = (I_{init}, I'_{init})$ und Konsensus K (schwarz)

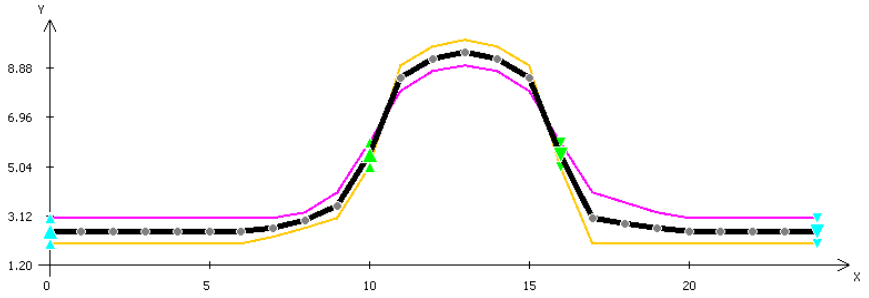

Abbildung 3.5: Korrespondierende verzerrte Profile im Vergleich mit vorheriger Abbildung, und $\mathbb{A} = (\mathbb{I}, \mathbb{I}')$ mit Konsensus K (schwarz)

Bei einem Vergleich beider Ausgangskurven mit dem Konsensus in Abbildung 3.4 hat K nur einige wenige Gemeinsamkeiten. Dies kann durch eine vorausgehende Bestimmung eines optimalen Alignments verhindert werden. In diesem Fall wird keine Alignierung von beiden initialen Intervallen durchgeführt, sondern eine Alignierung zweier Zerlegungen berechnet (vgl. Abbildung 3.1 (1) und (2)). Bei genauer Betrachtung lässt sich erkennen, dass eine Alignierung $\mathbb{A} = (\mathbb{I}, \mathbb{I}')$ eine Sequenz von paarweisen Intervallalignments ist. In Abbildung 3.2 Schritt (1) ist dargestellt, wie zwei Intervalle mit unterschiedlicher Anzahl an Datenpunkten zunächst aligniert werden. Gegeben sind folgende zwei Intervalle $I_1 = [1, 7]_{10}$, $I_1' = [1, 12]_{10}$ für P, P'. Die Länge von I_1 ist mit 7 kleiner als die Länge von I_1' mit 12. Im Alignierungsschritt werden nun für beide Intervalle dieselbe Anzahl an Datenpunkten benötigt. Dazu wird aus den Intervalllängen die mittlere Anzahl an Datenpunkten $\hat{l}$ ermittelt. Die Berechnung für $\hat{l}$ ist in Gleichung 3.5 definiert.

$$\hat{l} = \frac{l + l'}{2} \text{ mit } I = [a, b]_l, I' = [a', b']_{l'}, l \in \mathbb{N}^+ \tag{3.5}$$

In diesem Beispiel ist $\hat{l} = 10$. Wie bereits in Formel 2.25 und 2.26 definiert ist, müssen für I_1 drei Punkte durch Interpolation hinzugefügt werden. Analog dazu werden in I_1' zwei Punkte verworfen. Da nun beide Intervalle die gleiche Anzahl an Datenpunkten zur Verfügung stellen, kann, mit Hilfe von Gleichung 3.3, die Mittelwertberechnung für das Zielintervall des Konsensusprofils K erfolgen (vgl. Abbildung 3.2 (2)).

Sind alle paarweisen Intervallalignierungen durchgeführt, ist die Alignierung beider Zerlegungen abgeschlossen und das Konsensusprofil K liegt vor (vgl. Abbildung 3.1 (4)). Das Ergebnis der Alignierung zeigt Abbildung 3.5.

3.2 Paarweises intervallbasiertes Kurvenalignment (PICA)

Das paarweise intervallbasierte Kurvenalignment (pairwise intervall-based curve alignment, PICA) bestimmt möglichst optimale Intervallalignments für zwei gegebene Profile. Die PICA-Methode verfolgt dabei einen Greedy-Ansatz, welcher die Distanz eines Alignments minimiert. Dieser Ansatz wurde in [Bender et al., 2012] eingeführt und wird im Folgenden detailliert dargestellt.

Ein Greedy-Algorithmus trifft die besten lokalen Entscheidungen. Konkret bedeutet dies für die PICA-Methode, dass für ein Paar aus gegebenen Intervallen (I, I') eine binäre Zerlegung gesucht wird, welche die Distanz am besten minimiert. Eine binäre Zerlegung bedeutet, dass genau eine Stelle in einem Intervall benutzt wird um das Intervall in genau zwei Intervalle zu zerlegen. Für das Paar aus Intervallen entsteht so ein linker und rechter Teil. Die Summen der Distanzen aus linkem und rechtem Teil müssen dabei echt kleiner sein als die Distanz beider Ausgangsintervalle, damit die Zerlegung als besser erachtet wird. Über alle möglichen Kombinationen aus Zerlegungspunkten bestimmt der Greedy-Algorithmus so die beste lokale Zerlegung, welche die minimale Distanz aufweist. Im nächsten Schritt wird der Greedy-Algorithmus für den linken und rechten Teil wiederum die beste lokale Entscheidung treffen. Die Summe der lokalen Entscheidungen ergeben, aus globaler Sicht, vermutlich nicht die minimalste, jedoch hinreichend optimale Distanzen für die Zerlegung $(\mathbb{I}, \mathbb{I}')$ für das Paar (I, I'). Aus Performancegründen wird der Greedy-Algorithmus zur Bestimmung der optimalen Zerlegung verwendet. Nach erfolgreicher Bestimmung der optimalen Zerlegung wird, wie in Kapitel 3.1 behandelt, die paarweise Konsensusbestimmung durchgeführt. Das Ergebnis, welches durch eine optimale Zerlegung durch die PICA-Methode zu Stande kommt, ist in Abbildung 3.5 zu sehen.

3.2.1 Distanzberechnung

Wie bereits in Kapitel 3.2 erwähnt ist, erfordert der Greedy-Ansatz die Bestimmung einer Distanz zwischen zwei Intervallen I, I'. Die Distanz zwischen zwei Intervallen gibt an, wie unterschiedlich zwei Intervalle zueinander sind. Dabei bedeutet eine Distanz von 0, dass beide Intervalle identisch sind. Daraus lässt sich folgern, je höher der Wert der Distanz, desto unterschiedlicher sind die beiden verglichenen Intervalle.

Bei der Distanzberechnung wird der Grad der Verzerrung beider Intervalle I, I' für die Konsensusbestimmung berücksichtigt. Dieser lässt sich über ein Verhältnis der Datenpunkte im Intervall vor dem Verzerren und nach dem Verzerren feststellen.

Der Grad einer Verzerrung eines Intervalls $I = [a, b]_l$ ist durch den Verzerrungsfaktor v definiert (vgl. Gleichung 3.6). Dabei gibt $\hat{l}$ die mittlere Anzahl an Datenpunkten im korrespondierenden verzerrten Intervall $\hat{I}$ an (vgl. Gleichung 3.7).

$$v(I, \hat{l}) = v([a, b]_l, \hat{l}) = \frac{l}{\hat{l}} \tag{3.6}$$

$$\hat{l} = \frac{l + l'}{2} \text{ mit } I = [a, b]_l, I'[a', b']_{l'}, \hat{I} = [\hat{a}, \hat{b}]_{\hat{l}} \tag{3.7}$$

Die Gleichung 3.8 definiert die Distanzoperation zwischen zwei Intervallen I, I' auf der Grundlage der Steigungswerte s. Während die Datenwerte y ausschließlich die absolute Position eines Punktes des Profils beschreiben, liefert ein Steigungswert die Information wie ein Punkt in der Relation zu seinen benachbarten Punkten steht. Deshalb beschreibt ein Steigungswert besser die Gestalt der Profildaten als dies der Datenwert tut. Durch die Verwendung der Steigungswerte s für die Distanzberechnung erhält man einen Distanzwert, welcher besser im Bezug auf die Gestalterhaltung interpretiert werden kann. Dabei werden die beiden Ausgangsintervalle $I = [a, b]_l, I'[a', b']_{l'}$ für die Distanzberechnung auf die Anzahl der Datenpunkte im Zielintervall $\hat{I}$ mit $\hat{l}$ verzerrt. Zudem wird der erste Punkt im Intervall nicht berücksichtigt und muss gegebenenfalls außerhalb der Distanzberechnung ermittelt werden. Dies ist notwendig, da sonst bei der Distanzbestimmung von vollständigen und disjunkten Zerlegungen die Intervallgrenzen mehrfach in die Distanzberechnung einfließen.

$$\begin{aligned} d_s(I, I') = &\quad d_s([a, b]_{\hat{l}}, [a', b']_{\hat{l}'}) \\ = &\quad \sum_{i=2}^{\hat{l}} (v(I, \hat{l}) \cdot s([a, b]_{\hat{l}})_i - v(I', \hat{l}) \cdot s([a', b']_{\hat{l}})_i)^2 \end{aligned} \tag{3.8}$$

Analog zu Gleichung 3.8 ist $d_y(I, I')$ als die Distanz zweier Intervalle auf Basis der y-Werte, anstelle der s-Werte, definiert.

Weiterführend wird die Distanzberechnung $RMSD_s(\mathbb{I}, \mathbb{I}')$ für zwei Zerlegungen $\mathbb{I}, \mathbb{I}'$ durch Gleichung 3.9 auf der Grundlage der Steigungswerte s definiert. $RMSD$ steht für „root mean square deviation“ (vgl. [Bender et al., 2012]). Dabei gibt $\hat{l}_i$ die mittlere Anzahl an Datenpunkten im korrespondierenden verzerrten Intervall $\hat{I}_i$ von $I_i \in \mathbb{I}$ und $I'_i \in \mathbb{I}'$ an. Zudem wird die Distanz für den ersten Punkt in der Zerlegung extra durch d_{s_1} berechnet (vgl. Gleichung 3.10), da aufgrund von

Gleichung 3.8 dieser Punkt nicht berücksichtigt würde.

$$RMSD_s(\mathbb{I}, \mathbb{I}') = \sqrt[2]{\frac{d_{s_1} + \sum_{i=1}^{|\mathbb{I}|} d_s(I_i, I_i')}{1 + \sum_{i=1}^{|\mathbb{I}|} \hat{l}_i - 1}} \text{ mit } |\mathbb{I}| = |\mathbb{I}'| \tag{3.9}$$

$$d_{s_1} = (v(I_1 \in \mathbb{I}, \hat{l}_1) \cdot s_1 - v(I_1' \in \mathbb{I}', \hat{l}_1) \cdot s_1')^2 \tag{3.10}$$

Analog zu Gleichung 3.9 ist $RMSD_y(\mathbb{I}, \mathbb{I}')$ als Distanz für zwei Zerlegungen auf Basis der y-Werte definiert.

Im Folgenden wird für die Distanzberechnung immer die steigungsbasierte Variante verwendet. Dabei wird zur Vereinfachung $RMSD$ statt $RMSD_s$ und d statt d_s verwendet.

3.2.2 Intervallzerlegung

Der Pseudocode in Listing 3.1 zeigt die Bestimmung der besten Intervallzerlegung für zwei Profile P, P'. Zu Beginn existieren zwei Intervalle $I = I_{init} = [a, b]_l$, $I'_{init} = [a', b']_{l'}$. Über die verfügbaren Referenzpunkte $c, c' \in \mathbb{R} \backslash (\mathbb{R}_S \cup \mathbb{R}_E)$ der Profile P, P' wird versucht die Ausgangsintervalle zu zerlegen, wobei gilt $a < c < b$ und $a' < c' < b'$. Bei dem Zerlegungsschritt handelt es sich um eine binäre Zerlegung. Als Beispiel dafür steht der gewählte Zerlegungspunkt c im Intervall I, welcher genau zwei Teilintervalle $\check{I}_1, \check{I}_2$ erzeugt (vgl. Abbildung 3.6). Die Zerlegung für I ist durch Gleichung 3.11 definiert und analog dazu für I' in Gleichung 3.12. Durch die Bedingung in Gleichung 3.13 ist sichergestellt, dass eine Zerlegung genau dann als besser erachtet wird, wenn die Distanz der Zerlegung geringer ist als die Distanz beider Ausgangsintervalle. Zudem ist Gleichung 3.13 entscheidend dafür, dass die Distanz der resultierenden Zerlegung minimal ist.

$$[a, b]_l \rightarrow [a, c]_x, [c, b]_y \text{ mit } x + y = l + 1, a < c < b \tag{3.11}$$

$$[a', b']_{l'} \rightarrow [a', c']_{x'}, [c', b']_{y'} \text{ mit } x' + y' = l' + 1, a' < c' < b' \tag{3.12}$$

$$d([a, c]_x, [a', c']_{x'}) + d([c, b]_y, [c', b']_{y'}) < d([a, b]_l, [a', b']_{l'}) \tag{3.13}$$

Die in Abbildung 3.6 dargestellte Zerlegung $(\check{\mathbb{I}}, \check{\mathbb{I}}')$ für (c, c') ist nur eine mögliche Zerlegung. Die beste Zerlegung für alle möglichen Zerlegungsstellen c, c' wird durch Berücksichtigung aller möglichen Zerlegungspunktkombinationen c, c' gefunden.

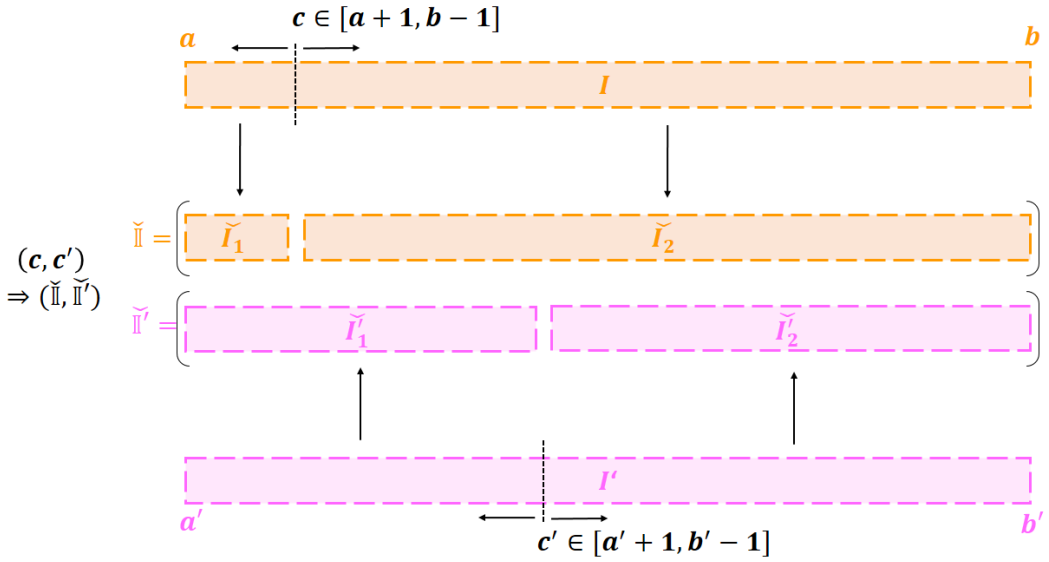

Abbildung 3.6: Intervalle I und I' mit resultierender Zerlegungen $\check{\mathbb{I}}$ und $\check{\mathbb{I}}'$ für gewählte Zerlegungspunkte c und c'.

Gilt für die so gefundene beste Zerlegung $(\check{\mathbb{I}}, \check{\mathbb{I}}')$ folgende Gleichung 3.14, so wird versucht die Zerlegung weiter zu zerlegen. Hierbei handelt es sich um die verkürzte Form der in Gleichung 3.13 bereits beschriebenen Bedingung.

$$d(\check{\mathbb{I}}, \check{\mathbb{I}}') < d(I, I') \tag{3.14}$$

```
// Given Profile P,P' with I=[a,b] covers P and I'=[a',b'] covers P'
optimalDecomposition(I,I')
   if I and I' decomposable
      bestDist = MAXVALUE
      for reference point R in I
         for reference point R' in I'
            if type R == type R'
               // decompose I to IL and IR at point R
               // decompose I' to IL' and IR' at point R'
               IL=[a,R],IR=[R,b],IL'=[a',R'],IR=[R',b']
               if IL,IR,IL',IR' > allowed decomposition length
                  distL = d(IL,IL')
                  distR = d(IR,IR')
                  if distL + distR < bestDist
                     // store IL,IR and IL',IR' as current best decomposition
                     bestIL = IL
                     bestIR = IR
                     bestIL' = IL'
                     bestIR' = IR'
                     bestDist = distL + distR
      if bestDist < d(I,I')
         // better local decomposition found
```

```
        // try to decompose again
        optimalDecomposition(bestIL,bestIL')
        optimalDecomposition(bestIR,bestIR')
     else
        // print I,I' as best local decomposition
  else
     // print I,I' as best local decomposition
```

LISTING 3.1: Pseudocode-Funktion zur Bestimmung der binären Intervallzerlegung mit minimaler Distanz $d(\mathbb{I}, \mathbb{I}')$

Nachdem alle Referenzpunkte, die in Intervall I und I' liegen, für die Aufteilung der jeweiligen Intervalle in Betracht gezogen wurden, liegt die optimale lokale Zerlegung vor (vgl. Listing 3.1 bestIL,bestIL' and bestIR,bestIR'). Ist jedoch keine bessere lokale Zerlegung zu I und I' bestimmt worden, wird I und I' als optimale lokale Zerlegung als Ergebnis zurückgegeben. Im ersten Fall wird durch Rekursion versucht die optimalen lokalen Zerlegungen weiter aufzuteilen. Dies geschieht so lange, bis die Intervalle nicht mehr aufgeteilt werden können. Das Abbruchkriterium für die Aufteilung eines Intervalls $I = [a, b]_l$ ist eine minimale Länge l_{min}. Ist $l_{min} > (a - b)$, so darf Intervall I nicht weiter zerlegt werden und beide Ausgangsintervalle werden als lokales Ergebnis zurückgegeben. Zum Schluss liegt die bestmögliche Zerlegung für I, I' vor, die die Distanz aus Gleichung 3.15 minimiert. Dabei ist eine Optimierung von Gleichung 3.15 analog zur Optimierung des $RMSD$ in Gleichung 3.9, da die Verhältnisse aus den Ergebnissen beider Gleichungen proportional ansteigen. Gleichung 3.15 verzichtet, im Gegensatz zu Gleichung 3.9, auf die Wurzelberechnung und die Normierung der Distanzwerte. Analog zu Gleichung 3.9 muss in Gleichung 3.15 auch die Distanz des ersten Punktes d_1 in beiden Profilen berechnet werden. In Bezug auf die Berechnungszeit ist Gleichung 3.15 schneller zu bestimmen. Deshalb wird bei der Bestimmung der Intervallzerlegung Gleichung 3.15 verwendet.

$$d(\mathbb{I}, \mathbb{I}') = d_1 + \sum_{i=1}^{|\mathbb{I}|} d(I_i, I_i') \text{ mit } |\mathbb{I}| = |\mathbb{I}'| \tag{3.15}$$

3.2.3 Laufzeitanalyse PICA

Für die Laufzeitanalyse wird die Intervallzerlegung der PICA-Methode betrachtet. Ausgegangen wird von zwei Profilen, für die eine optimale Zerlegung gefunden werden soll. Beginnend von den initialen Intervallen der beiden Profile wird ein binärer

Zerlegungsschritt durchgeführt. Dabei haben beide initialen Intervalle $O(n)$ Datenpunkte. In dem einen Intervall gibt es folglich $O(n)$ mögliche Stellen die Zerlegung durchzuführen. Im anderen Intervall gibt es auch genau $O(n)$ Stellen für eine Zerlegungsdurchführung. Daraus ergeben sich $O(n^2)$ mögliche Zerlegungen, die getestet werden müssen. Da für jeden Zerlegungstest die Distanz berechnet und die Intervalle verzerrt werden müssen, fällt zudem eine Laufzeitkomplexität von $O(2n) = O(n)$ an. Zudem sind $O(n)$ Zerlegungsschritte möglich. Dadurch ergibt sich eine resultierende Laufzeitkomplexität von $O(n^4)$ für die Intervallzerlegung.

Als Optimierung wurden im PICA-Verfahren nur Referenzpunkte als mögliche Zerlegungspunkte betrachtet. Dabei gibt es viel weniger Referenzpunkte als Datenpunkte $|\mathbb{R}| = r \ll n$. Dies führt zu somit lediglich zu $O(r^2)$ Zerlegungsüberprüfungen. Zudem wird die Laufzeitkomplexität für die Distanzberechnung und das Verzerren der Intervalle mit $O(n)$ für einen Zerlegungstest berücksichtigt. Allerdings sind nur $O(r)$ Zerlegungen möglich. Somit ergibt sich eine resultierende Laufzeitkomplexität für die Intervallzerlegung mit Referenzpunkten von $O(r^3 \cdot n)$.

Die Nutzung von Referenzpunkten für die Zerlegung liefert eine deutliche Laufzeitoptimierung $O(r^3 \cdot n) \ll O(n^4)$ für $r \ll n$.

3.3 Multiple Alignierung

Durch die multiple Alignierung wird aus einer Menge von Ausgangsprofilen $\mathbb{P} = \{P_1, \ldots, P_k\}$ ein Konsensusprofil K erstellt, welches die Charakteristika aller Profile aus $\mathbb{P}$ bestmöglich repräsentiert. Dabei wird K aus dem Alignment der Zerlegungen $\mathbb{I}_1, \ldots, \mathbb{I}_k$ der korrespondierenden Profile aus $\mathbb{P}$ gebildet. Für die Zerlegungen gilt $|\mathbb{I}_i| = |\mathbb{I}_j|$ für $1 \leq i < j \leq k$. Das Ziel der multiplen Alignierung ist die Minimierung der Distanz über die Zerlegungen $d(\mathbb{I}_1, \ldots, \mathbb{I}_k)$.

Bei einer Vorgehensweise analog zur PICA-Methode würde dieser Ansatz zu einer Laufzeitkomplexität von $O(r^{k+1} \cdot n)$ für die Intervallzerlegung führen. Dabei erhöht sich die Laufzeit mit zunehmender Anzahl an Profilen in $\mathbb{P}$.

Aufgrund der hohen Laufzeit für eine multiple Alignierung analog zur PICA-Methode wird deshalb im Folgenden ein progressiver Ansatz verwendet, welcher in der Bioinformatik in ähnlicher Form für multiple Sequenzalignments eingeführt wurde (vgl. [Feng and Doolittle, 1987], [Otto et al., 2008], [DG et al., 1994]). Dabei

wird das multiple Alignment nicht durch einen Schritt erstellt, sondern sukzessive auf Basis von paarweisen Alignments zusammengesetzt. Ein progressiver Schritt im Alignment entspricht dabei einer paarweisen Alignmentoperation. Für gegebene $k = |\mathbb{P}|$ Profile sind $(k-1) \approx k$ progressive Operationen erforderlich, um die multiple Alignierung durchzuführen. Zusätzlich müssen für jede progressive Operation $O(k)$ Distanzberechnungen mit Hilfe von paarweisen Alignments durchgeführt werden. Die daraus resultierende Laufzeit beträgt $O(k^2 \cdot r^3 \cdot n)$. Damit ist der progressive Ansatz der multiplen Alignierung effizienter in Bezug auf die Laufzeit als eine Vorgehensweise der multiple Alignierung, welche analog zur PICA-Methode durchgeführt wird.

3.4 Multiples intervallbasiertes Kurvenalignment (MICA)

Die MICA-Methode basiert auf einem progressiven Alignmentverfahren. Die Idee dahinter ist, das Alignment nicht als Ganzes zu berechnen, sondern basierend auf paarweisen Alignmentdistanzen über einzelne Schritte das resultierende multiple Alignment zusammenzusetzen. Dabei entspricht ein progressiver Alignmentschritt der paarweisen Alignierung aus Kapitel 3.1 und einer zusätzlichen internen Aktualisierung der resultierenden paarweisen Alignmentdistanzen.

Zu Begin der Alignierung wird jedes Profil einem Cluster $\mathbb{C}_i = \{P_i\}$ zugewiesen. Zu diesem Zeitpunkt hat das Cluster eine Clustergröße $|\mathbb{C}_i| = 1$. Für jeden Cluster $\mathbb{C}$ exisiert ein Konsensus Profil $K_{\mathbb{C}}$, welches die im Cluster enthaltenen Profile repräsentiert und entsprechende paarweise optimale Distanzen zwischen zwei Clustern bestimmt. Ein progressiver Schritt verwendet das Clusterpaar, welches die geringste Distanz aufweist und verzerrt die Profile der jeweiligen Cluster entsprechend dem optimalen Alignment beider Clusterkonsensusprofile. Für den neu entstandenen alignierten Cluster werden im Anschluss wieder Distanzen seines Konsensus zu allen restlichen Clustern bestimmt und das Verfahren wiederholt. Im Laufe der Alignierung werden somit sukzessiv wachsende Cluster gebildet. Das Alignmentverfahren endet, wenn alle Profile in einem Cluster vereint sind.

Die Information der Clustergröße soll im Folgenden in die Distanzberechnung mit einfließen. Dies stellt sicher, dass große Cluster gegenüber kleinen Clustern differenziert bewertet werden. Das Ziel soll sein, dass Profile eines großen Clusters

in folgenden Alignierungsschritten weniger verzerrt werden. Dies erreicht man, indem die Distanz mit steigender Clustergröße erhöht wird. Um dies zu realisieren wird im folgenden Kapitel ein neuer Verzerrungsfaktor für die Distanzberechnung definiert.

3.4.1 Distanzberechnung

Die Distanzen eines paarweisen Alignments werden dabei analog zu Gleichung 3.9 und Gleichung 3.8 berechnet.

Allerdings wird, anders als in Gleichung 3.6, ein anderer Verzerrungsfaktor v verwendet, welcher die Clustergrößen berücksichtigt. Die Clustergröße $|\mathbb{C}|$ gibt an, wie viele Profile in einem Cluster vorhanden sind. Dabei entspricht $|\mathbb{C}|$ der Clustergröße des einen Konsensus $K_{\mathbb{C}}$, auf das die Verzerrung berechnet wird, und $|\mathbb{C}'|$ der Clustergröße des anderen Konsensus $K_{\mathbb{C}'}$ der Distanzberechnung.

$$v(I, \hat{l}, |\mathbb{C}|, |\mathbb{C}'|) = \frac{l}{\hat{l}} \cdot \frac{|\mathbb{C}|}{|\mathbb{C}| + |\mathbb{C}'|} \tag{3.16}$$

Folgendes Beispiel verdeutlicht, warum die Clustergrößen berücksichtigt werden müssen. An einem zufälligen Punkt im progressiven multiplen Alignment sollen zwei Cluster $\mathbb{C}, \mathbb{C}'$ aligniert werden. Die Clustergrößen sind dabei wie folgt: $|\mathbb{C}| = 1, |\mathbb{C}'| = 2$. Durch die Berücksichtigung der Clustergrößen in Gleichung 3.16 erhalten wir für die Distanzberechnung für $K_{\mathbb{C}}$ folgenden Verzerrungsfaktor aus Gleichung 3.17. Analog dazu Gleichung 3.18 für $K_{\mathbb{C}'}$. Es ist zu erkennen, dass ein hoher Clusterwert zu einem größeren Verzerrungsfaktor führt.

$$v(I, \hat{l}, 1, 2) = \frac{l}{\hat{l}} \cdot \frac{1}{3} \tag{3.17}$$

$$v(I, \hat{l}, 2, 1) = \frac{l}{\hat{l}} \cdot \frac{2}{3} \tag{3.18}$$

Betrachtet man zudem die Verwendung des Verzerrungsfaktors in Gleichung 3.8, ist zu sehen, dass bei einem großen Verzerrungsfaktor die Werte des einen Profils mehr in die Distanzberechnung einfließen. In diesem Beispiel bedeutet das, dass eine hohe Clustergröße zu einem höheren Verzerrungsfaktor führt, und dieser zu einer stärkeren Gewichtung des jeweiligen Konsensusprofils. Dies führt dazu, dass weniger Verzerrungen auf diesen Cluster angewandt werden.

3.4.2 Progressive Alignierung

Als Initialisierungsschritt für das progressive multiple Alignment werden zunächst für $P_1, \ldots, P_k$ insgesamt k Cluster $\mathbb{C}_i$ generiert. Im Anschluss wird die Distanzmatrix $D_{i,j}$ mit $1 \leq i, j \leq k$ mit den paarweisen Distanzen der Cluster $\mathbb{C}_i, \mathbb{C}_j$ generiert. Hierbei wird jeder Cluster $\mathbb{C}_i$ durch seinen Konsensus $K_{\mathbb{C}_i}$ repräsentiert. Im Folgenden steht $PICA(\mathbb{C}, \mathbb{C}')$ für das optimale paarweise Alignment der Konsensi von $\mathbb{C}$ und $\mathbb{C}'$. Damit gilt $D_{i,j} = d(PICA(\mathbb{C}_i, \mathbb{C}_j))$. Der kleinste Eintrag $D_{i,j}$ bestimmt dabei das Profil-/Clusterpaar, welches im nächsten progressiven Alignmentschritt zu alignieren ist.

Ein progressiver Alignmentschritt besteht grundlegend aus folgenden Teilschritten:

- Bestimme Alignment $PICA(\mathbb{C}_i, \mathbb{C}_j) = (\mathbb{I}_i, \mathbb{I}_j)$ mit minimaler Distanz in D.
- Verzerre Profile aus $\mathbb{C}_i$ mit $\mathbb{I}_i$ und $\mathbb{C}_j$ mit $\mathbb{I}_j$ und generiere daraus neuen Cluster $\mathbb{C}'$.
- Entferne Cluster $\mathbb{C}_i$ und $\mathbb{C}_j$ aus der Menge der verfügbaren Cluster für die progressive Alignierung. Dazu zählt auch die Entfernung der korrespondierenden Alignmentdistanzen aus der Distanzmatrix.
- Berechne paarweise Distanzen $d(PICA(\mathbb{C}', \mathbb{C}))$ zu allen noch vorhandenen Clustern $\mathbb{C}$ und füge entsprechende Einträge in D ein.

In jedem Schritt, in dem ein paarweises Alignment durchgeführt wird, verringert sich die Anzahl der Cluster für die weitere Alignierung um eins. Dies hat zur Folge, dass die multiple Alignierung genau dann terminiert, wenn keine paarweise Alignierung mehr durchgeführt werden kann und alle Profile in einem Cluster vereinigt wurden. Das verbleibende Konsensusprofil repräsentiert gleichzeitig den Konsensus K des multiplen Alignments.

3.4.3 Dynamischer Ansatz versus statischer Ansatz

In [Bender et al., 2012] wird ein statischer progressiver Ansatz für die multiple Alignierung verwendet. Das bedeutet, dass die Distanzmatrix D zu Beginn mit allen Distanzen der paarweisen Kombinationen aus $\mathbb{P}$ initialisiert wird. Allerdings sind diese Distanzen im Folgenden fixiert. Die Distanzmatrix wird nicht durch

neue Distanzwerte von Konsensusprofilen der Cluster erweitert. Stattdessen wird die Distanz zwischen zwei Clustern durch die minimale Distanz zweier enthaltener Ausgangsprofile definiert. Mit Hilfe dieser fixierten Distanzmatrix ist bereits zu Beginn festgelegt, in welcher Reihenfolge die Profile paarweise aligniert werden. Da bei diesem Ansatz weniger Distanzberechnungen durchgeführt werden müssen, ist der statisch progressive Ansatz schneller in der Durchführung als der hier vorgestellte dynamisch progressive Ansatz. Unter Umständen liefert die statisch progressive Vorgehensweise einen Konsensus, welcher die Charakteristika der Profile in $\mathbb{P}$ schlechter darstellt. Im Gegensatz dazu verspricht man sich vom dynamischen Ansatz eine bessere Gestalterhaltung.

Das Evaluationskapitel am Ende dieser Arbeit wird genau diesen Vergleich zwischen den beiden Ansätzen betrachten.

3.4.4 Beispiel

Im folgenden Beispiel wird aus drei gegebenen Profilen P, P', P'' das Konsensusprofil K als Ergebnis des dynamisch progressiven multiplen Alignments erstellt. Dabei handelt es sich bei P um das gleiche Profil, welches im Beispiel von 3.1 verwendet wurde, mit dazugehöriger Wertetabelle A.3. Auch Profil P' wurde bereits in Wertetabelle A.4 im Beispiel von 3.1 definiert. Zudem wird in diesem Beispiel nun das Profil P'' eingeführt, welches über Wertetabelle A.5 definiert ist. Alle drei Profile sind in Abbildung 3.7 zu sehen.

Der Ansatz der nativen Mittelwertberechnung, ohne Intervallzerlegung und Verzerrung der Profile, bietet in diesem Fall nur ein bedingt brauchbares Ergebnis. Bei der Betrachtung der Charakteristika der drei Ausgangsprofile werden diese nicht optimal im Konsensus dargestellt (vgl. Abbildung 3.8).

Abbildung 3.9 zeigt das Ergebnis der dynamisch progressiven multiplen Alignierung. Dabei wurden die drei Ausgangsprofile entsprechend ihrer Distanzen paarweise aligniert, verzerrt und der entsprechende Konsensus abgeleitet. Das Konsensusprofil gibt dabei die Gestalt aller drei Ausgangsprofile optimal wieder.

Abbildung 3.10 zeigt die initiale Distanzmatrix für dieses Beispiel. Dabei beinhaltet die Distanzmatrix die drei paarweisen Distanzen zwischen den Profilen $(P, P'), (P, P''), (P', P'')$ beziehungsweise die Distanzen zwischen den optimale Zerlegungen $(\mathbb{I}, \mathbb{I}'), (\mathbb{I}, \mathbb{I}''), (\mathbb{I}', \mathbb{I}'')$ der Profile. Die Anzahl der zu Berechnenden Zellen

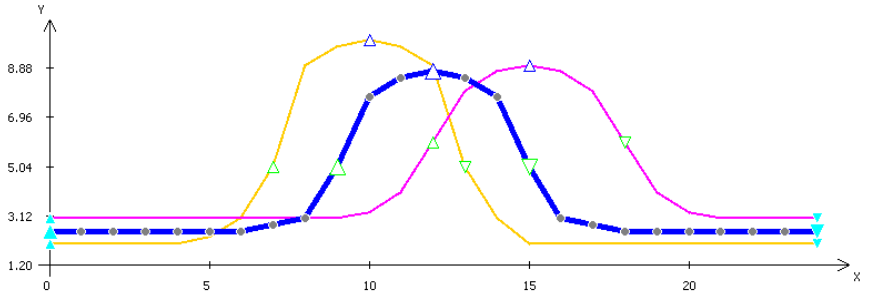

Abbildung 3.7: Profil P (orange) und P' (pink) aus Beispiel von Kapitel 3.1. Zusätzlich Profil P'' (blau) als Erweiterung durch das Beispiel von Kapitel 3.3. Mit neun Referenzpunkten

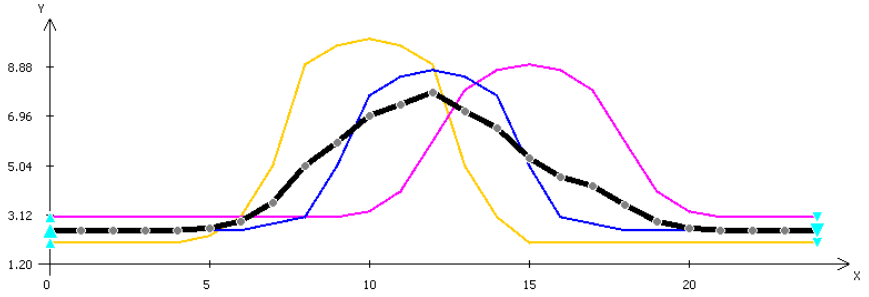

Abbildung 3.8: Konsensus K (schwarz) durch native Mittelwertberechnung als Ergebnis von Abbildung 3.7

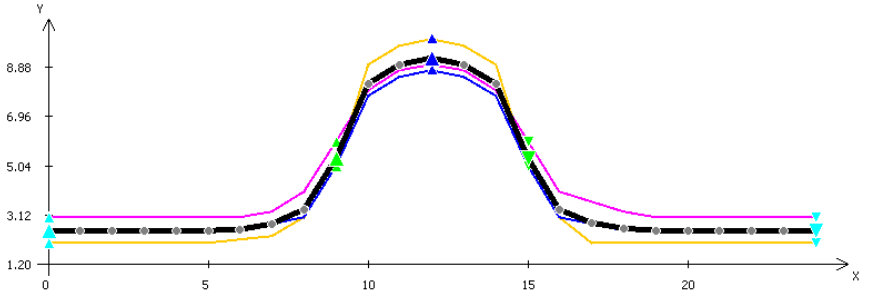

Abbildung 3.9: Korrespondierende verzerrte Profile im Vergleich mit vorheriger Abbildung 3.7 und Konsensus K (schwarz) durch multiple Alignierung von P, P' und P''

der Distanzmatrix lässt sich durch Gleichung 3.19 ermitteln und ist direkt von der Anzahl der zu alignierenden Ausgangsprofile $\mathbb{P} = \{P, P', P''\}$ abhängig.

$$\frac{|\mathbb{P}|^2}{2} - \frac{|\mathbb{P}|}{2} \tag{3.19}$$

Die minimale Distanz findet sich in Zelle (P, P'') mit 0,258. Aufgrund dessen wird die paarweise Alignierung $\mathbb{A} = (P, P'')$ durchgeführt, welche als Ergebnis einen Konsensus $K_{P,P''}$ liefert.

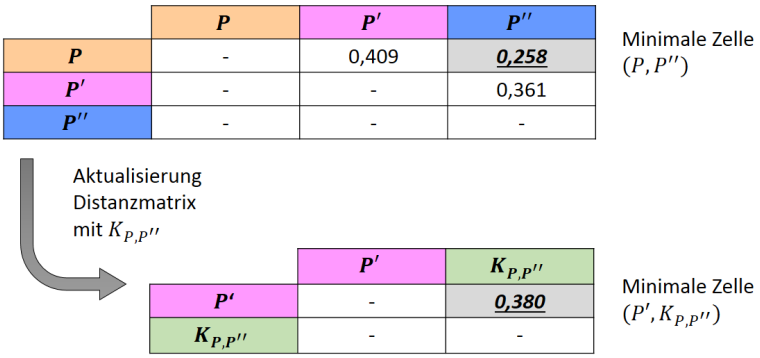

Abbildung 3.10: Distanzmatrix für die drei Ausgangsprofile P, P', P'' und deren Erweiterung beim ersten Alignmentschritt zwischen P, P''. Minimale Distanz jeweils unterstrichen hervorgehoben.

Durch den Konsensus $K_{P,P''}$ der paarweisen Alignierung werden die bei der Alignierung verwendeten Profile P, P'' in der Distanzmatrix ersetzt. Für die beiden verbleibenden Profile $P', K_{P,P''}$ wird die Distanzmatrix, wie in Abbildung 3.10 dargestellt, aktualisiert. Da nur noch zwei Profile für die Alignierung zur Auswahl stehen, werden diese nun im letzten Schritt paarweise aligniert. Das Ergebnis der Alignierung ist der Konsensus K, welcher gleichzeitig das Ergebnis der multiplen Alignierung darstellt.

Der Baum in folgender Abbildung 3.11 beschreibt die Reihenfolge, in der das dynamisch progressive multiple Alignment erstellt wurde. Dabei befinden sich die Ausgangsprofile aus $\mathbb{P}$ in den Blättern des Baumes. Analog zum Beispiel, in dem im ersten Schritt die Profile (P, P'') aligniert wurden, ist in Abbildung 3.11 die Erstellung des Konsensus $K_{P,P''}$ zu sehen. Innere Knoten repräsentieren generierte

Cluster, die während des progressiven Alignments entstehen. Der Wurzelknoten des Baumes repräsentiert den Konsensus K des multiplen Alignments. Sowohl die Erstellung der Distanzmatrix als auch die Erstellung des Baumes findet dynamisch statt.

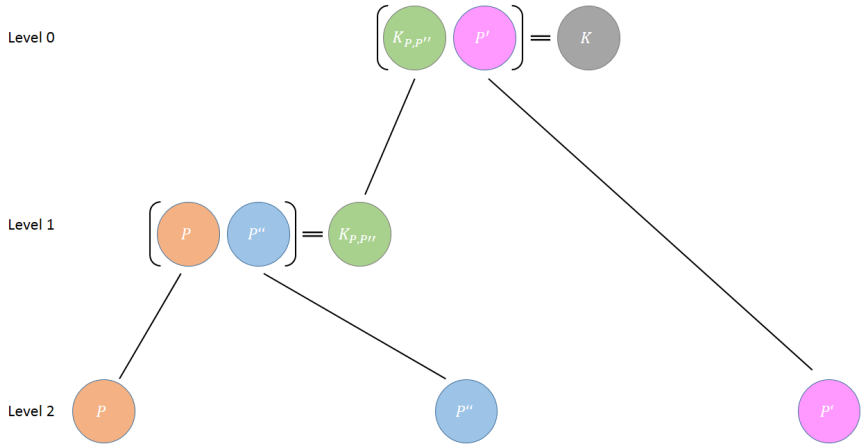

Abbildung 3.11: Alignment-Reihenfolge des dynamisch progressiven multiplen Alignments repräsentiert durch einen Baum. (Zu Beispiel aus Kapitel 3.3)

3.4.5 Laufzeitanalyse MICA

Die MICA-Methode setzt sich aus einer initialen Phase und einer progressiven Phase zusammen. Dabei folgt die progressive Phase direkt nach der initialen Phase. Im Folgenden wird zunächst die Laufzeit der einzelnen Phasen analysiert und abschließend die Gesamtlaufzeit der MICA-Methode benannt.

Am Anfang der MICA-Methode ist es erforderlich die Distanzmatrix D aus den zu alignierenden Profilen in $\mathbb{P}$ initial zu berechnen. Für $k = |\mathbb{P}|$ sind somit $\frac{k^2}{2} \approx k^2$ Zellen zu berechnen. Da die MICA-Methode direkt die paarweise Alignierung der PICA-Methode mit einer Komplexität von $O(r^3 \cdot n)$ verwendet (vgl. Kapitel 3.2.3), ergibt sich eine Laufzeitkomplexität in der initialen Phase von $O(k^2 \cdot r^3 \cdot n)$.

Im Folgenden sind $(k-1) \approx k$ progressive Schritte durchzuführen. Jeder dieser Schritte erfordert $O(k)$ PICA-Berechnungen und hat eine Laufzeitkomplexität von $O(k \cdot r^3 \cdot n)$. Somit ergibt sich für die progressive Phase der MICA-Methode eine Laufzeitkomplexität von $O(k^2 \cdot r^3 \cdot n)$.

Für die MICA-Methode als Ganzes beträgt die Laufzeitkomplexität $O(k^2 \cdot r^3 \cdot n)$, welche sich aus der sequenziellen Ausführung der Initialisierungsphase und der progressiven Phase zusammensetzt.

3.5 Referenzbasiertes multiples intervallbasiertes Kurvenalignment (RMICA)

Bei der RMICA-Methode handelt es sich um eine spezielle Form der MICA-Methode. Als Ausgangssituation existiert eine Menge $\mathbb{P}$ von Profilen, welche aligniert werden sollen. Zudem wird genau ein Profil aus $\mathbb{P}$ als Referenzprofil P_R gekennzeichnet (vgl. Abbildung 3.12). Ziel der RMICA-Methode ist es alle Profile an das Referenzprofil zu alignieren und anschließend ein entsprechendes Konsensusprofil zu bilden.

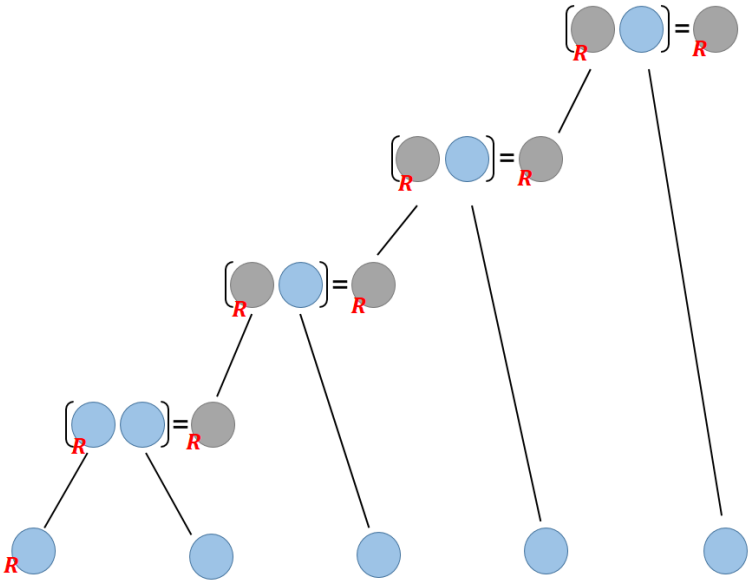

Abbildung 3.12: Alignment-Reihenfolge der RMICA-Methode. Referenzprofil durch **R** (rot) gekennzeichnet. Ausgangsprofile aus $\mathbb{P}$ (blau), Konsensusprofile aus Clustern (grau).

Unter Berücksichtigung dieser Bedingung müssen nur noch Distanzen von allen Profilen aus $\mathbb{P} \setminus \{P_R\}$ gegen das Referenzprofil P_R ermittelt werden. Ein Profil,

das als Referenzprofil gekennzeichnet ist, wird bei der Alignierung nicht verzerrt. Die RMICA-Methode führt dazu paarweise Alignierungsschritte durch, in denen eines der beiden Profile als Referenzprofil gekennzeichnet ist. Das bedeutet, dass immer das nicht Referenzprofil auf das Referenzprofil angepasst wird. Anschließend wird ähnlich zur MICA-Methode progressiv das multiple Alignment zusammengesetzt, jedoch mit der Einschränkung, dass immer nur an ein Profil mit der Referenzkennzeichnung paarweise aligniert werden darf. Bei der Erstellung eines Konsensusprofils $K_{P_R,P}$ aus zwei Profilen wird das Konsensusprofil selbst zu einem Referenzprofil, da es sowohl P_R als auch P ersetzt. Abbildung 3.12 zeigt dabei beispielhaft die Reihenfolge, in der die RMICA-Methode die progressiven Schritte durchführen würde.

Eine Mischform der RMICA-Methode und der MICA-Methode stellt die PRMICA-Methode dar. PRMICA steht für „progressiv referenzbasiertes multiples intervallbasiertes Kurvenalignment“. Bei der PRMICA-Methode kann mehr als ein Referenzprofil definiert werden. Folgende drei Fälle können bei der paarweisen Alignierung zweier Profile P, P' auftreten:

- P ist kein Referenzprofil und P' ist kein Referenzprofil, dann wie MICA.
- P ist Referenzprofil und P' ist Referenzprofil, dann wie MICA.
- P ist Referenzprofil und P' ist kein Referenzprofil, dann wie RMICA.

Dies bedeutet, wurden zwei Profile auf der Grundlage der minimalen Alignierungsdistanz bestimmt und keines der beiden Profile ist als Referenzprofil gekennzeichnet, so wird identisch zur MICA-Methode verfahren. Sind beide Profile als Referenzprofile gekennzeichnet wird auch identisch zur MICA-Methode vorgegangen. Ist nur eines der beiden Profile als Referenzprofil gekennzeichnet, so wird analog zur RMICA-Methode vorgegangen und das Referenzprofil nicht verzerrt.

Abbildung 3.13 zeigt beispielhaft die Reihenfolge, in der die PRMICA-Methode die progressiven Alignierungsschritte durchführen würde. Gut zu sehen ist, dass auf der rechten Seite zunächst nach der MICA-Methode verfahren wurde. Auf der linken Seite wurde die RMICA-Methode angewandt.

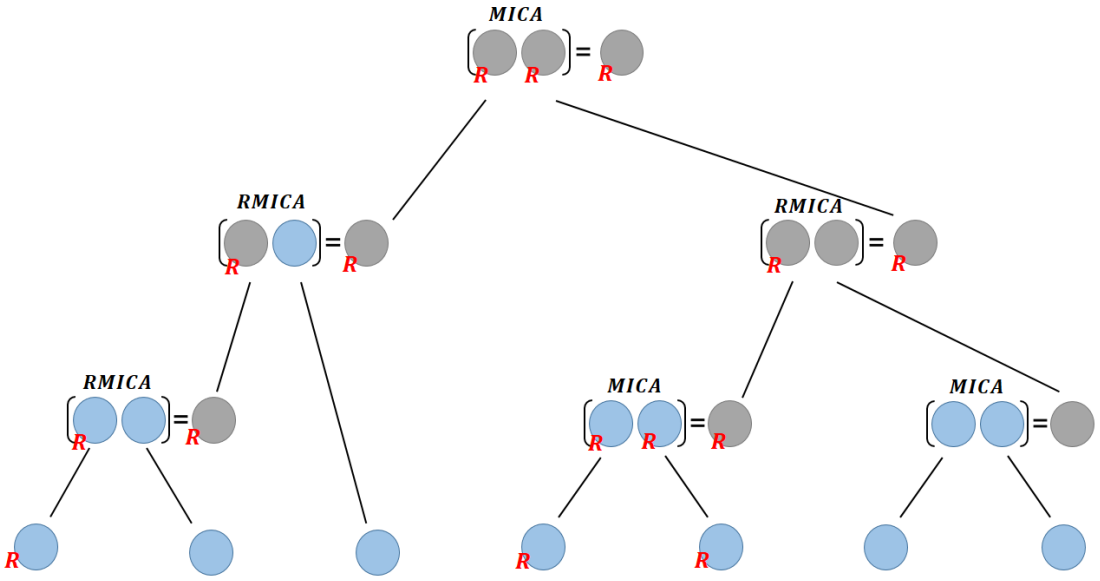

Abbildung 3.13: Alignment-Reihenfolge der PRMICA-Methode. Kennzeichnung analog zu Abbildung 3.12. Kennzeichnung der verwendeten Alignierungsmethode (MICA, RMICA) an den inneren Knoten.

3.6 Splitbasiertes multiples intervallbasiertes Kurvenalignment (SMICA)

Bei der SMICA-Methode gibt es zu den Ausgangsprofilen $\mathbb{P} = \{P_1, \ldots, P_k\}$ für die Alignierung zusätzlich eine Splitinformation $\mathbb{S}$. $\mathbb{S}$ defniert hierbei eine Vorzerlegung der Profile in $\mathbb{P}$, indem für jedes Profil zu alignierende Splitpunkte angegeben werden. Solch eine Zusatzinformation kann zum Beispiel das Wissen über den Zusammenhang bestimmter Punkte in den Daten sein.

Die Splitinformation $\mathbb{S}$ beinhaltet $|\mathbb{P}|$-Tupel an Splitpunkten. Dabei bezieht sich der erste Wert des Tupels auf das erste Profil in $\mathbb{P}$ und der $|\mathbb{P}|$-te Wert des Tupels auf das k-te Profil in $\mathbb{P}$. Zudem gilt, dass sich die Punkte eines Splittupels nicht mit den Punkten eines anderen Splittupels überkreuzen dürfen. Für eine beispielhaft gegebene Splitinformation $\mathbb{S} = \{(a, b), (c, d), (e, f)\}$ mit $a, b, c, d, e, f \in \mathbb{X}$ für $\mathbb{P} = \{P_1, P_2\}$ muss gelten: $1 \leq a < c < e \leq |P_1|$ und $1 \leq b < d < f \leq |P_2|$.

Folgende Abbildung 3.14 verdeutlicht den Ablauf der SMICA-Methode für zwei gegebene Profile P, P' mit Splitinformation $\mathbb{S}$.

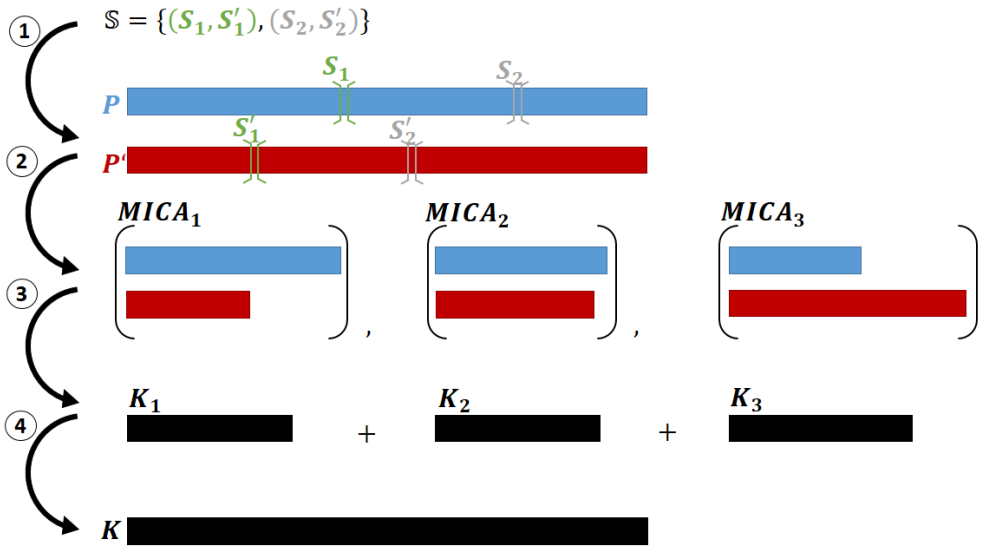

Abbildung 3.14: SMICA: (1) Anwendung der Splitinformation $\mathbb{S}$ auf die Profile aus $\mathbb{P}$. (2) Zerlegung der Profile durch $\mathbb{S}$ in unabhängige Teilprobleme. (3) Getrennte MICA-Berechnung der einzelnen unabhängigen Teilprobleme liefert Teilergebnisse. (4) Zusammenführung der Teilergebnisse zum SMICA-Ergebnis.

Im ersten Schritt (vgl. Abbildung 3.14 (1)) wird die Splitinformation $\mathbb{S}$ auf die Profile angewandt. Anschließend liegen $|\mathbb{S}|+1$ unabhängige Teilprobleme vor (vgl. Abbildung 3.14 (2)). Diese können von nun an getrennt behandelt werden. Dazu wird für jedes dieser Teilprobleme jeweils unabhängig die MICA-Methode angewandt. Als Resultat der MICA-Berechnungen liegen entsprechend viele Alignments mit Konsensus K_i vor, als dass es Teilprobleme gibt (vgl. Abbildung 3.14 (3)). Im letzten Schritt (vgl. Abbildung 3.14 (4)) werden die Teilergebnisse zusammengeführt und das Ergebnis der SMICA-Methode erstellt. Dabei werden die Ergebnisse konkateniert.

Der Vorteil der SMICA-Methode liegt darin, bereits vorhandenes Wissen mit in den Alignierungsvorgang einzubringen. Daraus ergeben sich unabhängige Probleme, die getrennt voneinander berechnet werden können.

Kapitel 4

Implementierung

Dieses Kapitel legt den Fokus auf die Implementierung der MICA-Anwendung. Zunächst wird der Aufbau im Allgemeinen beschrieben und anschließend näher auf die Implementierung der Methoden eingegangen. Zum Schluss dieses Kapitels wird auf die Benutzerschnittstelle im Detail diskutiert. Dabei wird zunächst auf die grafische Benutzeroberfläche, aber auch auf die konsolenbasierten Verwendung der Anwendung.

4.1 Aufbau

Für die Implementierung einer MICA-Anwendung wurde das MVC-Muster (englisch für Model-View-Controller) angewandt (vgl. [Gamma et al., 1994]). Dies stellt sicher, dass die Anwendungskomponenten zur Darstellung, Steuerung und Berechnungsdurchführung voneinander getrennt sind. Dies ermöglicht den einfachen Austausch einzelner Komponenten. Dadurch kann ohne großen Aufwand eine neue Benutzeroberfläche integriert werden, ohne die restliche Funktionalität des Systems anpassen zu müssen. Abbildung 4.1 zeigt dabei das MVC-Muster mit einigen wichtigen Klassen.

Der grobe Ablauf sieht für die Klassen in Abbildung 4.1 wie folgt aus. Bei Programmstart erzeugt die **MicaMain**-Klasse den **Controller**. Dieser wiederum erzeugt ein **Model**-Objekt und das **ViewImportFilterExecute**-Objekt. **ViewImportFilterExecute** ist die Hauptklasse der grafischen Benutzerschnittstelle. Sie übernimmt die Darstellung der Profile über die **ColorProfileDataPlot**-Klasse.

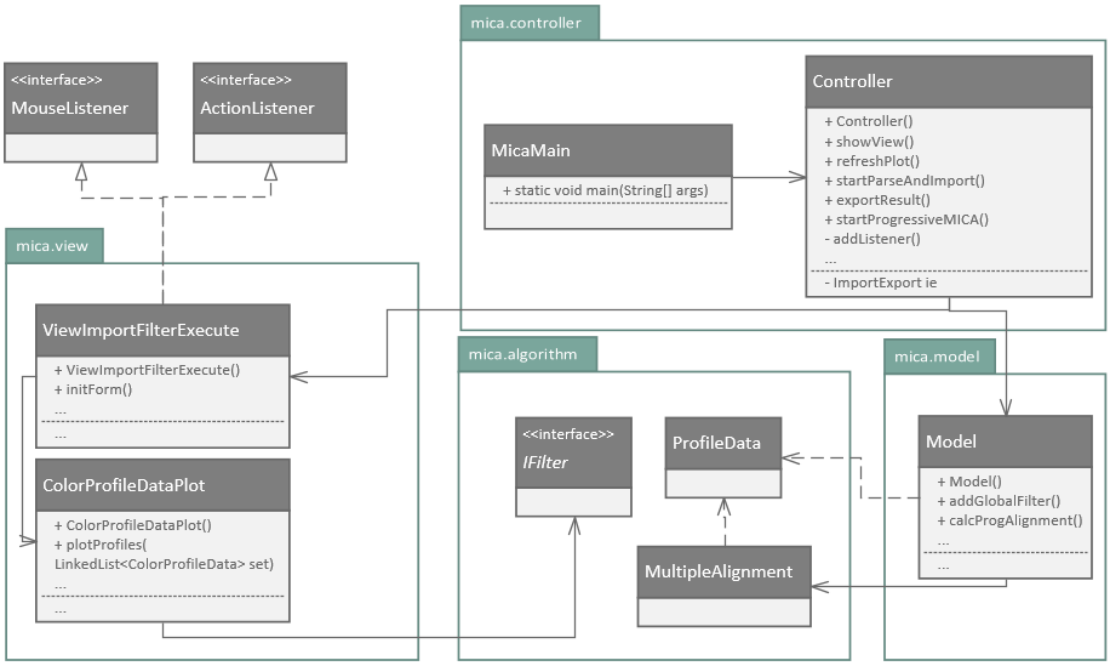

Abbildung 4.1: Klassendiagramm Model-View-Controller-Pattern

Zudem bietet sie dem Benutzer die Möglichkeit mit dem System zu interagieren. Das **Model** kapselt die eigentliche Anwendung zur Berechnung des multiplen Alignments. Über den Controller werden die Ergebnisse vom Model in die View Klassen geladen und schließlich angezeigt. Dabei dienen die View-Klassen zur Darstellung der Ergebnisse. Umgekehrt werden Benutzereingaben in einer View durch den Controller an das Model weitergereicht.

Das Importieren von Profilen wird über die **ImportExport**-Klasse durchgeführt. Dabei wird die Klasse **DataPointDerivator** verwendet, um die Anzahl der Profildatenpunkte anzupassen. Für die Dateiformatunterstützung wurde ein Strategiemuster (vgl. [Gamma et al., 1994]) verwendet, welches zusätzliche Dateiformate erlaubt. In der gegenwärtigen Implementierung ist nur ein Dateiformat implementiert. Dabei handelt es sich um das CSV (comma separated values) Dateiformat. Abbildung A.1 zeigt sowohl das Strategiemuster für das Dateiformat als auch die Importierungsklasse. Über die **ImportExport**-Klasse kann zudem das Exportieren von Profilen durchgeführt werden.

Abbildung A.2 und Abbildung A.3 zeigen weitere Anwendungsfälle von Strategiemustern in der Implementierung. In Abbildung A.2 kann für die Instanziierung von **RMSDDistanceSlope** eine Distanzberechnung verwendet werden, welche auf den Steigungswerten basiert. Im anderen Fall wird eine Distanzberechnung

basierend auf den Datenwerten instanziiert (**RMSDDistanceData**). Abbildung A.3 beschreibt die Möglichkeit einen Filter des Typs Wendepunktfilter oder einen Filter des Typs Extrempunktfilter zu instanziieren.

Ein Profil repräsentiert im System einen Datensatz. Die **Profil**-Klasse beinhaltet dabei den Namen, die Datenpunkte und die daraus berechneten Steigungswerte. Abbildung 4.2 zeigt, wie die **Profil**-Klasse durch die **ProfileData**-Klasse erweitert wird. Durch die Erweiterung stehen dem System Referenzpunkte zur Verfügung. Zudem können dadurch Filter für einzelne Profile registriert werden, um Referenzpunkte einzuschränken. Die nächste Erweiterung stellt die Klasse **ColorProfileData** dar, die ausschließlich in der View für Darstellungszwecke benötigt wird. Das Profil wird hier durch einen Farbwert erweitert.

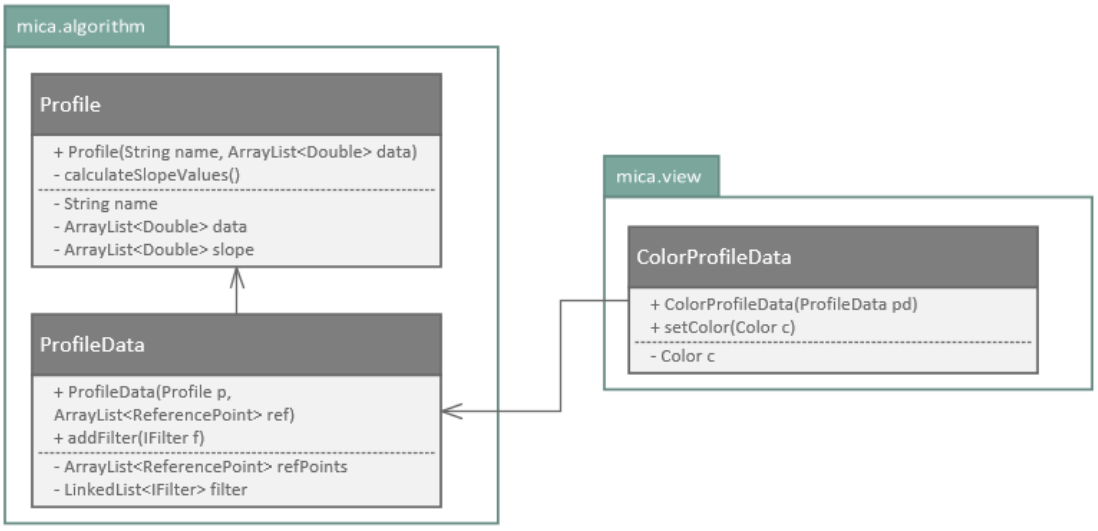

Abbildung 4.2: Klassendiagramm zur Profile-Klasse und deren Erweiterung ProfileData. Zudem ProfileColorData als Erweiterung für die Darstellung in der View.

4.2 Methoden

In diesem Abschnitt wird näher auf die Implementierung der Alignmentmethoden eingegangen: Zunächst auf die paarweise Alignierung, anschließend auf die multiple Alignierung.

Abbildung 4.3 zeigt den Zusammenhang der wichtigsten Klassen für die paarweise Alignierung. Im Zentrum des Klassendiagramms steht die **PairwiseAlignment**-Klasse. Sie repräsentiert die PICA-Methode. Es ist zu sehen, dass die **PairwiseAlignment**-Klasse nur über die **MultipleAlignment**-Klasse erreichbar ist. Die paarweise Alignierung kann als Spezialfall der multiplen Alignierung gesehen werden. Auf der Grundlage der **ProfileData**-Klasse und deren **ReferencePoint**-Klasse kann die PICA-Methode eine Intervallzerlegung bestimmen. Die Intervallzerlegung entspricht einer Sequenz aus **IntervalPair**-Instanzen. Auf dieser Grundlage erstellt die **PairwiseAlignment**-Instanz das Konsensusprofil.

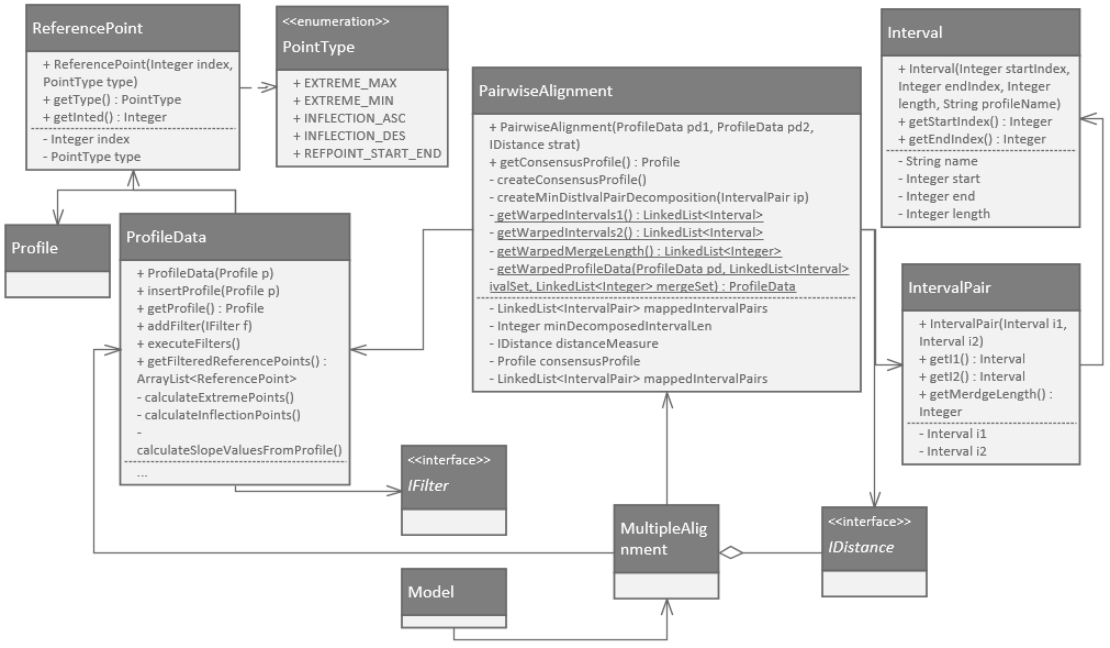

Abbildung 4.3: Klassendiagramm paarweise Alignierung PICA mit in Relation stehenden Klassen.

Abbildung 4.4 zeigt den Zusammenhang der wichtigsten Klassen für die multiple Alignierung. Dabei wird das progressive multiple Alignment schrittweise durch paarweise Alignierungen zusammengesetzt. Die Distanzwerte der multiplen Alignierung werden in der Klasse **DistanceMatrix** verwaltet. Die **TreeNode**-Klasse kann zu einem Baum zusammengesetzt werden und bildet so den Verlauf der progressiven Alignierung ab. Zudem beinhaltet diese Klasse die Information über das angewandte Alignment und die Konsensusresultate der einzelnen Alignmentschritte. Über die **MultipleAlignment**-Klasse wird das Alignment für eine gegebene Menge an Profilen durchgeführt.

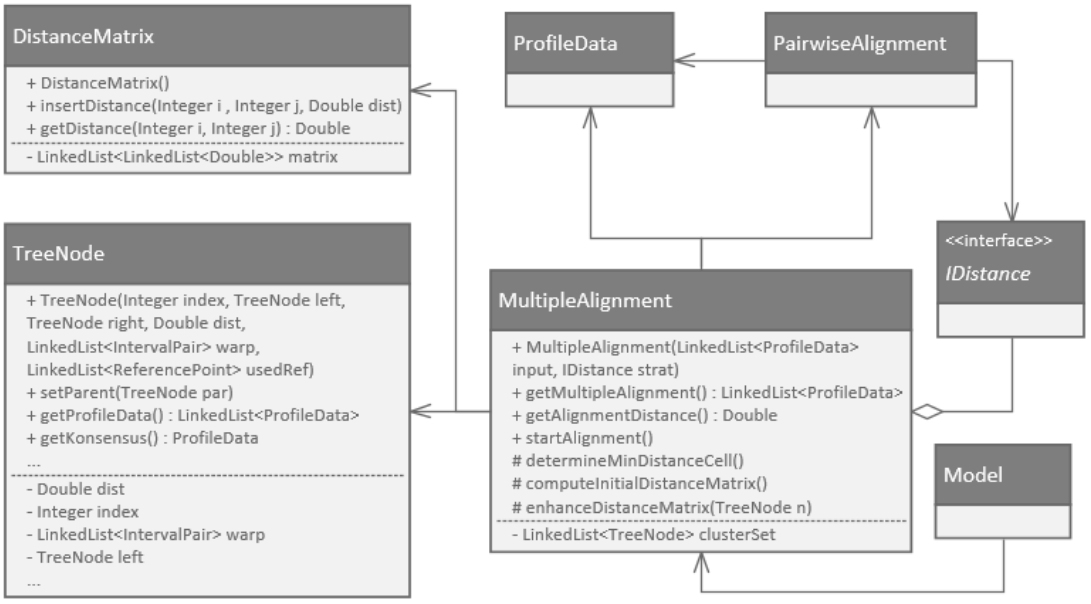

Abbildung 4.4: Klassendiagramm multiple Alignierung MICA mit in Relation stehenden Klassen.

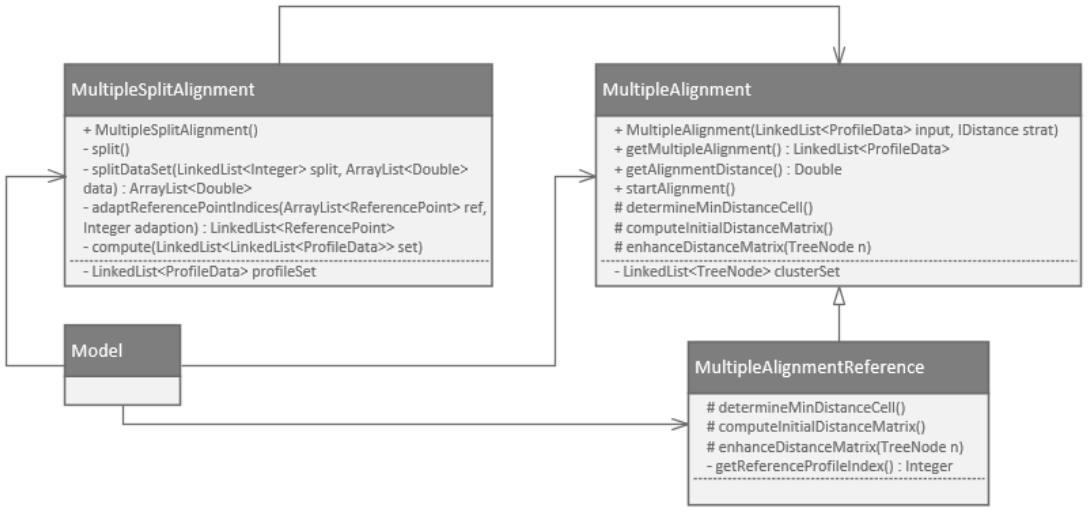

Abbildung 4.5: Klassendiagramm referenzbasierte multiple Alignierung RMICA und splitbasierte multiple Alignierung SMICA

Abbildung 4.5 zeigt den Zusammenhang der Klassen für die verschiedenen Alignierungsverfahren (RMICA/PRMICA, SMICA). Dabei ist ersichtlich, dass durch die **Model**-Klasse gesteuert wird, welche Alignierungsart verwendet wird. Die zentrale Klasse für die Alignierung ist die **MultipleAlignment**-Klasse. Von ihr erbt die Klasse **MultipleAlignmentReference**, welche für die referenzbasierte multiple Alignierung zuständig ist. Dabei werden drei Funktionen der Oberklasse überschrieben, welche im referenzbasierten Fall differenziert behandelt werden. Dazu zählt die Auswahl der minimalen Distanz, die Berechnung der initialen Distanzmatrix und die Erweiterung der Distanzmatrix. Die Klasse **MultipleAlignmentReference** repräsentiert dabei das RMICA als auch das PRMICA-Verfahren. Die Klasse **MultipleSplitAlignment** repräsentiert das SMICA-Verfahren für die Alignierung. Durch eine Vorverarbeitung werden die Daten entsprechend der Zerlegungsinformation aufbereitet. Anschließend wird direkt das **MultipleAlignment**-Objekt für die Alignierung der einzelnen Teilprobleme verwendet.

4.3 Parallelisierung

Die Berechnung der Distanzmatrix bietet Raum für Parallelisierung. Hierbei wird zwischen zwei Fällen unterschieden.

- Initiale Berechnung
- Erweiterung der Distanzmatrix

Bei der initialen Berechnung ist es erforderlich $\frac{k \cdot k-1}{2}$ paarweise Alignments für die entsprechenden Distanzeinträge in der Distanzmatrix zu berechnen. Dabei ist k die Anzahl der zu alignierenden Profile. Da diese Berechnungen unabhängig voneinander durchgeführt werden können, ist es sinnvoll an dieser Stelle zu Parallelisieren.

Die Umsetzung sieht dabei vor, nicht mehr parallele Berechnungen anzustoßen, als Rechenkerne auf dem System vorhanden sind. Dabei wird eine Teilberechnung in einem leichtgewichtigen Prozess berechnet. Hierbei spricht man auch von einem Thread. Durch die Erzeugung von mehreren, parallel prozessierten Threads für die aufteilbare Gesamtberechnung wird die Parallelisierung erreicht.

Folgende Codezeile zeigt, wie die Anzahl der Rechenkerne des zugrundeliegenden Systems ermittelt wird.

```
int numCores = Runtime.getRuntime().availableProcessors();
```

Listing 4.1: Bestimmung der Anzahl der Rechenkerne

Mit Hilfe der Anzahl an Rechenkernen (siehe Listing 4.1) lässt sich ermitteln, wie viele Elemente der Matrix ein Thread berechnen muss. Da die Erzeugung eines Threads Ressourcen und Berechnungszeit auf dem System in Anspruch nimmt, ist die Erzeugung eines Threads vor allem dann sinnvoll, wenn dieser Thread eine Berechnung durchführt, die dazu im Verhältnis steht. Sind die durchzuführenden Berechnungen sehr klein, lohnt es sich nicht Threads zu erzeugen. Die sequentielle Berechnung wäre im Vergleich schneller. Deshalb wird in der Implementierung überprüft, ob die Berechnung die erforderliche Größe hat. Zunächst muss festgelegt sein, wie viele Threads auf einem Rechenkern ausgeführt werden sollen. Anschließend wird bestimmt, wie viele Zellen der Distanzmatrix berechnet werden müssen. Mit Gleichung 4.1 wird bestimmt wie viele Zellen ein jeder Thread zu berechnen hat. Liegt dieser Wert unterhalb einem definierten Grenzwert, dann findet keine Parallelisierung statt. Im anderen Fall wird die Berechnung parallelisiert. In der vorliegenden Implementierung wurde dieser Wert auf Basis von Testläufen auf 3 festgelegt.

$$CellsPerThread = \frac{CellsToCompute}{ThreadsPerCore} \tag{4.1}$$

Bei der aktualisierung der Distanzmatrix werden die Distanzen eines neuen Konsensusprofils zu allen verbleibenden Profilen neu berechnet. Je weiter der Algorithmus fortgeschritten ist, desto weniger paarweise Alignments gilt es zu berechnen. Dadurch werden die Berechnungen an sich sehr viel kleiner als zu Beginn. Anders als bei der initialen Berechnung sind hier im Durchschnitt deutlich weniger Zellen zu berechnen. Wird der Grenzwert für die Erzeugung der Threads durch Gleichung 4.1 unterschritten, werden keine Threads mehr erzeugt, sondern die Berechnung mit einem einzelnen Thread durchgeführt. Die Parallelisierung kommt an dieser Stelle hauptsächlich zu Beginn des Algorithmus zum Tragen.

4.4 GUI

In diesem Kapitel wird näher auf die wichtigsten grafischen Benutzeroberflächen der MICA-Anwendung eingegangen, welche zur Durchführung eines Alignments genutzt werden können.

4.4.1 Hauptfenster

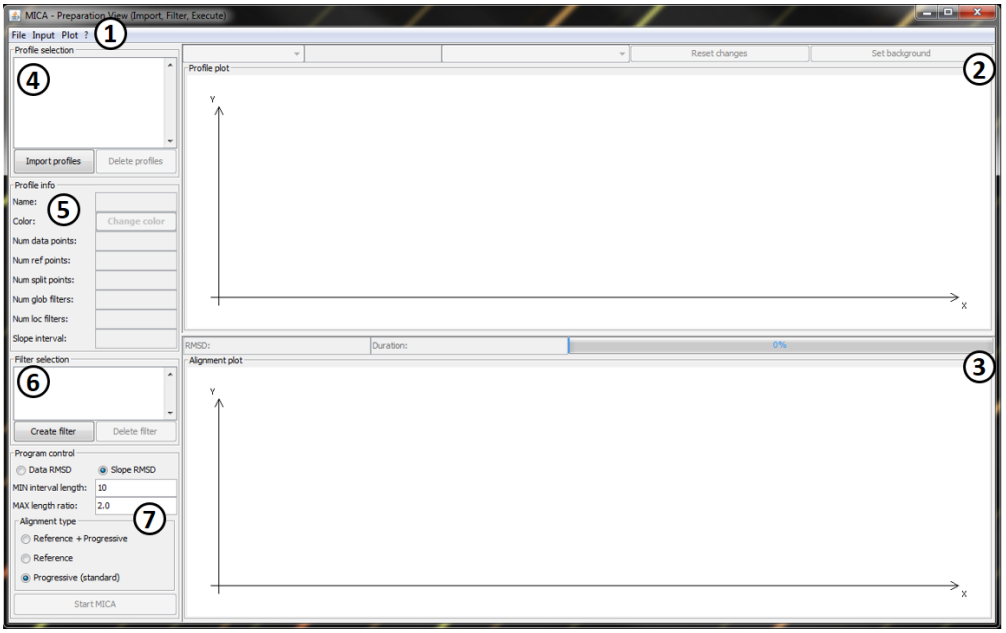

Abbildung 4.6: Hauptfenster der MICA-Anwendung. (1) Menüleiste, (2) Visualisierung und Bearbeitung des Inputs, (3) Visualisierung des Ergebnisses, (4) Verwaltung importierter Profile, (5) Bearbeitung von Profileigenschaften, (6) Verwaltung angelegter Filter, (7) Parametrisierung für die Durchführung des Alignments.

Abbildung 4.6 zeigt das Hauptfenster der MICA-Anwendung, wie es nach einem Programmstart erscheint. Nummerierung (1) verweist auf die Menüleiste. Dort befindet sich die Möglichkeit Ergebnisse zu exportieren, den aktuellen Stand der MICA-Anwendung zu sichern oder eine Konfigurationsdatei für die kommandozeilenbasierte Ausführung zu erstellen (siehe Kapitel 4.5). Zudem kann über die Menüleiste die Initialisierung der SMICA-Methode durchgeführt werden. Abbildung 4.6 zeigt die Visualisierungen der Eingabeprofile (2) und der alignierten Profile inklusive des Konsensusprofiles (3). Auf der linken Seite des Hauptfensters ist Bereich (4) für die Auflistung und Selektion der importierten Profile zuständig. In Bereich (5) können Eigenschaften eines Profils manipuliert werden. Dazu zählt die Farbeinstellung für die Visualisierung, der Name des Profils, aber auch die Größe des gleitenden Fensters für die Steigungswerteberechnung. Zudem sind dort Informationen zu sehen, welche die Anzahl der Datenpunkte oder Referenzpunkte eines Profils wiedergeben. Die Einstellungsmöglichkeiten in (5) beziehen sich immer direkt auf die in (4) selektieren Profile. Bereich (6) zeigt die aktiven Filter, welche auf den Input angewandt werden, an. Hier lässt sich über die Auswahl eines Filters,

durch Doppelklick, der Wert anpassen. In (7) befinden sich Einstellungsmöglichkeiten, die für die Durchführung des Alignments relevant sind. Dazu zählen die Festlegung des gewählten Distanzverfahrens (datenbasiert, steigungsbasiert), die Definition der minimalen Intervallzerlegungslänge sowie die Spezifizierung der Alignierungsstrategie (MICA, RMICA, PRMICA).

4.4.2 Importierung von Profilen

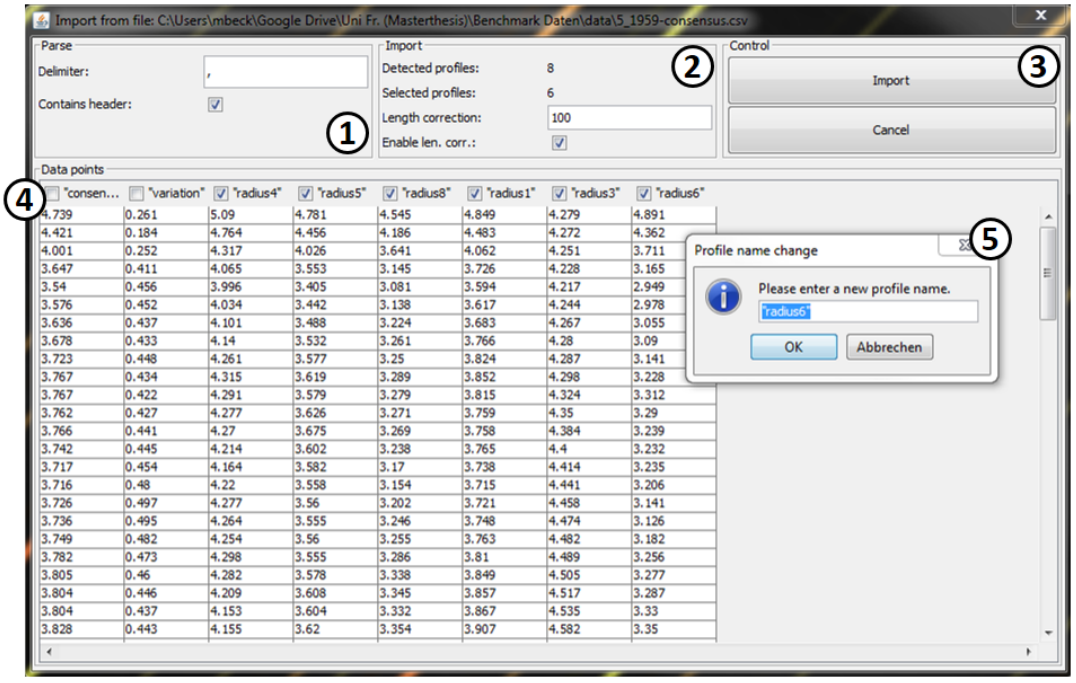

Abbildung 4.7: Dialog zur Importierung von Profilen. (1) Einstellung für das Parsen, (2) Importierungseinstellung, (3) Durchführung der Importierung, (4) Visualisierung der zu importierenden Datei, (5) Änderung vom Profilnamen.

Abbildung 4.7 zeigt das Fenster, welches während der Importierung von Profilen den Vorgang unterstützt. Zunächst befinden sich bei (1) die Einstellungsmöglichkeiten, welchen den Parsevorgang der Datei betreffen. Dazu zählt die Definition des Trennzeichens, welches in der Datei Anwendung findet. Zudem kann hier festgelegt werden, ob die Datei eine Kopfzeile beinhaltet. Sind alle Angaben verwertbar, wird automatisch der Inhalt der Datei bei (4) dargestellt. In Bereich (2) befindet sich die Information über die gegenwärtig ausgewählten Profile für diesen Importierungsschritt. Zudem kann hier angegeben werden, auf welche Länge alle Profile normalisiert werden sollen. Dabei werden Punkte interpoliert oder verworfen um

die gewünschte Länge zu erreichen. Durch deaktivieren der Längenkorrektur werden die Profile mit den Längen, wie sie in der Eingabedatei vorhanden sind, in das System importiert. In der Kopfzeile von Bereich (4) ist zudem eine Auswahl möglich, um nur bestimmte Profile zu importieren. Durch Rechtsklick auf einen Profilnamen kann dieser geändert werden. Der Dialog zur Änderung des Profilnamens zeigt (5). In (3) können alle Einstellungen, welche den Importierungsvorgang betreffen, angewandt werden. Im Folgenden sind die Profile im Hauptfenster zu sehen.

4.4.3 Visualisierung der Inputprofile

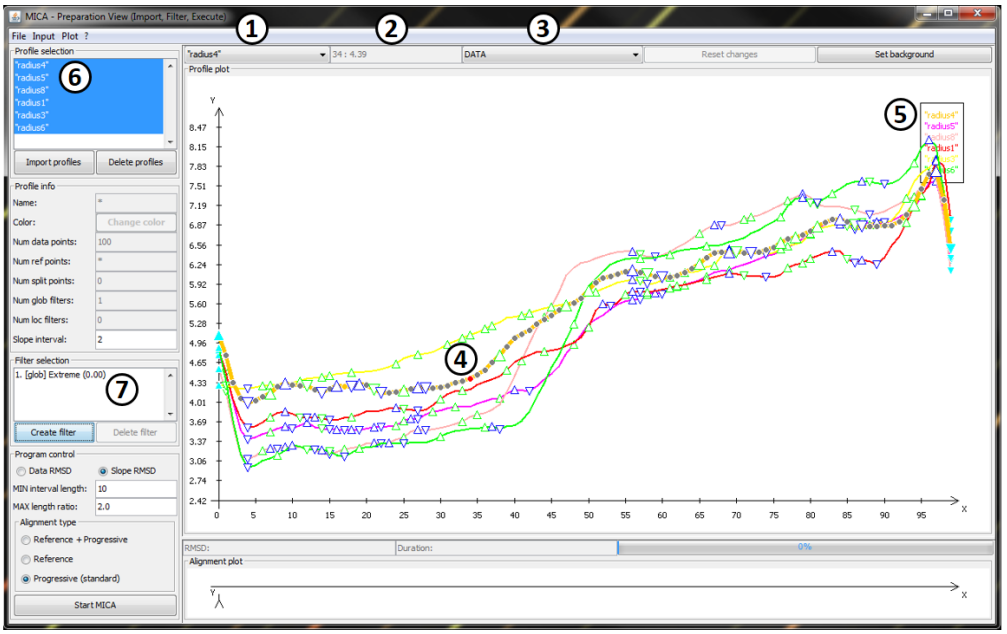

Abbildung 4.8: Visualisierung der importierten Profile. (1) Auswahl des im Vordergrund stehenden Profils, (2) Eigenschaften des ausgewählten Punktes im Profil (siehe (4)), (5) Legende der dargestellten Profile, (6) Ausgewählte Profile für die Visualisierung, (7) Eintrag eines Filters.

Abbildung 4.8 zeigt die Visualisierung von importierten Profilen. Dabei bietet das Dropdownmenü an Stelle (1) die Möglichkeit ein bestimmtes Profil in den Vordergrund der Visualisierung zu bringen. Bereich (2) zeigt dabei die Information über einen ausgewählten Punkt des im Vordergrund befindlichen Profils dar. Der dazu korrespondierende Punkt ist durch (4) in der Visualisierung gekennzeichnet. Durch das Dropdownmenü bei (3) wird der Typ des ausgewählten Punktes angezeigt. In diesem Fall handelt es sich um einen Datenpunkt. An dieser Stelle kann der Punkt

manuell auf einen beliebigen Referenzpunkt (außer START, END) geändert werden. Bereich (5) der Visualisierung zeigt eine Legende der in der Visualisierung befindlichen Profile. In (6) besteht die Möglichkeit, durch gezielte Auswahl, nur einige wenige Profile zu visualisieren. In Bereich (7) ist der Eintrag eines Filters zu sehen, welcher aktuell auf den Input angewandt wird.

4.4.4 Filtererstellung und -manipulation

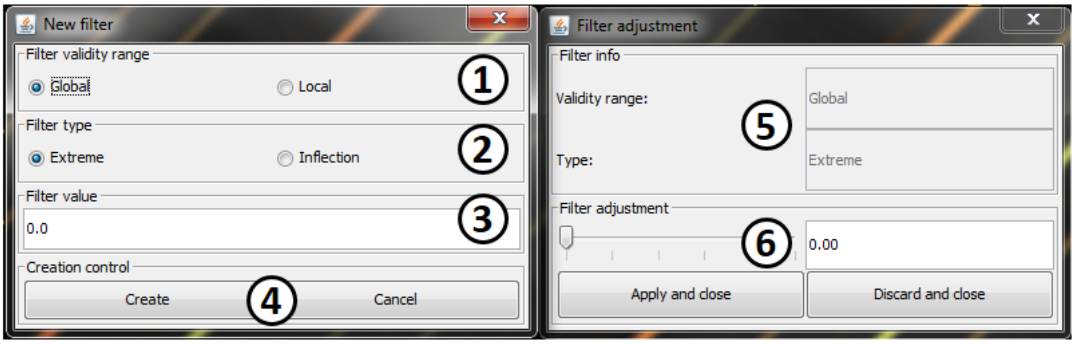

Abbildung 4.9: Dialog zur Filtererstellung und -manipulation. (1) Gültigkeit des Filters, (2) Art des Filters, (3) Wert des Filters, (4) Erstellung des Filters, (5) Eigenschaften eines erstellten Filters, (6) Anpassung des Filterwerts.

Abbildung 4.9 zeigt den Dialog, welcher beim Erstellen eines Filters zu sehen ist, und den Dialog, welcher die Anpassung eines Filters zulässt. In (1) wird der Gültigkeitsbereich des Filters definiert. „Global“ bedeutet, dass der Filter auf alle importierten Profile angewendet wird. „Lokal“ bedeutet, dass nur die gegenwärtig ausgewählten Profile diesen Filter erhalten. Bei (2) wird festgelegt, um welche Art von Filter es sich handelt (Extrempunkt, Wendepunkt). In Bereich (3) lässt sich zum Zeitpunkt des Erstellens des Filters ein entsprechender Wert festlegen. In (4) wird der Filter erstellt oder die gegenwärtigen Einstellungen verworfen. Durch Doppelklick auf einen Eintrag in der Filterliste (vgl. Abbildung 4.8 (7)) öffnet sich der Dialog in Abbildung 4.9 auf der rechten Seite. Bereich (5) gibt hier Auskunft über die Filtereigenschaften. In (6) kann der Filterwert angepasst werden. Dabei erfolgt direkt eine Aktualisierung der Visualisierung in Abbildung 4.8 (4).

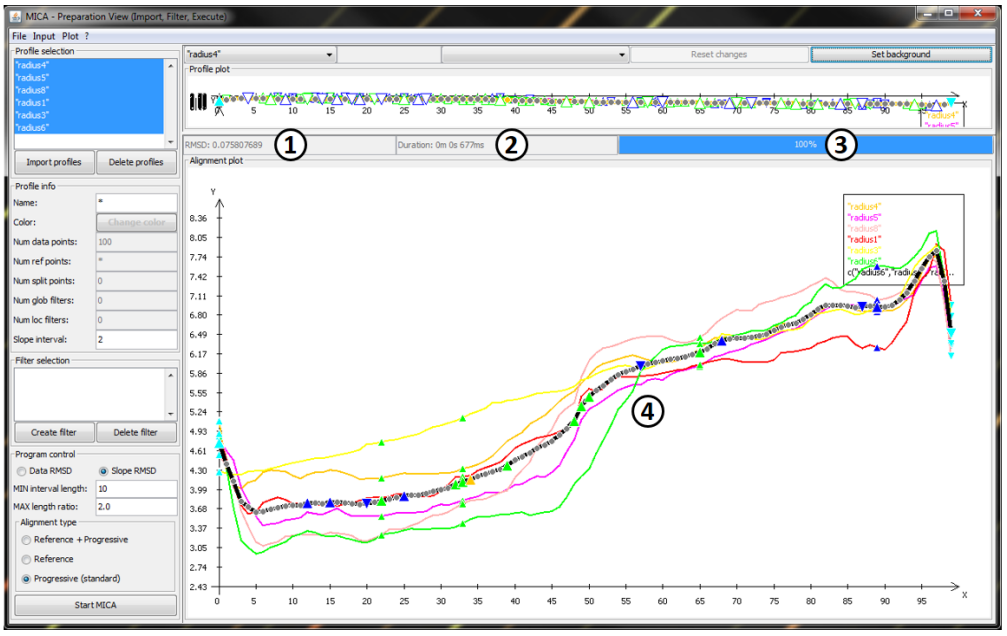

Abbildung 4.10: Visualisierung der verzerrten Profile und Konsensus als Ergebnis des Alignments. (1) Distanzwert des Alignments, (2) Benötigte Zeit zur Erstellung des Alignments, (3) Fortschrittsanzeige der Alignierungsdurchführung.

4.4.5 Visualisierung des Ergebnisses

Abbildung 4.10 zeigt die Visualisierung des Ergebnisses der Alignierung. Dabei ist (1) die Distanz (RMSD) des angezeigten Alignments und (2) gibt die für die Alignierung benötigte Zeit an. Bereich (3) stellt den Fortschritt der Alignierung dar. In (4) sind die verzerrten/alignierten Inputprofile zu sehen und schwarz im Vordergrund befindet sich das Konsensusprofil des Alignments. Sobald ein Ergebnis der Alignierung vorliegt, wird Bereich (4) automatisch maximiert. Bei einer Änderung an den Inputeinstellungen tritt die Visualisierung des Ergebnisses in den Hintergrund und die Inputvisualisierung wird größer dargestellt. Der Benutzer hat zudem die Möglichkeit die Trennlinie zwischen beiden Visualisierungen in die gewünschte Position zu ziehen.

4.4.6 Menüleiste

Abbildung 4.11 zeigt die verschiedenen Menüleisten. Auf der linken Seite befindet sich das Menü zu „File“. Bei Bereich (1) hat der Benutzer die Möglichkeit den aktuellen Stand der MICA-Anwendung zu sichern und später zu laden. Hierbei werden

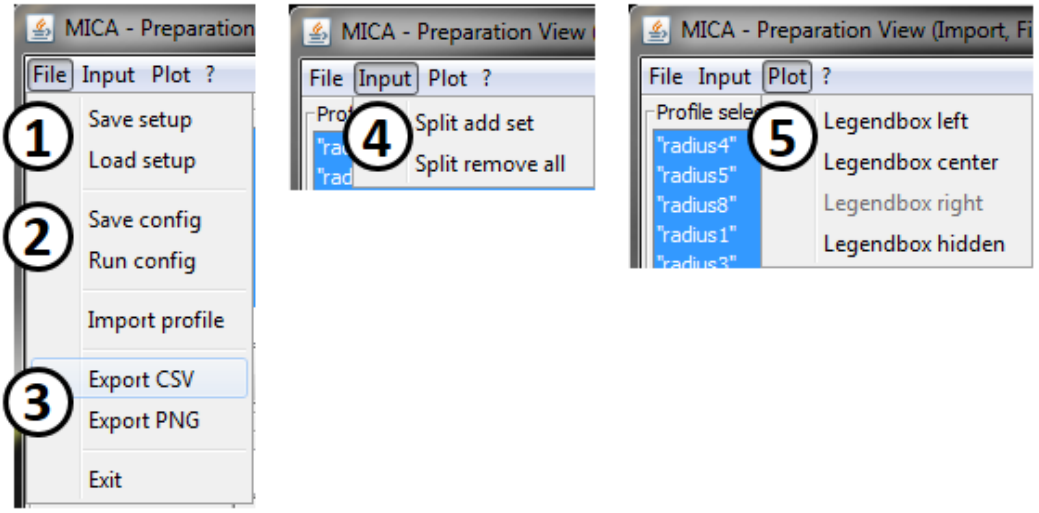

Abbildung 4.11: Menüleiste. (1) Speichern oder Laden des aktuellen Zustands der MICA-Anwendung, (2) Erstellen oder Ausführen einer kommandozeilenbasierten Konfigurationsdatei, (3) Exportieren von Profildaten oder Visualisierungen, (4) Einfügen einer Vorzerlegung der Profile für das Alignment, (5) Positionierung der Legende in der Visualisierung.

alle Einstellungen und Parameter der View abgespeichert. Nach dem Laden des gespeicherten Setups hat der Benutzer so die Möglichkeit genau an der gleichen Stelle die Arbeit wieder aufzunehmen. Bereich (2) ist für die Erstellung und Ausführung einer Konfigurationsdatei für die kommandozeilenbasierte Ausführung zuständig. Bei der Erstellung der Konfigurationsdatei werden alle relevanten Parameter der MICA-Anwendung berücksichtigt. Das bedeutet nicht zwangsläufig, dass diese Datei bereits direkt verwendbar ist. Unter Umständen muss die Datei noch manuell angepasst werden. Analog zur Ausführung dieser Konfigurationsdatei über eine Konsole lässt sich dies über die grafische Benutzeroberfläche durchführen. In Bereich (3) lassen sich zum einen die Profile aus der Anwendung exportieren. Zudem besteht hier die Möglichkeit Bilder der Visualisierungen als Datei zu exportieren. In der Mitte befindet sich das Menü zu „Input“. In (4) kann eine Vorzerlegung (Splitpunkte, siehe Kapitel 3.6) der Profile für das Alignment durchgeführt werden. Dabei wird der Benutzer durch die grafische Oberfläche unterstützt. Das Menü „Plot“ bietet in Bereich (5) die Möglichkeit die Position der Legende in der Visualisierung zu bestimmen. Zudem existiert die Option die Legende auszublenden.

4.4.7 Exportierung von Profilen und Visualisierungen

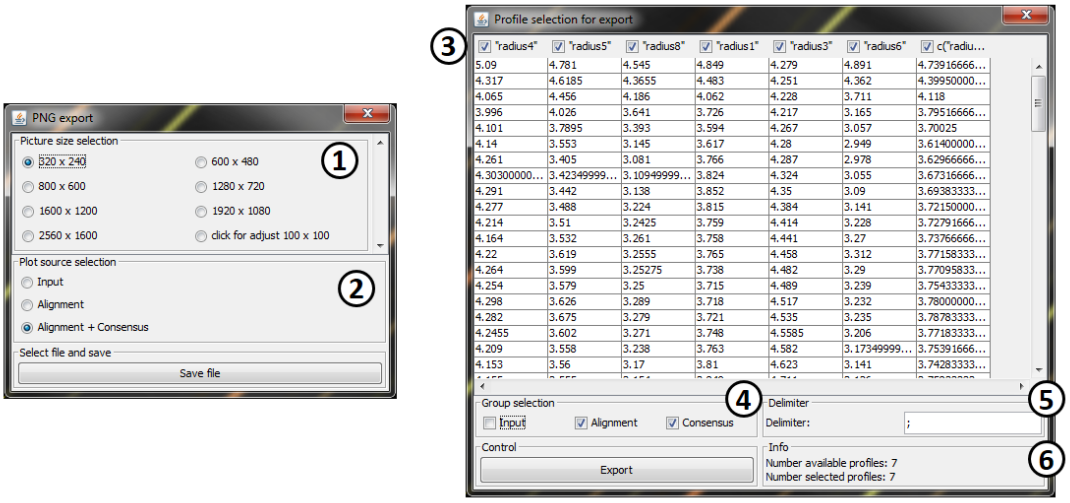

Abbildung 4.12: Dialoge zur Exportierung von Profilen und Visualisierungen. (1) Auflistung der zu exportierenden Visualisierung, (2) Spezifizierung über den Inhalt der exportierten Visualisierung, (3) Auswahl der zu exportierenden Profile, (4) Gruppenweise Vorselektierung der zur exportierenden Profile, (5) Trennzeichen in der exportierten Datei, (6) Information über die Anzahl der zu exportierenden Profile.

Abbildung 4.12 zeigt die Dialoge für die Exportierung von Daten aus der MICA-Anwendung. Nummerierung (1) legt die Auflösung des zu exportierenden Bildes fest. Durch (2) kann festgelegt werden, welchen Inhalt das Bild zeigen soll. Für die Exportierung von Profildaten aus der MICA-Anwendung lässt sich in (4) eine grobe Vorauswahl treffen, welche Profile für den Exportierungsvorgang berücksichtigt werden sollen. In (3) wird die grobe Vorauswahl dargestellt. Hier können gezielt einzelne Profile aus dem Exportierungsvorgang ausgeschlossen werden. Bereich (5) zeigt die Spezifizierung des Trennzeichens in der Ausgabedatei. In (6) wird die Information gegeben, wie viele Profile mit der gegenwärtigen Auswahl exportiert werden.

4.5 Kommandozeilencontroller

Beim Kommandozeilencontroller handelt es sich um eine Implementierung einer Komponente, welche die automatisierbare Ausführung der MICA-Anwendung

zulässt. Um den Kommandozeilencontroller zu verwenden sind folgende drei Voraussetzungen erforderlich:

- Ausführbare MICA-Anwendung
- Terminal/Eingabeaufforderung
- MICA-Konfigurationsdatei

Zunächst wird näher auf die Verwendung des Terminals eingegangen. Im Anschluss daran wird die erforderliche MICA-Konfigurationsdatei genauer beschrieben.

4.5.1 Kommandozeilenaufruf und Konfigurationsdatei

Mit der Eingabe von **java -jar mica.jar** wird die Anwendung über das Terminal mit den Standardparametern gestartet. Fügt man zusätzlich **-?** als Parameter an, erhält man alle verfügbaren Kommandozeilenparameter, welche in folgendem Listing 4.5.1 zu sehend sind.

```
java -jar mica.jar -?

+++ MICA - Multiple Interaval based Curve Alignment +++
The following command line parameters are allowed
-v -view
   Starts the program with a graphical user interface
-c -conf [file.cfg]
   Starts the program without a graphical user interface.
   Additionally the program expects an configuration file.
-t -template [file.cfg]
   Creates an empty configuration template file.
-h -? -help
   Prints this usage message.
```

Listing 4.2: Verfügbare Kommandozeilenparameter

Für die Ausführung der Anwendung mit Hilfe des Kommandozeilencontrollers ist zum einen der Parameter **-t** zum Erstellen einer leeren Konfigurationsdatei relevant. Zudem ist der Parameter **-c** für die Ausführung einer Konfigurationsdatei erforderlich. Mit dem Befehl **java -jar mica.jar -t example.cfg** erhält man eine leere Konfigurationsdatei **example.cfg**, welche in Listing 4.5.1 zu sehen ist.

```
#
# Property configuration file for setting up the MICA application.
# Created at 2014-01-07 11:39:13
```

```
#

# Input CSV file specification. Multiple input files have to be separated by
# semicolon.
input_csvset=

# Input batch processing flag.
# true - For every input profile file an separated alignment will be created.
#        The name of the corresponding input file will be extended to the
#        result files of the alignment.
# false - The alignment will be created on basis of all defined input profile
#         files
input_batchproc=

# Output CSV file specification
output_csv=

# Output picture files definition (optional)
# pic_input defines location for the input profile picture
# pic_align defines the location for the aligned warped input profiles picture
# pic_cons defines the location of the consensus result picture
pic_input=
pic_align=
pic_cons=

# Output PNG resolution specification (width and height)
pic_width=
pic_height=

# Specifies the number of data points a profile will be imported
# A negative number disables the data point derivation. But the
# minimum number of required data points will be derived nevertheless.
num_datapoints=

# Specifies the global filter value for extreme filtering
# (Negative value disables the filter creation)
extreme_filter=

# Specifies the global filter value for inflection filtering
# (Negative value disables the filter creation)
inflection_filter=

# Specification of the distance computation values
# DATA  - The profiles data points are used
# SLOPE - The profiles slope points are used
distance_strategy=

# Specification of the alignment type
# MICA - Progressive multiple alignment will be used
# RMICA - Reference alignment will be used (only the first profile name in
#         property reference_profiles will be considered)
# PRMICA - Progressive reference alignment will be used (all profile names in
#          the property reference_profiles will be considered)
alignment_type=
```

```
# Reference profile specification separated by semicolon. At least one profile
# has to be specified if RMICA or PRMICA is used as alignment_type property
reference_profiles=

# Specifies the maximum warping length factor.
max_warpfactor=

# Specifies the separator of the CSV files
csv_separator=

# Specifies the header existence in the CSV files
# (true, false)
csv_hdr=

# Specifies the slope interval computation length
slope_ival=

#
# End of property input file
#
```

Listing 4.3: Template Konfigurationsdatei

Das erste Attribut **input_csvset** in der Konfigurationsdatei erwartet Pfade zu CSV-Dateien, welche die Daten für die zu alignierenden Profile beinhalten. Es ist möglich mehr als eine Datei zu spezifizieren. Dabei werden mehrere Dateipfade durch das „**;**“ Symbol voneinander getrennt. Bei diesem Attribut handelt es sich um ein Pflichtattribut. Das bedeutet, dass das Setzen von **input_csvset** zwingend erforderlich ist, um das Programm auszuführen.

Das nächste Attribut **input_batchproc** spezifiziert, wie mit dem vorherigen Attribut **input_csvset** verfahren werden soll. Bei **input_batchproc=true** wird für jede spezifizierte Eingabedatei ein eigenes Ergebnis erzeugt. Dabei wird die Ergebnisdatei in Abhängigkeit der benutzten Eingabedatein entsprechend im Dateiname gekennzeichnet. Mit **input_batchproc=false** werden alle Dateien importiert und es wird genau ein Ergebnis erzeugt, auf der Grundlage aller Daten. Dieses Attribut ist auch zwingend erforderlich für die Programmausführung und darf nicht leer belassen werden.

Bei **output_csv** handelt es sich um das Attribut, welches angibt wo das Ergebnis der Berechnung gespeichert werden soll. Bei diesem Attribut handelt es sich um ein Pflichtattribut.

Mit **pic_input, pic_align, pic_cons** kann festgelegt werden, wo die Bilder der grafischen Darstellung abgespeichert werden sollen. Dabei steht **pic_input** für

das Bild des Graphen, welcher die Profile zum Zeitpunkt der Importierung darstellt. Bei **pic_cons** sind alle Profile des Ergebnisses zu sehen, einschließlich dem Konsensusprofil. Im Gegensatz dazu fehlt bei **pic_align** das Konsensusprofil im Bild der Ergebnisdarstellung. Alle drei eben beschriebenen Attribute sind optional. Das bedeutet, dass beim Leerlassen eines der Attribute kein entsprechendes Ergebnisbild erzeugt wird.

Mit **pic_width** und **pic_height** wird die Auflösung für die Bilder, welche zuvor durch **pic_input, pic_align, pic_cons** definiert wurden, festgelegt.

Das Attribut **num_datapoints** legt fest, wie viele Datenpunkte ein jedes der Profile nach dem Importieren haben soll. Entsprechend dem festgelegten Wert werden unter Umständen Datenpunkte verworfen oder durch Interpolation eingefügt. Wird dieser Wert auf „-1“ gesetzt, so wird zunächst keine Anpassung der Datenpunktanzahl durchgeführt. Das bedeutet, dass Profile mit unterschiedlicher Anzahl an Datenpunkten importiert werden können. Wenn ein Profil weniger als die erforderlichen minimalen Datenpunkte besitzt (derzeit ist diese 25), dann greift folgende Ausnahme. Für diese Ausnahme findet dennoch ein Normierung der Profile statt. Allerdings nur bis zu der minimal erforderlichen Anzahl an Datenpunkten.

Die beiden Attribute **extreme_filter** und **inflection_filter** erzeugen bei einem positiven Wert einen entsprechenden globalen Filter. Die Erzeugung und Verwendung von lokalen Filtern ist bei der Nutzung des Kommandozeilencontrollers nicht vorgesehen. Das Attribut **extreme_filter** erzeugt einen Filter, welcher für das Filtern von Hoch- und Tiefpunkten verantwortlich ist. Das Attribut **inflection_filter** erzeugt einen Filter, welcher für das Filtern von Wendepunkten verantwortlich ist. Ist der Wert des Attributes negativ, wird kein Filter erzeugt. Bei beiden Attributen handelt es sich um Pflichtattribute.

Das Pflichtattribut **distance_strategy** legt die Strategie für die Distanzberechnung fest. Dabei wird bei **distance_strategy=DATA** die Distanzberechnung auf der Grundlage der Datenwerte berechnet. Bei **distance_strategy=SLOPE** werden die Steigungswerte verwendet.

Das Attribut **alignment_type** legt fest, welche Vorgehensweise bei der Alignierung verwendet wird. Bei **MICA** handelt es sich um die normale Vorgehensweise für die multiple Alignierung. Bei **RMICA** wird eine Alignierung gegen ein definiertes Referenzprofil durchgeführt. Das bedeutet, dass immer das verfügbare Profil mit der besten Distanz gegen das Referenzprofil aligniert wird. Alternativ

dazu werden mit **PRMICA** mehrere Referenzprofile zugelassen. Das Verfahren ist eine Kombination aus den anderen beiden. Dieses Attribut ist ein Pflichtattribut.

Durch das Attribut **reference_profiles** werden die Profile festgelegt, welche als Referenzprofile betrachtet werden. Dabei handelt es sich um die Profilnamen aus der Eingabedatei, welche durch das Strichpunkt-Symbol („**;**“) voneinander getrennt werden. Sind mehrere Referenzprofile definiert und der Alignierungstyp ist auf **REFERENCE** gesetzt, so wird lediglich das erste Profil als Referenzprofil behandelt.

Mit **max_warpfactor** kann ein Faktor festgelegt werden, welcher in das Verzerren eingreift und eine obere Schranke liefert. Hierbei handelt es sich um ein Pflichtattribut.

Durch **csv_separator** wird festgelegt, welches Trennzeichen innerhalb einer CSV-Datei zu erwarten ist. Dieses Attribut ist ein Pflichtattribut.

Das Attribut **csv_hdr** legt fest, dass mit **csv_hdr=true** Kopfzeilen in den CSV-Dateien zu erwarten sind. Mit **csv_hdr=false** wird von CSV-Dateien ohne Kopfzeile ausgegangen. Dieses Attribut ist ein Pflichtattribut.

Mit dem Attribut **slope_ival** wird die Größe des verschiebbaren Fensters festgelegt, welches verwendet wird, um die Steigungswerte von Profilen zu ermitteln. Dieses Attribut ist ein Pflichtattribut.

4.5.2 Aufbau

Im Folgenden wird nun auf die Umsetzung des Kommandozeilencontrollers eingegangen. Abbildung 4.13 zeigt das UML-Klassendiagramm, welches für den Kommandozeilencontroller relevant ist.

Das Objekt **MicaMain** fragt nach dem Start der Anwendung die Parameter des Terminals ab. Diese Parameter sind über die Signatur der **main(String[])** Methode abfragbar. Wird erkannt, dass die Parameter eine Verwendung des Kommandozeilencontrollers erfordern, erzeugt das **MicaMain**-Objekt sowohl das **CommandLineConfig**-Objekt als auch das **CommandLineController**-Objekt.

Beim **CommandLineConfig**-Objekt handelt es sich um eine Datenstruktur, welche ein exaktes Mapping zu einer initialisierten Konfigurationsdatei darstellt. Im

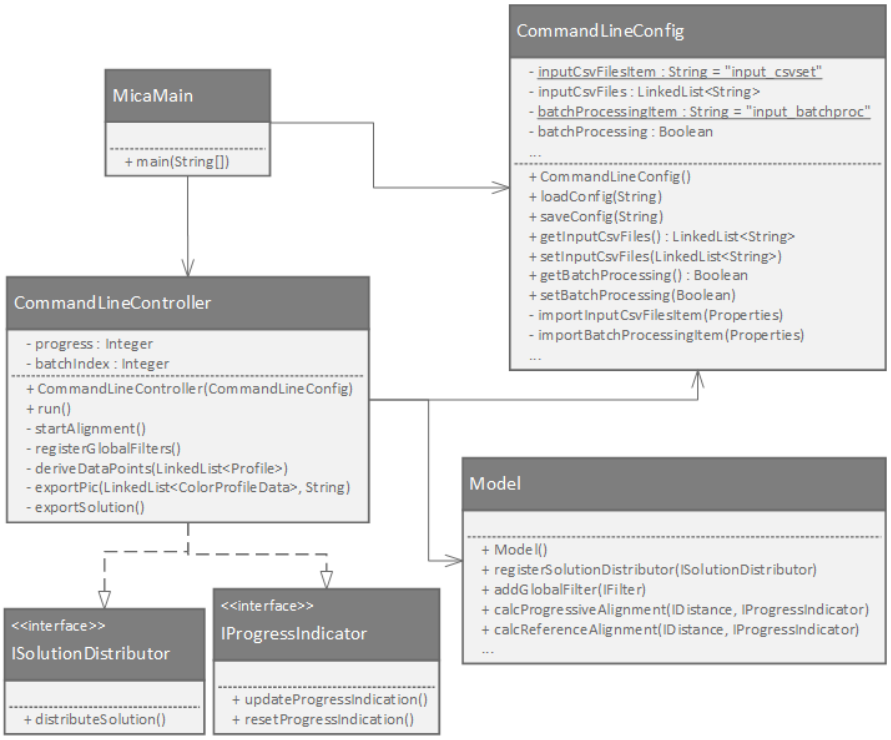

Abbildung 4.13: Klassendiagramm Kommandozeilencontroller

Klassendiagramm sind einige Methoden und Attribute aus Darstellungsgründen nicht aufgeführt. Die Konfigurationsdatei wird verwendet, um mit Hilfe der Methode **loadConfig(String)** das **CommandLineConfig**-Objekt zu initialisieren. Durch Methoden wie **importInputCsvFileItem(Properties)** und den konstant definierten Attributen wie **inputCsvFilesItem** können die entsprechenden Attribute des Objektes durch die Konfigurationsdatei initialisiert werden. Das für die eben erwähnte Methode relevante Attribut wäre **inputCsvFiles**. Dies geschieht nun für alle Attribute, welche bereits zuvor am Beispiel der leeren Konfigurationsdatei beschrieben wurden. Abgefragt werden die Attribute über die Zugriffsfunktionen der einzelnen Attribute (zum Beispiel **getInputCsvFiles()**). Zudem bietet die Klasse Schnittstellen an, um Attribute des Objektes zu setzen. Dies ist erforderlich, wenn eine Konfigurationsdatei mit den gegenwärtig gewählten Parametern persistent gespeichert werden sollen.

Mit Hilfe des initialisierten **CommandLineConfig**-Objekts lässt sich nun das **CommandLineController**-Objekt instanziieren. Anschließend initialisiert das Controller-Objekt das Model gemäß der Konfigurationsdatei. Dabei implementiert das Controller Objekt das **ISolutionDistributor**-Interface und **IProgressIndicator**-Interface um nach einer erfolgreichen Berechnung die Ergebnisse zu

verwalten. Die Methode **run()** startet die Alignierung im **Model**, welches die Ergebnisse über das Interface **ISolutionDistributor** an den Controller zurückgibt. Der Controller wird nun die Ergebnisse exportieren. Dabei werden die entsprechenden Attribute des **CommandLineConfig**-Objektes abgefragt.

Kapitel 5

Evaluation

Dieses Kapitel befasst sich mit der Evaluation des MICA-Systems. Dazu zählt die Evaluierung des Systems mit Parallelisierung im Gegensatz zum System ohne Parallelisierung. Am Ende dieses Kapitels wird die Evaluation des hier umgesetzten dynamischen MICA-Systems mit dem bereits existenten statischen MICA-System verglichen.

5.1 Parallelisierung

In der Tabelle 5.1 können die Ergebnisse der Laufzeitevaluation entnommen werden.

Die Daten der Testumgebung und Evaluationsumgebung sind wie folgt:

- CPU mit vier Rechenkernen (Intel(R) Core(TM) i7 CPU M620 @ 2,67GHz)
- 8 GB Arbeitsspeicher
- 64 Bit-Betriebsystem
- Windows 7 Professional
- Java Runtime Environment (Version 1.7.0_45)
- Zufällig ausgewählte Profildaten aus einem Datenpool[1]. Alle für den Test genutzten Profile wurden mit gleicher Anzahl an Datenpunkten importiert.

[1] `http://users.stat.umn.edu/~sandy/alr3ed/website/`, Applied Linear Regression, 3rd Ed.

Dabei sind die gewählten Profildaten für den Testlauf mit und ohne Parallelisierung identisch.

- Unabhängige Durchführung der Messung mit Threads und ohne Threads

Anz. Profile	Ohne Parallelisierung	Mit Parallelisierung	Speedup
2	0m 0s 39ms	0m 0s 35ms	+10,26 %
8	0m 1s 895ms	0m 2s 21ms	-6,65%
20	0m 15s 601ms	0m 9s 861ms	+63,79%
38	0m 35s 405ms	0m 20s 316ms	+42,62%
54	2m 2s 658ms	1m 7s 2ms	+45,37%

Tabelle 5.1: Messwerte für zufällig gewählte Profildaten mit identsicher Datenpunktanzahl ohne Parallelisierung und mit Parallelisierung. Zudem Laufzeitbeschleunigung durch Parallelisierung in Prozent.

Die Ergebnisse der Messungen sind in Tabelle 5.1 zusammengefasst. Wie bereits beschrieben, lohnt sich eine Parallelisierung erst bei größeren Berechnungen. Dies lässt sich auch deutlich der Tabelle 5.1 und Abbildung 5.1 entnehmen. Bei zwei oder gar acht Profilen müssen verhältnismäßig wenige Zellen der Distanzmatrix berechnet werden. Bei 20, 38 oder gar 54 Profilen lohnt sich der Einsatz von Threads zur Parallelisierung enorm. Denn dort steigt die Laufzeitkurve deutlich weniger an (siehe Abbildung 5.1). Die mittlere Laufzeitbeschleunigung durch die Parallelisierung für diese Evaluation liegt bei 25,68%.

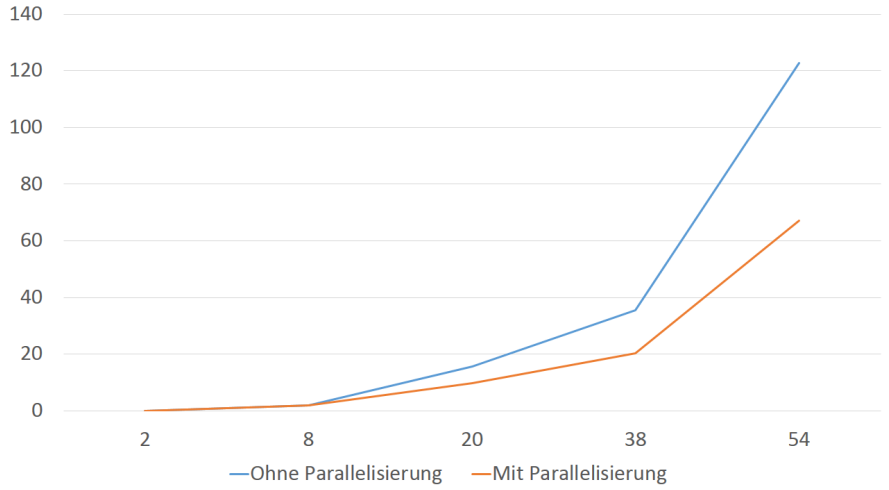

Abbildung 5.1: Vertikale Achse: Laufzeit der MICA-Berechnung in Sekunden. Horizontale Achse: Anzahl an Profilen für die Durchführung. Ohne Parallelisierung (blau). Mit Parallelisierung (orange).

5.2 Vergleich dynamisches MICA versus statisches MICA

Für die Durchführung der Evaluation zwischen beiden Alignierungsvarianten werden Dichteprofile von Baumstämmen verwendet. Abbildung 5.2 zeigt, wie die Daten erhoben werden. Anhand eines Baumquerschnitts werden verschiedene radiale Messungen zur Bestimmung der Dichte durchgeführt. Abbildung 5.3 zeigt dabei einen Ausschnitt einer Radialmessung, welche vier Wachstumsjahre abdeckt. Für ein Jahr wird genau ein Profil erstellt.

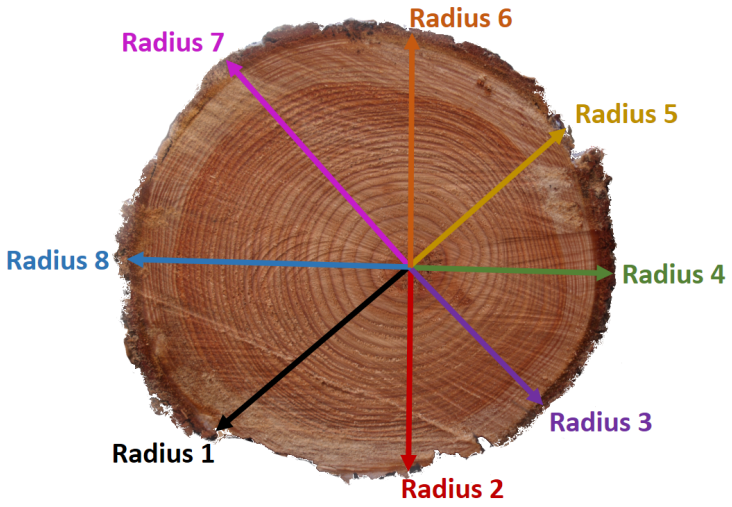

Abbildung 5.2: Querschnitt eines Baumstamms mit Jahresringen.[2] Radialmessungen ausgehend vom Zentrum liefern die Grundlage für die Profildaten.

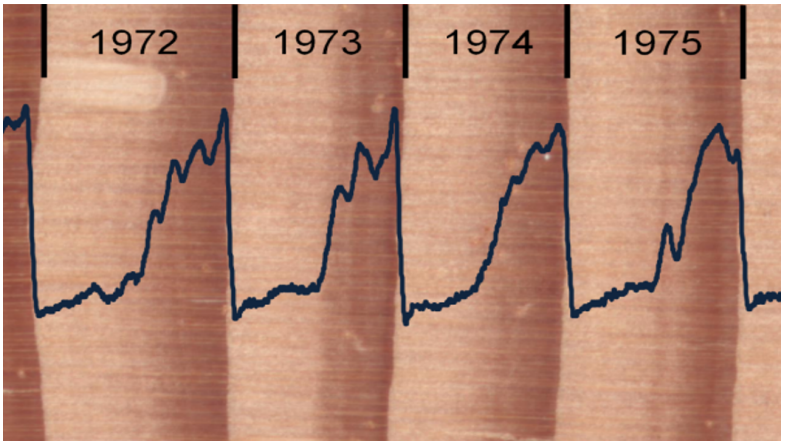

Abbildung 5.3: Ausschnitt aus den Messdaten eines Radius. Ein Jahresabschnitt entspricht einem Profil. (Abbildung aus [Bender et al., 2012])

[2] `http://commons.wikimedia.org/wiki/File:Tree_rings.jpg`, Fotovorlage von Abbildung 5.2, Jahresringe im Querschnitt eines Baumstamms

Mit den Dichteprofilen lassen sich Aussagen über das Wachstum eines Baumes treffen oder Beziehungen zu klimatischen Bedingungen aufzeigen (vgl. [Fritts, 1976], [Schweingruber, 1988], [Vaganov et al., 2006] und [Briffa et al., 2001]). Dabei ist die Dichte des Holzes direkt von der Wasserverfügbarkeit während einer Vegetationsperiode abhängig (vgl. [Rozenberg et al., 2002] und [Kozlowski, 1971]).

Für die Datengrundlage der Evaluation gilt folgende Ausgangssituation:

- 7 Bäume
- Je Baum 8 Radien-/Dichtemessungen
- Messzeitraum von 50 Jahren je Baum

Für die Evaluationsdurchführung gelten folgende Parameter:

- Für je ein Jahr und Baum und Radius wird ein Profil mit einer y-Normierung von 200 Datenpunkten erstellt.
- Ein globaler Extrempunktfilter mit einer Rauschunterdrückung von 5% filtert die Hoch- und Tiefpunkte für jedes Profil.
- Ein globaler Wendepunktfilter, welcher alle Wendepunkte in jedem Profil entfernt (entspricht einem Filterwert von 2 für die verwendeten Daten).
- Steigungswertermittlung durch ein gleitendes Fenster der Länge 2
- Distanzbewertung auf der Grundlage von Steigungswerten
- Ein Faktor von 2 für die maximale Verzerrungslänge
- Eine minimale Intervallzerlegung der Länge 10

Für die Evaluationsdurchführung wurde auf die Verwendung von Wendepunkten als Referenzpunkte verzichtet. Dies ist nötig, da der Filtermechanismus für die Wendepunkte in beiden Implementierungsvarianten inkompatibel ist. Die R Implementierung setzt auf einen Rauschfilter, welcher auf normierte Steigungswerte aufsetzt und diese zum globalen Maximum und Minimum in Relation setzt. Bei der Java-Implementierung wird ein Schwellwert verwendet, um die unnormierten Steigungswerte zu vergleichen.

Im Folgenden wird die Evaluation der in dieser Arbeit in Java entwickelten dynamisch progressiven Alignierungsmethode beschrieben und die Ergebnisse der Durchführung dargestellt. Zum Vergleich dient ein bereits erwähnter statisch progressiver Alignierungsanstz, welcher in R entwickelt wurde (vgl. [Bender et al., 2012]). Die folgende Abbildung 5.4 zeigt den Aufbau der Evaluationsdurchführung.

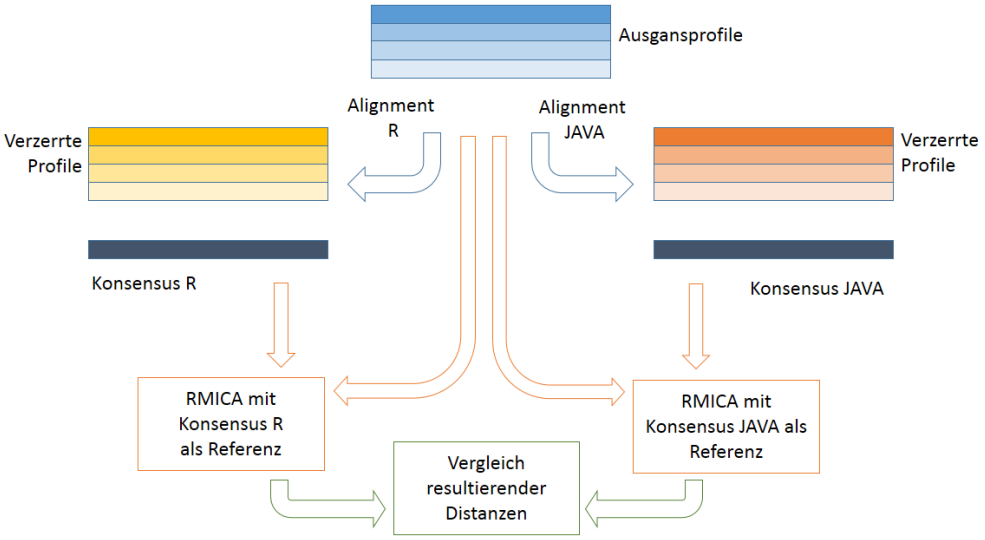

Abbildung 5.4: Ablauf der Evaluation. Alignierung der Ausgangsprofile mit beiden Implementierungen. Referenzbasiere Alignierung (RMICA) zwischen den Ausgangsprofilen und den Konsensusprofilen als Referenz. Vergleich der resultierenden Distanzbewertungen für die Java-Implementierung und R-Implementierung.

Die acht Radialmessungen pro Jahr bilden jeweils die Ausgangsprofile für eine Evaluation. Im nächsten Schritt wird sowohl für die Java-Implementierung als auch für die R-Implementierung die MICA-Methode auf die Ausgangsprofile angewandt. Das Ergebnis dieses Schrittes ist die Menge der verzerrten Ausgangsprofile, sowie das Konsensusprofil der jeweiligen Ausführung. Um nun zu vergleichen, wie genau der Konsensus die Ausgangsprofile repräsentiert, wird die RMICA-Methode verwendet, welche in Kapitel 3.5 beschrieben ist. Dabei wird das jeweilige Konsensusprofil als Referenz definiert und im Folgenden die Ausgangsprofile an die Referenz aligniert.

Ausgehend von 7 Bäumen und einem Messungszeitraum von 50 Jahren erhalten wir 350 Distanzwerte für die Evaluation einer Implementierung. Dabei wird für je

ein Jahr und Baum über die 8 Radien ein Distanzwert ermittelt. Dabei werden für jedes der Ausgangsprofile paarweise Kombinationen mit dem Konsensus gebildet. Auf jede dieser Kombinationen wird die RMICA-Methode angewandt, wobei der Konsensus in jedem Fall das Referenzprofil darstellt. Der finale Distanzwert aller Radien zum Konsensus wird als Mittelwert der paarweisen RMICA-Distanzen berechnet. Ein Vergleich der resultierenden Distanzbewertungen für die R und Java-Implementierung lässt einen Vergleich beider Ansätze zu. Dabei verspricht man sich von dem dynamischen Ansatz bessere Ergebnisse als von dem statischen Ansatz.

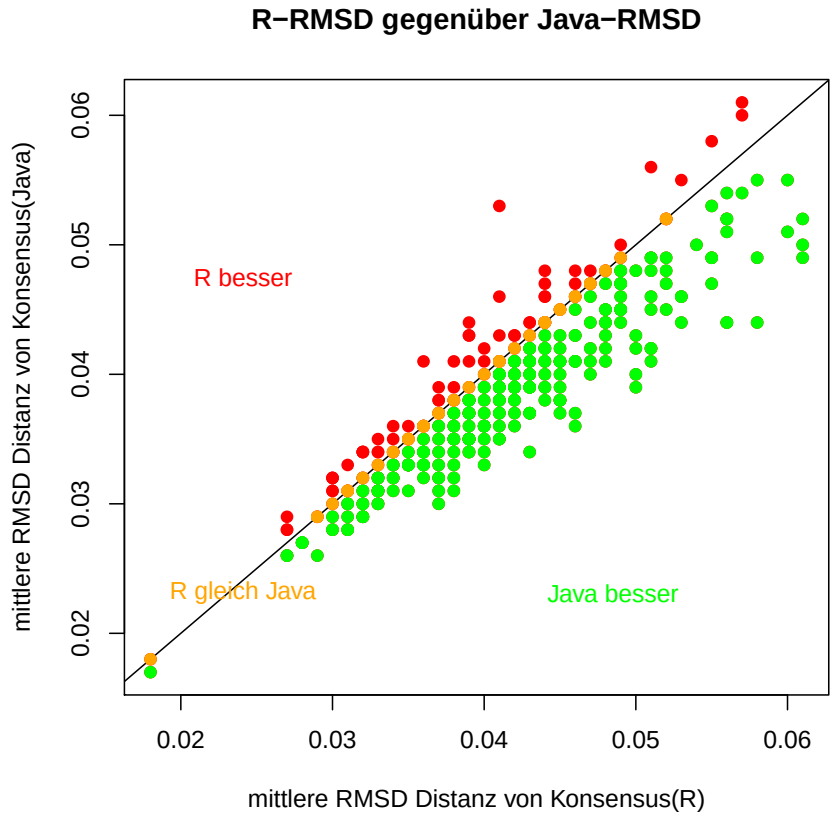

Abbildung 5.5: Vergleich der RMSD-Distanzen beider Implementierungen, x-Achse: R-Distanz, y-Achse: Java-Distanz, Punkt grün → Java hat geringeren Distanzwert, Punkt rot → R hat einen geringeren Distanzwert, Punkt orange → Java und R liefern denselben Distanzwert. Gleiche Distanz beider Implementierungen entspricht einem Punkt (orange) auf der Diagonalen. Ein Punkt entspricht der mittleren Distanz (über alle 8 Radien) eines Baumes für ein Jahr.

Abbildung 5.5 zeigt das Ergebnis der Evaluation. Dabei entspricht ein Punkt dem

jeweiligen Distanzwert der R und Java-Implementierung für die gleichen Ausgangsdaten. Befindet sich ein Punkt über der eingezeichneten Diagonale, ist die Distanz der Java-Implementierung größer als die der R Implementierung. In diesem Fall wird der Punkt rot gekennzeichnet und bedeutet, dass die R-Implementierung ein besseres Alignment erstellt hat. Analog dazu entspricht ein Punkt (grün) unterhalb der Diagonale für ein besseres Ergebnis der Java-Implementierung. Sind die Distanzwerte beider Implementierungen identisch, befindet sich der Punkt (orange) auf der Diagonale in der Abbildung. In 49 Fällen gab es keinen messbaren Unterschied zwischen den beiden Implementierungen. Der Distanzwert der R-Implementierung war in 51 Fällen besser und in 250 Fällen war der Distanzwert der Java-Implementierung besser. Alle Distanzwerte sind dabei auf die dritte Nachkommastelle gerundet. Das Ergebnis dieser Evaluation zeigt, dass die dynamisch progressive Implementierung in Java häufiger (zu 71%) zu einer geringeren Distanz und somit zu einem besseren Ergebnis führt als die statische Implementierung in R (zu 14%).

Ein Quotient aus $\frac{Java_{RMSD}}{R_{RMSD}}$ für ein Jahr und Baum gibt an, welche Implementierung das bessere Alignment erstellt hat. Ist dieser Quotient 1, so haben beide Alignments die gleiche Distanz. Ein Quotient kleiner als 1 bedeutet ein besseres Alignment (kleinere Distanz) der Java Implementierung und ein Quotient größer 1 bedeutet ein besseres Alignment der R-Implementierung. Der mittlere Quotient beträgt dabei $0,9490$ und dies bedeutet, dass die Java-Implementierung im Schnitt bessere Alignments erstellt hat. Für diese Evaluation errechnet sich eine Standardabweichung von $0,0697$. Unter Berücksichtigung einer Normalverteilung der Streuung der Quotientwerte gibt es im Schnitt einige wenige Alignments, welche einen Quotienten von $\frac{Java_{RMSD}}{R_{RMSD}} \geq 1$ aufweisen. Abschließend lässt sich sagen, dass die Java-Implementierung häufiger zu einem besseren Alignmentergebnis mit geringerer Distanz führt als dies die R-Implementierung tut.

Die folgenden Abbildungen 5.6 und 5.7 zeigen für alle Bäume, gruppiert nach Jahren, die RMSD-Distanzwerte der Evaluation. Abbildung 5.6 zeigt die Ergebnisse der Java-Implementierung und Abbildung 5.7 die Ergebnisse der R-Implementierung.

Mit Hilfe dieser Darstellung lassen sich problematische Jahre schnell identifizieren. In beiden Abbildungen ist zu sehen, dass es Ausreißer bei den RMSD-Werten gibt. Ein Ausreißer ist an einem hohen Distanzwert zu erkennen. Ein hoher Distanzwert bedeutet, dass das Alignment schlechter durchgeführt werden konnte als bei einem

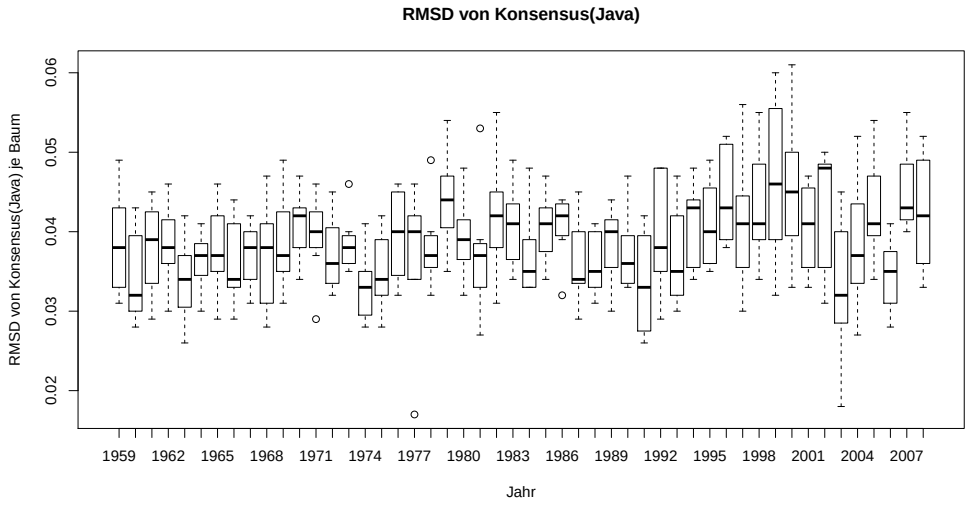

Abbildung 5.6: RMSD-Distanzen der Java-Implementierung gruppiert nach Jahren für alle 7 Bäume

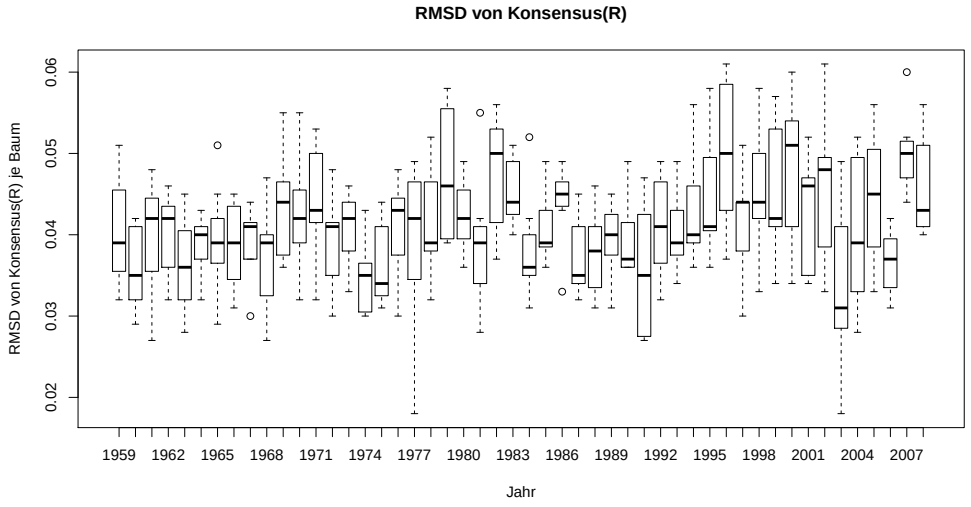

Abbildung 5.7: RMSD-Distanzen der R-Implementierung gruppiert nach Jahren für alle 7 Bäume

niedrigen Distanzwert. Bei genauer Betrachtung von Abbildung 5.6 befindet sich solch ein Ausreißer mit relativ hohem Distanzwert in den Jahren 1979, 1999, 2002 und 2007. Ein vergleichsweise niedriger Distanzwert lässt sich in den Jahren 1991 und 2003 finden.

Als Beispiel dazu zeigt Abbildung 5.8 das Alignment einer hohen Distanz $(0,050)$ für den Baum 6 aus dem Jahresring 2002. Es lässt sich deutlich erkennen, das die dargestellten Kurven sehr unterschiedlich in Form und Charakteristika sind. Auch die Konsensuskurve, welche schwarz und hervorgehoben in der Abbildung

dargestellt ist, spiegelt die Einzelkurven teilweise sehr schlecht wider. Im Vergleich dazu zeigt Abbildung 5.9 eine bessere Alignierung mit niedrigerer Distanz $(0,038)$ für den Baum 103 und dem Jahresring 1991. Hier stellt die Konsensuskurve die Charakteristika der verzerrten Ausgangsprofile in Abbildung 5.9 besser dar.

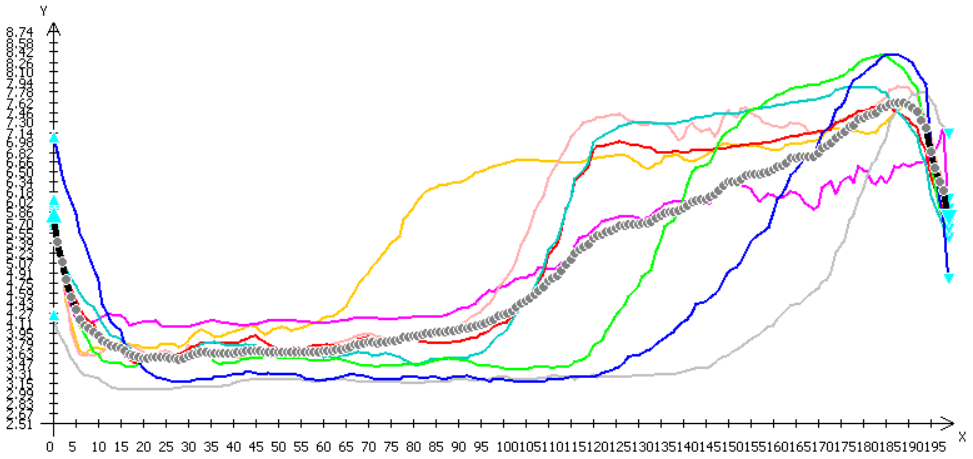

Abbildung 5.8: Ergebnis der Java-Alignierung von Baum 6 aus Jahresring 2002 mit hoher Distanz. Konsensus des Alignments grau gepunktet und dick hervorgehoben.

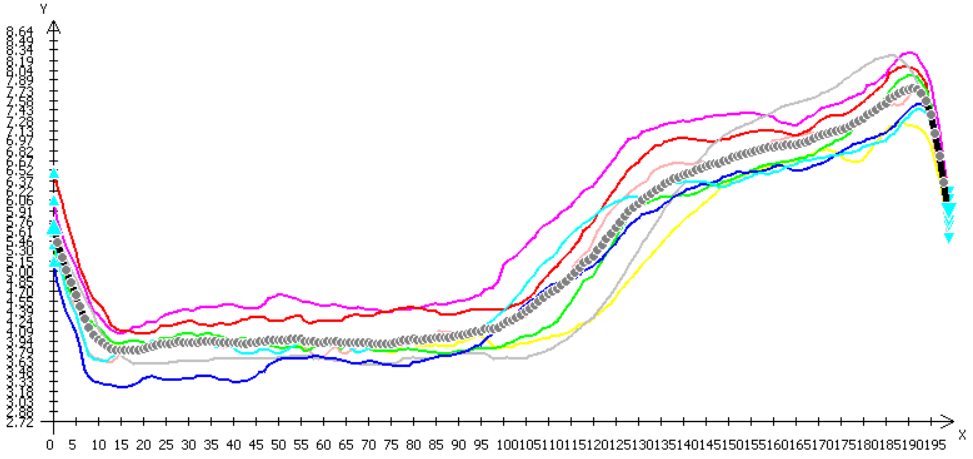

Abbildung 5.9: Ergebnis der Java-Alignierung von Baum 5 aus Jahresring 1991 mit geringer Distanz. Konsensus des Alignments grau gepunktet und dick hervorgehoben.

Kapitel 6

Zusammenfassung

Ausgehend von Kurven (Datensätzen), welche Abweichungen zueinander besitzen, bietet die Alignierung eine Möglichkeit eine repräsentative Kurve zu erstellen, die dabei die Gestalt aller Ausgangskurven widerspiegelt. Es wurde gezeigt, dass eine Mittelwertsberechnung ohne vorausgehende Alignierung eine ungenügend repräsentative Kurve als Ergebnis liefert.

Um die bestmögliche Alignierung auf Basis von Intervallabbildungen durchzuführen, muss zuvor eine optimale Zerlegung gefunden werden. Dies wird durch das Minimieren der Anstiegsdistanz erreicht, welche die Gestalterhaltung während der Alignierung sicherstellt. Dabei bietet die Einführung von Referenzpunkten für das Finden einer optimalen Zerlegung einen Geschwindigkeitsvorteil.

Ein weiterer Geschwindigkeitsvorteil bietet die umgesetzte Parallelisierung von unabhängigen Berechnungen während der Alignierung.

Die Durchführung der multiplen Alignierung basiert dabei auf einem progressiven Ansatz, welcher auf paarweise Alignierungen aufbaut. Die in dieser Arbeit umgesetzte Vorgehensweise basiert grundlegend auf dem Ansatz in [Bender et al., 2012], jedoch mit dem Unterschied, dass anstatt einer statischen Vorgehensweise für die progressive Erstellung eines multiplen Alignments eine dynamische Vorgehensweise umgesetzt wurde.

Die Evaluation des dynamischen Ansatzes und des statischen Ansatzes hat gezeigt, dass der dynamische Ansatz 5-mal häufiger ein Alignment erstellt, welches eine geringere Distanz als das korrespondierende Alignment des statischen Ansatzes aufweist. Dies bedeutet, dass eine dynamisch progressive Vorgehensweise

bessere Resultate liefert als ein vergleichbarer statischer Ansatz und deshalb einem statischen Ansatz vorzuziehen ist.

In der Evaluation wurde zudem gezeigt, wie mit Hilfe der Distanzwerte der Alignierungen problematische Datensätze identifiziert werden können.

Zudem wurden verschiedene Varianten der multiplen Alignierung (MICA, RMICA, PRMICA, SMICA) in dieser Arbeit beschrieben und umgesetzt.

Als Ergebnis dieser Arbeit ist eine Anwendung entstanden, welche eine grafische Benutzeroberfläche zur Verfügung stellt, um die Alignierung von Profildaten/Kurvendaten durchzuführen. Als Ergänzung dazu bietet die Anwendung die Möglichkeit, konsolenbasiert (ohne grafische Benutzeroberfläche) automatisiert Alignmentberechnungen durchzuführen.

Anhang A

Anhang

Im Anhang befinden sich UML-Klassendiagramme sowie Datentabellen zu Profilen aus Beispielen dieser Arbeit.

A.1 Datentabellen zu Kapitel 1

Dieses Kapitel beinhaltet Tabellen, die in Relation zu Kapitel 1 stehen. Die Datentabellen beinhalten Werte für die Datenpunkte (x_i, y_i) der Profile.

x_i	$y(x_i)$	x_i	$y(x_i)$	x_i	$y(x_i)$	x_i	$y(x_i)$
1	10,0	26	0,75	51	7,0	76	1,5
2	9,25	27	0,875	52	6,625	77	1,5
3	8,5	28	1,0	53	6,25	78	1,5
4	7,75	29	1,75	54	5,875	79	1,375
5	7,0	30	2,5	55	5,5	80	1,25
6	6,25	31	3,25	56	5,125	81	1,125
7	5,5	32	4,0	57	4,75	82	1,0
8	4,75	33	4,75	58	4,375	83	1,0
9	4,0	34	5,5	59	4,0	84	1,0
10	3,25	35	6,25	60	3,625	85	1,0
11	2,5	36	7,0	61	3,25	86	1,0
12	1,75	37	7,167	62	2,875	87	1,0
13	1,0	38	7,333	63	2,5	88	1,0

14	0,833	39	7,5	64	2,333	89	1,0
15	0,667	40	7,625	65	2,167	90	0,875
16	0,5	41	7,75	66	2,0	91	0,75
17	0,375	42	7,875	67	1,875	92	0,625
18	0,25	43	8,0	68	1,75	93	0,5
19	0,125	44	7,875	69	1,625	94	0,5
20	0,0	45	7,75	70	1,5	95	0,5
21	0,125	46	7,625	71	1,5	96	0,5
22	0,25	47	7,5	72	1,5	97	0,5
23	0,375	48	7,375	73	1,5	98	0,5
24	0,5	49	7,25	74	1,5	99	0,5
25	0,625	50	7,125	75	1,5	100	0,5

TABELLE A.1: Wertetabelle zu Kurve P mit x-Werten und y-Werten. Motivationsbeispiel aus Kapitel 1.

x_i	$y(x_i)$	x_i	$y(x_i)$	x_i	$y(x_i)$	x_i	$y(x_i)$
1	10,0	26	5,75	51	1,5	76	1,0
2	7,875	27	4,875	52	1,5	77	1,0
3	5,75	28	4,0	53	1,5	78	1,0
4	3,625	29	3,5	54	1,5	79	1,0
5	1,5	30	3,0	55	1,5	80	1,0
6	1,25	31	2,5	56	1,375	81	1,0
7	1,0	32	2,0	57	1,25	82	1,0
8	0,75	33	1,875	58	1,125	83	0,875
9	0,5	34	1,75	59	1,0	84	0,75
10	0,875	35	1,625	60	1,0	85	0,625
11	1,25	36	1,5	61	1,0	86	0,5
12	1,625	37	1,5	62	1,0	87	0,5
13	2,0	38	1,5	63	1,0	88	0,5
14	3,667	39	1,5	64	1,0	89	0,5
15	5,333	40	1,5	65	1,0	90	0,5
16	7,0	41	1,5	66	1,0	91	0,5
17	7,375	42	1,5	67	1,0	92	0,5
18	7,75	43	1,5	68	1,0	93	0,5

19	8,125	44	1,5	69	1,0	94	0,5
20	8,5	45	1,5	70	1,0	95	0,5
21	8,25	46	1,5	71	1,0	96	0,5
22	8,0	47	1,5	72	1,0	97	0,5
23	7,75	48	1,5	73	1,0	98	0,5
24	7,5	49	1,5	74	1,0	99	0,5
25	6,625	50	1,5	75	1,0	100	0,5

TABELLE A.2: Wertetabelle zu Kurve P' mit x-Werteb und y-Werten. Motivationsbeispiel aus Kapitel 1.

A.2 Datentabellen zu Kapitel 3

Dieses Kapitel beinhaltet Tabellen, die in Relation zu Kapitel 3 stehen. Die Datentabellen beinhalten Werte für die Datenpunkte (x_i, y_i) der Profile.

x_i	$y(x_i)$	x_i	$y(x_i)$
1	2,0	13	9,0
2	2,0	14	5,0
3	2,0	15	3,0
4	2,0	16	2,0
5	2,0	17	2,0
6	2,25	18	2,0
7	3,0	19	2,0
8	5,0	20	2,0
9	9,0	21	2,0
10	9,75	22	2,0
11	10,0	23	2,0
12	9,75	24	2,0
		25	2,0

Tabelle A.3: Wertetabelle für P, zu Beispiel paarweise Alignierung aus Kapitel 3.1

x_i	$y(x_i)$	x_i	$y(x_i)$
1	3,0	13	6,0
2	3,0	14	8,0
3	3,0	15	8,75
4	3,0	16	9,0
5	3,0	17	8,75
6	3,0	18	8,0
7	3,0	19	6,0
8	3,0	20	4,0
9	3,0	21	3,25
10	3,0	22	3,0
11	3,25	23	3,0
12	4,0	24	3,0
		25	3,0

Tabelle A.4: Wertetabelle für P', zu Beispiel paarweise Alignierung aus Kapitel 3.1

x_i	$y(x_i)$	x_i	$y(x_i)$
1	2,5	13	8,75
2	2,5	14	8,5
3	2,5	15	7,75
4	2,5	16	5,0
5	2,5	17	3,0
6	2,5	18	2,75
7	2,5	19	2,5
8	2,75	20	2,5
9	3,0	21	2,5
10	5,0	22	2,5
11	7,75	23	2,5
12	8,5	24	2,5
		25	2,5

Tabelle A.5: Wertetabelle für P'', zu Beispiel multiple Alignierung aus Kapitel 3.3

A.3 Klassendiagramme zu Kapitel 4

Dieses Kapitel beinhaltet Klassendiagramme, die in Relation zu Kapitel 4 stehen. Dabei stehen die Klassendiagramme in direktem Bezug zur Implementierung des MICA Systems.

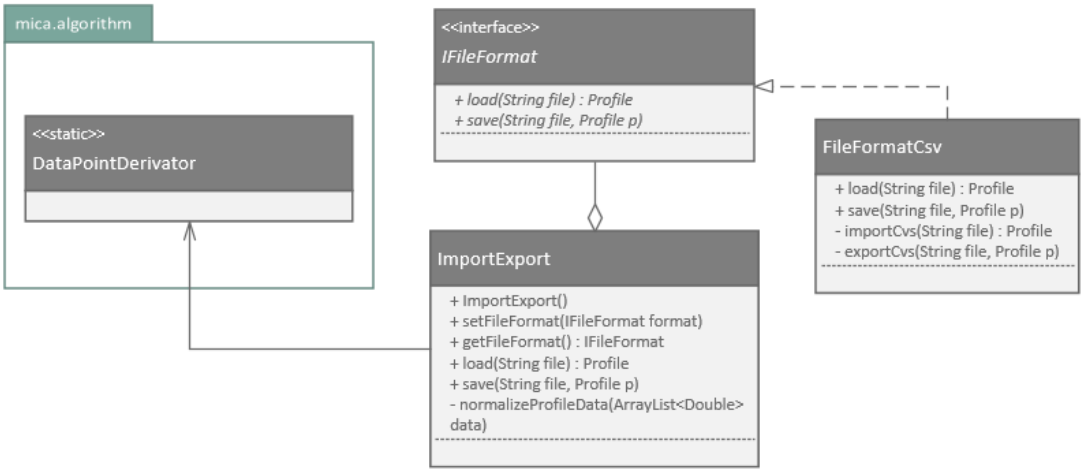

Abbildung A.1: Klassendiagramm Import und Strategie-Pattern

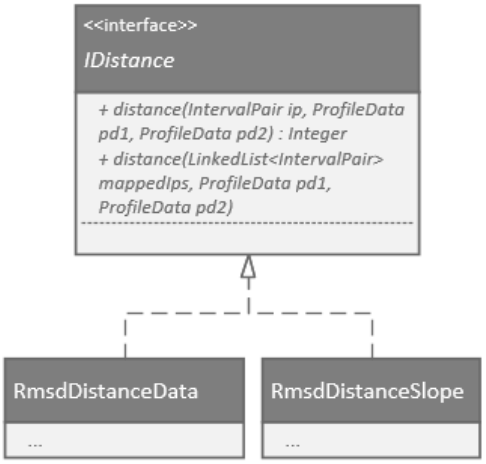

Abbildung A.2: Strategie-Pattern Distanzberechnung

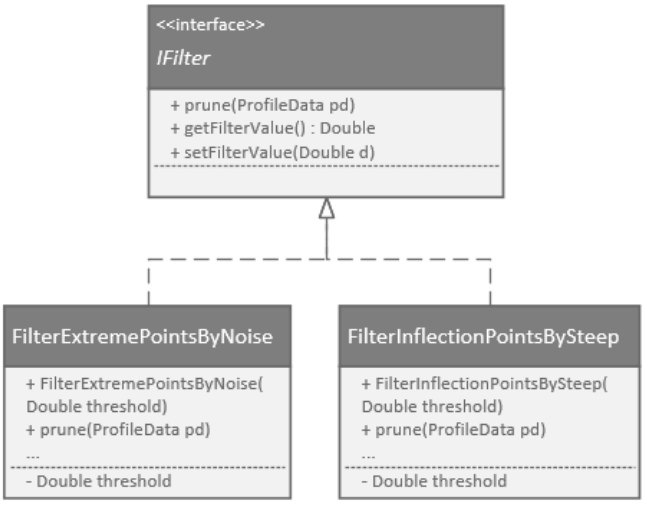

Abbildung A.3: Strategie-Pattern Filtering

Literaturverzeichnis

Ghazi Al-Naymat, Sanjay Chawla, and Javid Taheri. Sparsedtw: A novel approach to speed up dynamic timewarping. Australasian Data Mining, vol. 101, Melbourne, Australia, ACM Digital Library, pp. 117-127, 2012.

Bela J. Bender, Martin Mann, Rolf Backofen, and Heinrich Spiecker. Microstructure alignment of wood density profiles – an approach to equalize radial differences in growth rate. Trees, Structure and Function, Springer-Verlag, Volume 26, Issue 4, pp 1267-1274, 2012.

Keith R. Briffa, Timothy J. Osborn, Fritz H. Schweingruber, Ian C. Harris, Philip D. Jones andStepan G. Shiyatov, and Eugene A. Vagonov. Low-frequency temperature variations from northern ring density network. Journal of Geophysical Research, vol. 106, no. D3, pages 2929-2941, 2001.

Higgins DG, Thompson JD, and Gibson TJ. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22:4673-80, 1994.

D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol., 60:351-360, 1987.

H.C. Fritts. Tree rings and climate. Academic Press, London/ New York/ San Francisco, pp. 567, 1976.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns. elements of reusable object-oriented software. Addison-Wesley, 1994.

Theo Gasser and Alois Kneip. Searching for structure in curve samples. Journal of the American Statistical Association, Vol. 90, No. 432, pp. 1179-1188, 1995.

A. Kneip and J. O. Ramsay. Combining registration and fitting for functional models. Journal of the American Statistical Association, Volume 103, Issue 483, 2008.

Alois Kneip and Theo Gasser. Statistical tools to analyze data representing a sample of curves. The Annals of Statistics, Vol. 20, No. 3, 1266-1305, 1992.

T. T. Kozlowski. Growth and development of trees. Volume II, Academic Press, London/ New York, pp. 514, 1971.

Sebastian Kurtek, Anuj Srivastava, and Wei Wu. Signal estimation under random time-warpings and nonlinear signal alignment. Florida State University, 2011.

Daniel Lemire. Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern Recognition 42(9), pp. 2169-2180, 2009.

Xueli Liu and Mark C.K. Yang. Simultaneous curve registration and clustering for functional data. Computational Statistics and Data Analysis 53, 1361-1376, 2009.

Wolfgang Otto, Sebastian Will, and Rolf Backofen. Structure local multiple alignment of rna. Proceedings of German Conference on Bioinformatics (GCB'2008). LNI. 178-188, 2008.

François Petitjean and Pierre Gançarski. Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment. Elsevier B.V.,Theoretical Computer Science 414, 76–91, 2012.

François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping with applications to clustering. Elsevier Ltd., Pattern Recognition 44, 678–693, 2011.

Lawrence R. Rabiner and Carolyn E. Schmidt. Application of dynamic time warping to connected digit recognition. IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-28, No. 4, 1980.

J. O. Ramsay and Xiaochun Li. Curve registration. Journal of the Royal Statistical Society. Series B (Statistical Methodology), Vol. 60, No.2, pp. 351-363, 1998.

John A. Rice and B. W. Silverman. Estimating the mean and covariance structure nonparametrically when the data are curves. J. R. Statist. Soc. B 53, No. 1, pp. 233-243, 1991.

Philippe Rozenberg, Julien Van Loo, Bjorn Hannrup, and Michael Grabner. Clonal variation of wood density record of cambium reaction to water deficit in picea abies (l.) karst. Annals of Forest Science 59, 533–540, 2002.

Laura Maria Sangalli, Piecesare Secchi, Simone Vantini, and Valeria Vitelli. K-means alignment for curve clustering. MOX–Report No. 13, 2008.

Fritz Hans Schweingruber. Tree rings: Basics and applications of dendrochronology. Dordrecht, Reidel, pp. 276, 1988.

A. Srivastava, W. Wu, S. Kurtek, E. Klassen, and J. S. Marron. Registration of functional data using fisher-rao metric. Florida State University and University of North Carolina, 2011.

Kariya Takeaki and Kurata Hiroshi. Generalized least squares. John Wiley & Sons Ltd, ISBN: 0-470-86697-7, 2004.

Rong Tang and Hans-Georg Müller. Pairwise curve synchronization for functional data. Biometrika, 95, 4, pp. 875–889, 2008.

Eugene A. Vaganov, Malcolm K. Hughes, and Alexander V. Shashkin. Growth dynamics of conifer tree rings: Images of past and future environments: An image of past and future environments. Springer, Berlin/ Heidelberg/ New York, pp. 354, 2006.

Kongming Wang and Theo Gasser. Alignment of curves by dynamic time warping. The Annals of Statistics, Vol. 25, No. 3, 1251-1276, 1997.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Ort, Datum: _____

Unterschrift: _____