Johann Wolfgang Goethe-Universität
Frankfurt am Main

Fachbereich (15) Biologie & Informatik · Institut für Informatik

Diplomarbeit:

Ein adaptives Verfahren zur Modellierung und Verifikation der Unterschriftendynamik

vorgelegt von:

Michael Schneider
<m_schnei@cs.uni-frankfurt.de>
Höbäckerweg 26
36381 Schlüchtern

im September 2004

Betreuer:
PD Dr. habil. Rüdiger Brause
AG Adaptive Systemarchitektur

Erklärung

Hiermit versichere ich, daß ich diese Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur angefertigt habe.

(Michael Schneider)

Inhaltsverzeichnis

Notation

$\boldsymbol{v}$, $\mathbf{A}$	Vektoren und Matrizen
$\boldsymbol{v}^\top$, $\mathbf{A}^\top$	Transponierte des Vektors $\boldsymbol{v}$ bzw. der Matrix $\mathbf{A}$
$\boldsymbol{v}^\top \boldsymbol{w}$	inneres Produkt $\sum_j v_j w_j$ der Vektoren $\boldsymbol{v}$ und $\boldsymbol{w}$
$\boldsymbol{v} \perp \boldsymbol{w}$	zwei Vektoren sind orthogonal zueinander
$\|\boldsymbol{v}\|$	Norm eines Vektors
$\mathrm{d}(\boldsymbol{v}, \boldsymbol{w})$	Abstand zweier Vektoren $\boldsymbol{v}$ und $\boldsymbol{w}$
$u^{(i)}$	i-te Unterschrift
u_j	j-te Komponentensequenz der Unterschrift u
$\mathrm{sim}(u^{(i)}, u^{(k)})$	Ähnlichkeitsbewertung zweier Unterschriften
$(x_t)_{0 \leq t < T}$	Zeitsequenz der Länge T bestehend aus Samples x_t
$(x_t) \star (h_l)$	Konvolution einer Zeitsequenz (x_t) mit einem Filter (h_l)
$\langle (x_t) \rangle_t$, μ_{x_t}	Mittelwert $1/T \sum_t x_t$ aller Samples x_t der Zeitsequenz (x_t)
σ_{x_t}	Standardabweichung der Samples x_t einer Zeitsequenz (x_t)
$\mathrm{Ws}(x)$, $p(x)$	Wahrscheinlichkeit für das Auftreten des Ereignisses x
$\mathrm{H}(p)$	Entropie der Wahrscheinlichkeitsverteilung p
$\lvert M \rvert$	Kardinalität der Menge M
$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$	natürliche, ganze, rationale und reelle Zahlen
$\mathbb{U}$	Menge aller Unterschriftensequenzen (u_t)
$\mathbb{F}$	Featureraum $\mathbb{F}_1 \times \cdots \times \mathbb{F}_n$
$\boldsymbol{f}$	Featuremenge $\{f_1, \ldots, f_n\}$ zum Featureraum $\mathbb{F}$
$\boldsymbol{x}$, $(\boldsymbol{x}_t)$	Featurevektor $(x_1, \ldots, x_n)^\top$ und Featurevektor-Sequenz in $\mathbb{F}$
$\mathcal{M}_P$	Modell einer Person P
S_n	Permutationsgruppe der Zahlen $\{1, \ldots, n\}$
$\partial_x f(\ldots, x, \ldots)$	(partielle) Ableitung der Funktion f nach x
δ_{ik}	Kroneckersymbol: $\delta_{ik} = 1$, falls $i = k$, sonst 0
$\mathrm{O}(\cdot)$, $\Theta(\cdot)$, $\Omega(\cdot)$, $\mathrm{o}(\cdot)$	Asymptotische Notation

Kapitel 1

Einleitung

Unter den zahlreichen für die Personenidentifikation eingesetzten biometrischen Merkmalen nimmt die Unterschrift eine besondere Stellung ein. Ihr Gebrauch ist weltweit fest etabliert, in einigen Kulturkreisen bereits seit Jahrhunderten, den wohl meisten Menschen gilt sie als höchst individuell und nur schwer nachahmbar, und ihr Eigner verwendet sie bewußt, willentlich und nahezu ausschließlich für biometrische Zwecke; dies im Unterschied zu anderen personenspezifischen Kennzeichen wie Gesichtsform oder Stimme, welche sich ohne größeren Aufwand mit versteckter Kamera und Mikrophon erfassen und zur Personenbestimmung mißbrauchen lassen. Eine Unterschrift sorgt für Rechtsverbindlichkeit bei allen Formen von Verträgen genauso wie bei Steuererklärungen, schriftlichen Zeugenaussagen vor Gericht, sowie in unzähligen anderen Situationen des privaten und öffentlichen Lebens, und die Fälschung einer Unterschrift kann als Straftatbestand gewertet werden. Es ist anzunehmen, daß im Laufe der Jahre weitere biometrische Merkmale eine gesetzliche Verankerung ähnlich der der Unterschrift finden werden. Bislang aber, und dies auch erst seit wenigen Jahren, wurde in Deutschland und in der EU lediglich die sog. *digitale Signatur* [Sch96] der handgeschriebenen Unterschrift rechtlich gleichgestellt [Str02], ohne daß allerdings damit deren flächendeckende Verbreitung auch nur annähernd einhergegangen wäre.

In großer Zahl werden Unterschriften u.a. im Bankwesen eingesetzt. Beispielsweise *autorisiert* ein Kunde durch seine Unterschrift die Bank, in seinem Namen eine Überweisung durchzuführen, und zugleich *authentifiziert* er sich mit ihr dem Geldinstitut gegenüber, denn der Sachbearbeiter kann sie bei Bedarf mit einer zum Zeitpunkt der Kontoeröffnung hinterlegten Unterschrift vergleichen. Deutsche Banken sind sogar zu einer derartigen Überprüfung bei *allen* Finanztransaktionen verpflichtet (Amtsgericht Frankfurt, AZ: 30 C 58/97–24), was allerdings aufgrund deren schierer Masse (7.5 Mrd. Schecks und Überweisungen im Jahr 1998 in Deutschland, bzw. 68 Mrd. in den USA 1999 [SL01]) praktisch nicht durchführbar ist. Das führt jedoch zu hohen Verlusten bei Kreditinstituten und deren Kunden: Der Schaden durch Scheck-Fälschungen beläuft sich in den USA nach Schätzungen der US-Bankenorganisation auf ca. 1 Mrd. Dollar jährlich [SL01].

Hingegen können mittels automatischer Verifikationsverfahren im besten Fall *sämtliche* Transaktionen überprüft werden, und das Bankpersonal muß nur noch in fraglichen Fällen zur Begutachtung einer Unterschrift hinzugezogen werden. Der Vergleichsprozeß wird auf diese Weise zudem „objektiviert“ in dem Sinne, daß er nicht mehr von der Willkür des jeweiligen Sachbearbeiters abhängig ist. Ein Bankangestellter wird es nämlich i.a. zu vermeiden suchen einen ehrlichen Kunden zu verprellen, und er wird daher im Zweifelsfall eher von der Legitimität der betrachteten Unterschrift ausgehen. Aber auch sonst hat sich gezeigt, daß zumindest Laien sich leicht von Unterschriftenfälschungen beirren lassen – bei ihnen liegt die Fehlerrate bei bis zu 50% (Zitierung in [Roh03]) – und es ist kaum anzunehmen, daß viele der in Banken tätigen Personen über die nötigen forensischen Kenntnisse verfügen.

Die automatische Unterschriftenverifikation hat es allerdings mit verschiedenen Schwierigkeiten zu tun, allen voran die häufig hohe Variabilität in den Unterschriften einer Person.

Dies ist prinzipiell auch ein Problem anderer nichtstatischer biometrischer Merkmale, beispielsweise der Stimme, allerdings wird die Sprechererkennung durch anerkannte Theorien über die Lauterzeugung im Vokaltrakt getragen, und man scheint recht gut darüber Bescheid zu wissen, nach welchen Kriterien die einzelnen Sprechakte vernünftigerweise zu analysieren und zu vergleichen sind [Mil91]. Vergleichbares im Bereich der Unterschriftenerzeugung und -Erkennung ist dem Autor dieser Arbeit bislang nicht begegnet. Die von Gutachtern verwendeten und in der gängigen Literatur (z.B. [Mic82]) beschriebenen Merkmale haben sich wohl eher traditionell entwickelt, und in den vorhandenen Arbeiten, welche sich mit der Entwicklung von Systemen zur automatischen Unterschriftenverifikation beschäftigen, präsentieren die Autoren häufig einen eigenen Satz von Analysemerkmalen.

Die vorliegende Arbeit wurde im Rahmen des BiSP-Projektes [BIS] entwickelt, einem u.a. vom Bundesforschungsministerium (BMBF) geförderten Gemeinschaftsunterfangen mehrerer Hochschulen und Fachhochschulen, welches sich mit der Erfassung und Analyse der Dynamik des Schreibvorgangs befaßt. Es wird hier ein eigenes Verfahren zur Unterschriftenverifikation vorgestellt, welches nicht das statische Schriftbild einer Unterschrift auswertet, sondern Informationen aus ihrem *Entstehungsprozeß*. Die in der Arbeit verwendeten Daten wurden mit einem an der FH Regensburg entwickelten Schreibstift, dem „BiSP-Pen", erfaßt, der hier ebenfalls vorgestellt werden soll.

Bei der Entwicklung des Systems wurde von der Hypothese ausgegangen, daß eine Unterschrift sich auf natürliche Weise in eine *Folge von Segmenten* zerlegen läßt, welche „inhaltliche Einheiten" darstellen. Die Segmentfolgen zweier Unterschriften der *gleichen* Person werden dabei, so die Annahme, einen hohen Grad an Gemeinsamkeit aufweisen, womit gemeint ist, daß sich an ähnlichen relativen Positionen in den Segmentsequenzen zumeist auch strukturell zueinander passende Segmente befinden sollten. Die Aufspaltung der Unterschriften wird, aus Mangel an geeigneten Theorien über den Aufbau von dynamisch erfaßten Unterschriften, in *adaptiver* Weise erfolgen, für das Auffinden der jeweils zueinander passenden Segmentpaare wird hingegen auf die bewährte Methode des *Dynamic-Time-Warpings* zurückgegriffen werden. Völlige Übereinstimmung zwischen zwei Segmenten ist indessen nie zu erwarten, und so wird es einer geeigneten *Codierung* der Segmente bedürfen, welche die Ähnlichkeit oder Verschiedenheit zweier Segmente hervorhebt. Die Verifikation wird im Übrigen nicht zwischen zwei einzelnen Unterschriften erfolgen, sondern zwischen einer Unterschrift und einem Unterschriften-*Modell* für eine Person. Es wird eine Hauptaufgabe dieser Arbeit sein, aus mehreren von einer Person stammenden Unterschriftenproben sozusagen deren „wahre" Segmentsequenz zusammen mit der *spezifischen Variabilität* für jedes der Segmente zu gewinnen. Die Unterschrift einer Person wird dann in aller Regel in weiten Teilen mit der zu ihr gehörenden Modellsequenz vereinbar sein; für Fremdunterschriften wird dies hingegen normalerweise nicht gelten. Die Modelle werden darüber hinaus so beschaffen sein, daß Anpassungen an im Laufe der Jahre stattfindende Veränderungen einer Unterschrift ohne großen Zusatzaufwand möglich sein werden.

Zum Aufbau der Arbeit: Kapitel 2 führt im ersten Abschnitt in das Gebiet der Unterschriftenverifikation ein und beschäftigt sich alsdann mit diversen Aspekten der Schreibdynamik (Abschnitte 2.2–2.4). Hier wird auch der BiSP-Pen beschrieben. Anschließend wird in Abschnitt 2.5 der generelle Aufbau eines Verifikationssystems dargestellt, zusammen mit grundlegenden Methoden und Verfahren, die für die Realisierung eines solchen Systems von Bedeutung sein können. In 2.6 wird schließlich der Frage nachgegangen, nach welchen Kriterien sich die Leistung eines Verifikationssystems messen und beurteilen läßt. Kapitel 3 beschäftigt sich mit den Eigenschaften der vom BiSP-Pen erfaßten Unterschriftendaten. Dabei wird sowohl das eigentliche Zeitsignal als aus dessen Frequenzgang untersucht. Die hier gemachten Beobachtungen werden grundlegend sein für die Entwicklung des neuen Unterschriftenverifikationssystems des Autors, wie es in Kapitel 4 vorgestellt wird. Der Beschreibung der allgemeinen Datenvorverarbeitung von „rohen" Unterschriftendaten in 4.1 folgt in 4.2 die Darstellung der Generierung eines Modells für die Unterschrift einer Person, das sog. *„Enrollment"*. Die so erzeugten Modelle werden wie bereits erwähnt aktualisierbar sein. Ein entsprechendes *Mo-*

dell-Update wird in 4.3 behandelt. In 4.4 wird dann der eigentliche Verifikationsprozeß, d.h. der Vergleich zwischen einer einzelnen Unterschrift und dem zuvor generierten Modell einer Person beschrieben. Anhand eines Datenbestandes bestehend aus mehreren tausend Originalunterschriften, welcher allen Teilnehmern des BiSP-Projektes zur Verfügung steht, wurde ein großangelegter Leistungstest für das hier entwickelte System durchgeführt. Die zentralen Ergebnisse dieses Tests werden in Kapitel 5 präsentiert. Eine abschließende Beurteilung der Arbeit zusammen mit einem Ausblick auf potentielle zukünftige Entwicklungs- und Forschungstätigkeiten liefert Kapitel 6.

Zum Schluß noch ein Hinweis zur Darstellung von Graphiken: Ein Großteil der in dieser Arbeit enthaltenen Abbildungen wurde farbig angelegt, ihr Inhalt ist aber aufgrund zusätzlicher struktureller Darstellungsunterschiede generell auch auf schwarzweißen Ausdrucken bzw. Fotokopien erkennbar. Der das Bild beschreibende Text enthält dann entsprechende Verweise im Stile von *„rot gepunktete Linie“* oder *„durchgezogene blaue Linie“*.

Kapitel 2

Grundlagen

In diesem Kapitel werden die für die gesamte Arbeit grundlegenden Aspekte der Unterschriftenverifikation präsentiert. Bestimmte Basiskenntnisse aus Analysis, Linearer Algebra, Stochastik und Algorithmik werden hier vorausgesetzt und ohne Zitierung verwendet; der Leser sei auf entsprechende Standardlehrwerke verwiesen, beispielsweise [Heu91], [SW90], [Fel68], [OW93].

2.1 Unterschriftenverifikation

Ziel der Unterschriftenverifikation ist es festzustellen, ob eine vorliegende Unterschriftenprobe, die sog. *„Query-Unterschrift“* oder einfach *„Query“*, tatsächlich von derjenigen Person angefertigt wurde, von der dies behauptet wird (die *„Originalperson“*). Zu diesem Zweck wird ein im Einzelfall noch näher zu spezifizierender Vergleich der Query mit einem vorhandenen *Unterschriften-Modell* durchgeführt, welches die Unterschrift der Originalperson repräsentiert. Das Modell kann dabei im einfachsten Fall eine früher hinterlegte Unterschriftenprobe der Originalperson sein, es sind aber auch komplexere Formen denkbar. Wir werden das Thema der Modellierung in 2.5 diskutieren.

In jedem Fall aber muß die *Authentizität* des Modells gewährleistet sein, d.h. die in das Modell eingehenden Informationen haben wirklich von der Originalperson zu stammen. Dies kann in der Praxis z.B. durch Vorlegen eines amtlichen Ausweises mit Lichtbild bei der Anfertigung der Unterschriften geschehen. Wir wollen jedoch in dieser Arbeit auf das Thema der *Authentifikation* nicht weiter eingehen.

2.1.1 Verifikation vs. Fälschungserkennung

Das oben charakterisierte „Verifikationsproblem“ ist nicht identisch mit der Frage, ob eine Query eine *Fälschung* darstellt. Eine Fälschungsüberprüfung kommt nämlich ggf. auch ohne ein Modell und den damit auszuführenden Vergleich aus: Es ist in der Kriminalistik durchaus üblich, Unterschriften auf sog. „Fälschungsartefakte“ hin zu untersuchen, d.h. Spuren irgendwelcher Art, die darauf hindeuten, daß es sich bei der überprüften Unterschrift um eine Fälschung handelt. Zum Beispiel können bestimmte Unregelmäßigkeiten in der Strichbeschaffenheit feststellbar sein, resultierend aus einem unterschiedlichen Verlauf der Ablage der Schreibpaste (Tinte, Graphit, o.ä.), was heute von forensischen Experten z.T. mit Rasterelektronenmikroskopen untersucht wird [SL01]. Solche „physikalisch-technischen Untersuchungen“ sind nicht Gegenstand dieser Arbeit.

2.1.2 Fälschungstypen

Wir werden vier verschiedenen Typen von Fälschungen unterscheiden:

1. *„Fremdunterschrift“*: Der Fälscher verwendet seine *eigene* Unterschrift als Fälschung der Unterschrift der Originalperson.

2. *„Blind-Fälschung“*: Der Fälscher schreibt den ihm bekannten *Namen* der Originalperson in seiner üblichen Handschrift, wobei er keine Kenntnisse über das Aussehen der Originalunterschrift besitzt.

3. *„Kopierende Fälschung“*: Dem Fälscher liegt eine Unterschriftenprobe der Originalperson vor, sodaß er eine Fälschung durch *Nachzeichnen* der Originalunterschrift erstellen kann.

4. *„Imitierende Fälschung“*: Hier hatte der Fälscher die Gelegenheit, die Originalperson während der Erstellung einer Unterschrift zu beobachten (z.B. durch Aufzeichnung mit einer Videokamera). Die Fälschung entsteht hier durch *Imitation des Schreibprozesses.*

Fremdunterschrift (Typ 1) und Blindfälschung (Typ 2) lassen sich als „naive“ Fälschungen zusammenfassen, denn der Fälscher erzeugt tatsächlich nur Proben seiner eigenen Handschrift, die mit der Unterschrift der Originalperson normalerweise nichts zu tun haben. Ein Verifizierer, auch ein Laie, wird sich in aller Regel von einer solchen Fälschung nicht täuschen lassen. Daß der Fälscher bei der Blindfälschung den Namen der Originalperson kennt, dürfte in der Praxis selten zu besseren Fälschungen führen, denn die Unterschrift einer Person ist normalerweise nicht einfach eine handschriftliche Folge von Buchstaben ihres Namens (vgl. 2.2). Dennoch werden zumindest die Fremdunterschriften häufig für die Evaluation von Verifikationssystemen eingesetzt, wie wir in 2.6 noch sehen werden.

Kopierende (Typ 3) und imitierende (Typ 4) Fälschungen zielen auf die Reproduktion der Unterschrift der Originalperson ab: Die kopierende Fälschung versucht das Schriftbild, die imitierende Fälschung den Erzeugungsprozeß hinter dem Schriftbild nachzubilden. Diese Fälschungstypen lassen sich als *„qualifizierte“* Fälschungen bezeichnen. Die kopierende Fälschung stellt wahrscheinlich den häufigsten Fälschungstyp dar, denn sie läßt sich recht leicht bewerkstelligen und ist zumindest bei flüchtigem Schriftbildvergleich vom Original nur schwer zu unterscheiden; dem forensischen Experten stehen hier aber gemäß 2.1.1 moderne Methoden und Technologien zur Fälschungserkennung zur Verfügung.

Die zum Erreichen guter Ergebnisse sehr viel mehr Geschick und Aufwand erfordernde imitierende Fälschung ist eine Antwort auf die vermehrt auftretenden Methoden zur Analyse *dynamischer* und *pseudodynamischer* Merkmale, auf die wir in 2.3 zu sprechen kommen werden, und die den wesentlichen Aspekt dieser Arbeit ausmachen. Eine erste Einschätzung der Schwierigkeit einer solchen Fälschung wird uns die Betrachtung der Eigenschaften von Handunterschriften im nächsten Abschnitt ermöglichen.

2.2 Physiologische Aspekte

Die Individualität von Handschriften scheint stark ausgeprägt zu sein. Schmidt und Lenz berichten in [SL01], daß zwar bei eineiigen Zwillingen gelegentlich ähnlich aussehende Handschriften auftreten, diese aber bei genauerer Untersuchung stets wohlunterscheidbar sind. Nach Kenntnis der beiden Forscher wurde in der Fachliteratur noch kein Fall von „Doppelgängerhandschrift“ beschrieben, d.h. das Phänomen, daß zwei verschiedene Menschen eine auch für Experten nicht zu unterscheidende Handschrift besitzen. Die *Unterschrift* einer Person entwickelt sich zunächst aus der kindlichen Handschrift, wird dann i.d.R. aber im Laufe der Jugend bewußt in Richtung persönlichen Stils abgewandelt, sodaß im Vergleich zur Handschrift eine mindestens genauso niedrige, wenn nicht noch geringere Wahrscheinlichkeit für zwei Personen mit gleicher Unterschrift bestehen dürfte.

Physiologisch gehen beim Schreiben Einflüsse von Finger-, Handgelenk- und Armbewegungen in das Schriftbild ein, und es wirken, wie man durch moderne bildgebende Verfahren weiß, verschiedene Hirnareale zusammen, so der motorische und assoziative Cortex, mehrere tiefe Großhirnkerne und das Kleinhirn. Die Erzeugung der einzelnen Komponenten einer

Handschrift erfolgt i.d.R. so schnell, daß eine Korrektur auf der Grundlage optischer Rückmeldung i.a. nicht möglich ist: «Die Information über die Erzeugung z.B. eines Buchstabens muß demzufolge komplett [im Gehirn] gespeichert sein.» [SL01, 10.2.2].

Nun ist eine Unterschrift keine diskrete Buchstabenfolge, sondern, zumindest vom subjektiven Eindruck her, eine zusammengehörende, fließende und zügig ausgeführte Bewegung. Der Autor dieser Arbeit hält es daher für denkbar, daß für Unterschriften das Gleiche gelten könnte wie für Buchstaben: nämlich daß ihr Erstellungsprozeß *als Ganzes* neuronal repräsentiert ist. Für das Fälschen von Unterschriften hätte das eine wichtige Konsequenz: Ein vom Original schwer zu unterscheidendes Nachahmen der Unterschrift einer anderen Person (Fälschungstyp 4) wäre, wenn überhaupt, nur durch einen aufwendigen und langwierigen Trainingsprozeß zu erreichen, und sollte sonstigenfalls leicht an der Andersartigkeit des Bewegungsablaufes erkennbar sein. Somit dürften die allermeisten Fälschungen anhand der Schreibdynamik identifizierbar sein, womit ein automatisches System seinen wesentlichen Zweck bereits erfüllt hätte. Wir werden uns daher im folgenden genauer mit der Unterschriftendynamik beschäftigen.

2.3 Unterschriftendynamik

Herkömmlicherweise liegt die zu überprüfende Unterschrift in „statischer“ Form als Schriftbild vor. Wie in 2.1.1 berichtet wurde, gibt dies menschlichen Verifizierern zumindest prinzipiell die Gelegenheit, Dinge wie z.B. Besonderheiten bei der Schreibpastenablage zu vergleichen. Ein Computersystem zur statischen oder *„Offline“*-Analyse erhält hingegen normalerweise nur digitale Bildaufnahmen der Unterschriften als Eingabe. Handelt es sich dabei um ein reines Schwarzweißbild, erzeugt durch einen einfachen Scanner mit grober Auflösung, ist nicht zu erwarten, daß ein automatischer Verifizierer kopierende Fälschungen (Typ 3) besser in der Lage ist zu erkennen, als sein menschliches Pendant.

2.3.1 Pseudodynamische Merkmale

Die Variationen in der Strichbeschaffenheit können sich, wenn auch in beschränktem Maße, in Form von Helligkeitsschwankungen bemerkbar machen, welche dann auch auf Graustufenbildern hoher Qualität sichtbar sind. Darauf basierend lassen sich grobe Vermutungen über *dynamische* Aspekte wie Schreibdruck, Schreibgeschwindigkeit und Stiftneigung an jeder Stelle des Schriftbildes anstellen. Ein automatisches Verifikationssystem könnte über Heuristiken verfügen, die aus einem solchen Schriftbild derlei *pseudodynamische* Merkmale extrahiert und auf deren Grundlage einen Vergleich durchführt. Damit ließe sich eine Typ 3-Fälschung vermutlich leichter identifizieren als im zuvor betrachteten Szenario.

Sofern also die Qualität des Ausgangsdatenmaterials hinreichend gut ist, bietet dieser Bereich der pseudodynamischen Merkmale ein großes und vielversprechend erscheinendes Potential für automatische Verifikationssysteme, und tatsächlich verfolgen einige kommerzielle Anbieter diesen Weg, z.B. die Firma SOFTPRO (`info@softpro.de`). Der große praktische Vorteil dabei ist, daß keine wesentliche Änderung am Erfassungsprozeß nötig ist: Die Person unterschreibt wie gewohnt mit einem Stift ihrer Wahl,[1] und es ist auch keine ungewöhnliche zusätzliche Hardware zum Digitalisieren nötig, denn man braucht lediglich einen guten Scanner und ggf. Software mit Standardmethoden zur rudimentären Aufbereitung der Daten für den später stattfindenden Analyseprozeß. Insbesondere lassen sich auf diese Weise auch Unterschriften auf archivierten Dokumenten nachträglich erfassen und vergleichen.

[1] Man sollte sich allerdings darüber im Klaren sein, daß die Stricheigenschaften auch von dem verwendeten Stifttyp, der Schreibunterlage und dem Schriftträger abhängen. So gibt es allein über 3000 Papiersorten in Europa, die sich z.B. hinsichtlich Oberflächenstruktur und Absorbtionsverhalten unterscheiden [SL01].

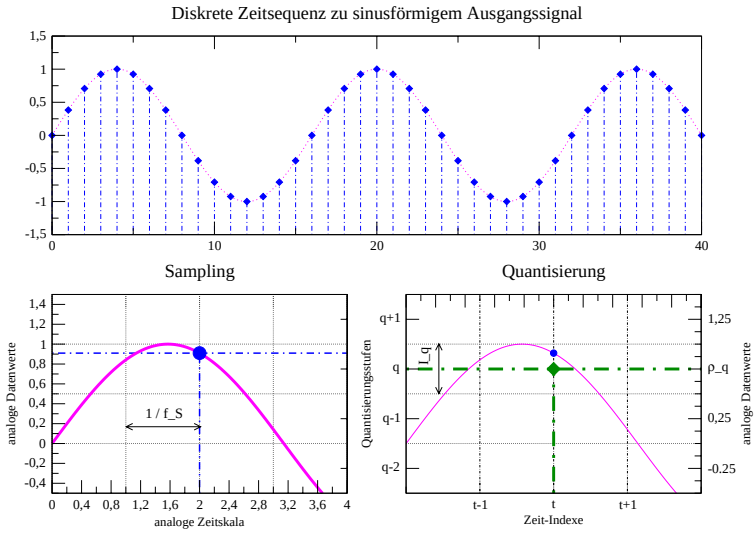

Abbildung 2.1: Zeitsequenzen, Sampling und Quantisierung

2.3.2 Echte dynamische Merkmale

Anstatt Helligkeitsunterschiede im Schriftbild als dynamische Merkmale zu interpretieren, kann man auch versuchen solche Merkmale direkt zu erfassen. Als Beispiele seien genannt:

- Ortskurve, Geschwindigkeits- und Beschleunigungsverlauf
- Anpreß- und Schreibdruck
- Stiftneigung
- Schreibgeräusche

Für die genannten Merkmale gibt es mittlerweile verschiedene Erfassungsgeräte (vgl. 2.4).

2.3.2.1 Darstellung dynamisch erfaßter Unterschriften

Wir repräsentieren dynamisch erfaßte Merkmale als *Zeitsequenzen* $(x_t)_{0\leq t<T}$ bestehend aus T-vielen *Samples* x_t, die zu diskreten, durch die Indexe t benannten Zeitpunkten aufgezeichnet wurden (vgl. Abb. 2.1, oben). Dabei ist der Zeitpunkt t nur als Zeit*index* zu verstehen: Das Erfassungsgerät zeichnet in bestimmten, üblicherweise gleichmäßigen Zeitabständen jeweils ein einzelnes Sample auf (*„äquidistantes“* Sampling), wobei die Anzahl solcher Erfassungen pro Sekunde als *Samplingrate* f_S bezeichnet wird (Abb. u.l.). Um den physikalischen Zeitpunkt τ eines Samples x_t gemessen in Sekunden zu berechnen, muß man daher f_S kennen: $\tau = t \cdot 1/f_S$.

Bei den Samples x_t handelt es sich in den von uns betrachteten Fällen um *digitalisierte* Werte: Nach der Erfassung z.B. eines physikalischen Druckwertes $\tilde{x}_t$ durch einen Drucksensor wird $\tilde{x}_t$ durch einen *A/D-Wandler* eine natürliche Zahl $q_{\tilde{x}_t} \in \mathbb{N}_0$, die sog. *Quantisierungsstufe* von $\tilde{x}_t$, zugeordnet. Ein n-Bit A/D-Wandler zerlegt einen bestimmten physikalischen Wertebereich B, der von den Eigenschaften des verwendeten Sensors abhängig ist und gelegentlich in bestimmten Grenzen justiert werden kann, in 2^n nicht notwendigerweise gleichlange Teilintervalle I_q, $0 \leq q < 2^n$ (im Falle des BiSP-Pens in 2.4.1 handelt es sich z.B. um den Spannungsbereich $B = [0, +10]V$). Stammt der Druckwert $\tilde{x}_t$ aus I_q, so gilt $q_{\tilde{x}_t} = q$. Das endgültige Sample x_t erhält i.d.R. wieder den Spannungswert eines *Repräsentanten* $\rho_{q_{\tilde{x}_t}} \in I_{q_{\tilde{x}_t}}$, wobei für ein ρ_q z.B. der Mittelpunkt des Intervalls I_q verwendet werden kann (Abb. u.r.). Die Fehlersequenz $(\tilde{x}_t - \rho_{q_{\tilde{x}_t}})_{0\leq t<T}$ zwischen den erfaßten Signalwerten $\tilde{x}_t$ und den zugehörigen Repräsentanten $\rho_{q_{\tilde{x}_t}}$ wird als *„Quantisierungsrauschen“* bezeichnet [Smi99].

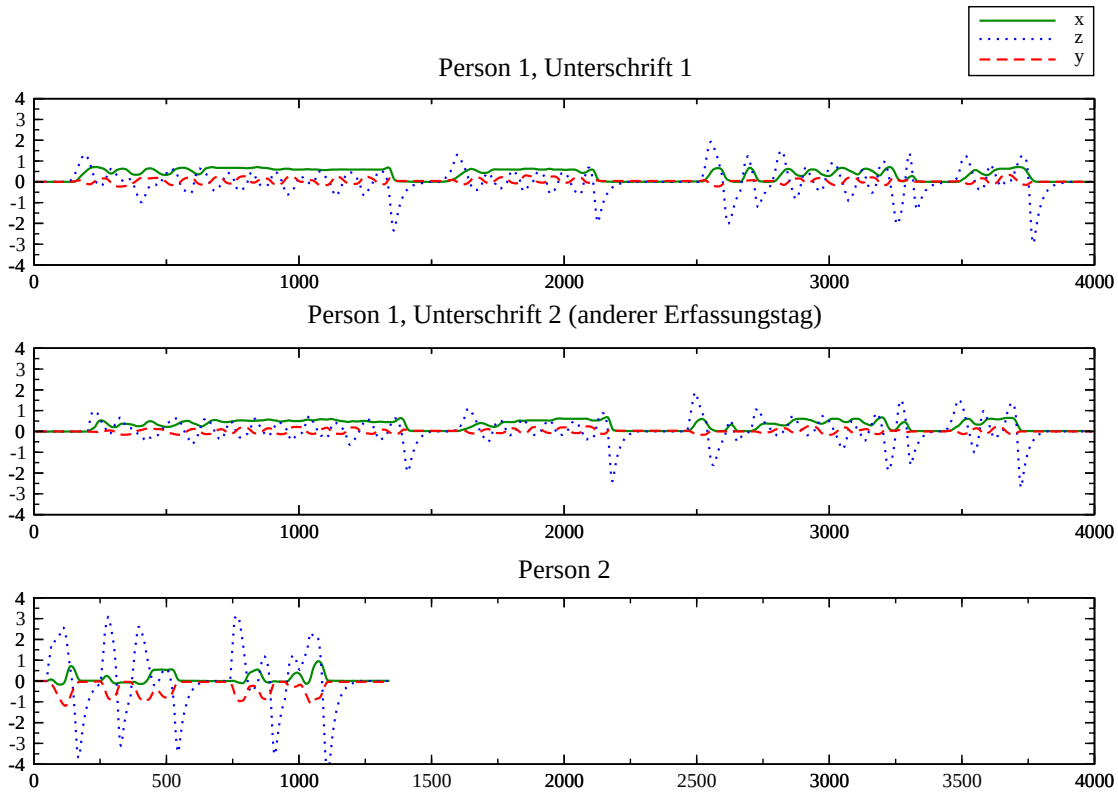

Abbildung 2.2: Dynamik der Unterschrift bei gleicher und verschiedener Person

Werden n-viele Merkmale gleichzeitig erfaßt, so lassen sich alle n zum Zeitpunkt t registrierten Samples $x_{j,t}$ zu einem Sample-Vektor $\boldsymbol{x}_t = (x_{1,t}, \ldots, x_{n,t})^\top$ zusammenfassen. Die Zeitsequenz $(\boldsymbol{x}_t)_{0\leq t<T}$ besteht dann aus den *Komponentensequenzen* $x_j := (x_{j,t})_{0\leq t<T}$ für das j-te Merkmal. Die vektorielle Repräsentation dient in erster Linie der kompakten schreibtechnischen Darstellung und der mathematischen Behandlung. In graphischen Präsentationen wird man dagegen eher sämtliche Komponentensequenzen als einzelne Zeitkurven im Stile von Abb. 2.1, oben, in einem gemeinsamen Funktionsgraphen auftragen. Als Beispiel kann Abb. 2.2 im anschließenden Abschnitt dienen, wo pro Diagramm jeweils drei sich überlagernde eindimensionale Zeitsignale zu sehen sind.

2.3.2.2 Vergleichspotential dynamisch erfaßter Unterschriften

Die Brauchbarkeit dynamisch erfaßter Unterschriften für die Verifikationsaufgabe soll an einem ersten Beispiel demonstriert werden. In Abb. 2.2 werden drei mit dem BiSP-Pen erfaßte Unterschriften einander gegenübergestellt. In horizontaler Richtung ist die *Zeit* aufgetragen, wobei die Samplingrate f_S=500Hz beträgt. Entlang der Ordinate ist der *Schreibdruck* aufgetragen, die Achsenwerte geben dabei die gemessene Spannung an. Jedes Diagramm enthält die drei Zeitreihen (x_t), (y_t) und (z_t), welche den Druck in Schreibrichtung, senkrecht dazu und in die Tischebene hinein bezeichnen. Es handelt sich um reale, aus dem in Kapitel 3 beschriebenen Feldversuch stammende Daten. Dabei wurden die oberen beiden Zeitsequenzen von der *gleichen* Person an zwei unterschiedlichen Tagen aufgezeichnet. Man kann gut erkennen, daß sich beide Unterschriften recht ähnlich sehen. Es gibt allerdings zahlreiche Unterschiede im Detail. Die dritte Unterschrift wurde von einer zweiten Person erfaßt. Sie unterscheidet sich von den Unterschriften der ersten Person sichtbar deutlich in Form und Schreibdauer. Der Vergleich anderer Beispiele kommt zu ähnlich deutlichen Ergebnissen.

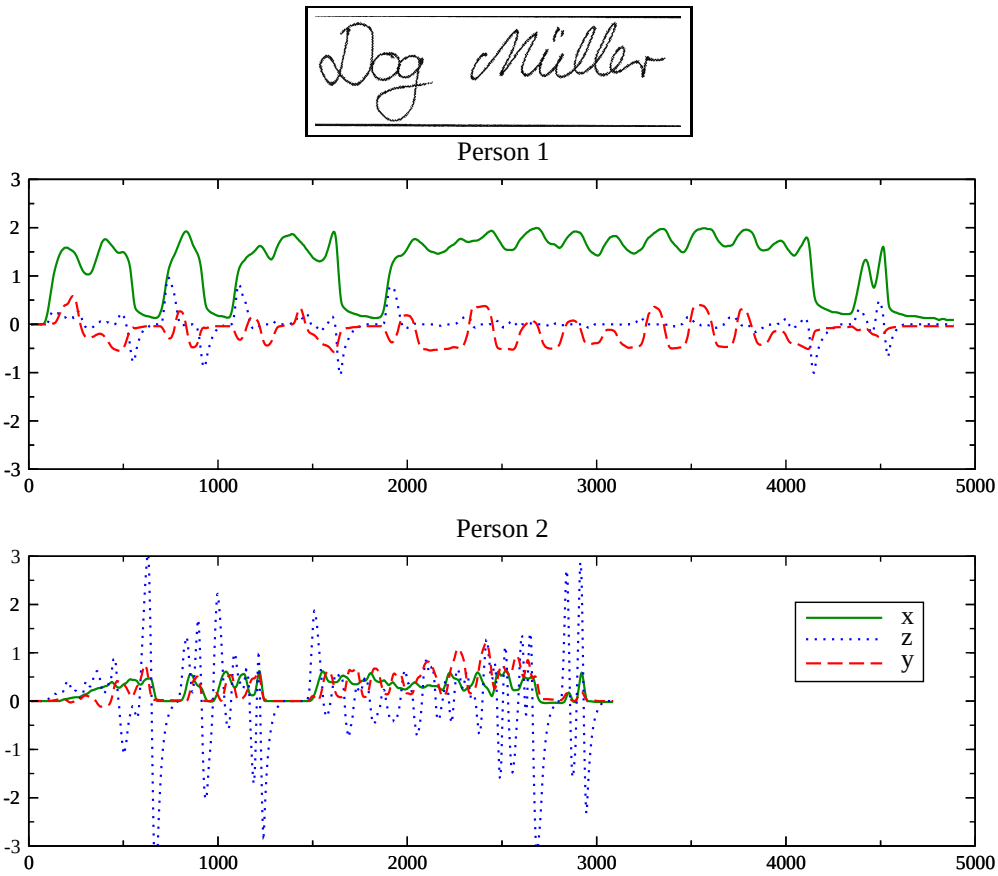

Abbildung 2.3: Schriftbildkopien verschiedener Fälscher der gleichen Unterschrift

2.3.2.3 Vor- und Nachteile der dynamischen Erfassung

Aus den Sequenzen bestimmter dynamischer Merkmale läßt sich im Prinzip das Schriftbild rekonstruieren, so im Falle der *Ortskurve* $(x_t, y_t)^\top_{0\leq t<T}$, bei der das Schriftbild gerade ihrer *Spur*

$$\left\{(\xi,\eta)^\top \in \mathbb{R}^2 \mid \exists t \in \{0,\ldots,T-1\} : \xi = x_t \wedge \eta = y_t\right\}$$

entspricht. Sogesehen umfaßt die Analyse der Schreibdynamik („Online-Analyse“) die Offline-Analyse. Man überlege sich zudem, daß es zu einem Schriftbild unterschiedliche Zeitsequenzen geben kann: Für die in Abb. 2.3 (oben) dargestellte Unterschrift *„Dog Müller“* läßt sich ohne zusätzliches Wissen nicht entscheiden, ob die Punkte über dem ‘ü’ noch während des Schreibens des Buchstabens ‘u’ oder möglicherweise erst am Ende des kompletten Namens gesetzt wurden.

In der Praxis zeigen sich darüberhinaus i.d.R. auch Unterschiede hinsichtlich der Charakteristik der Schreibdynamik: In Abb. 2.3 (unten) sind die Versuche zweier Personen, eine Schriftbildkopie des Schriftzuges *„Dog Müller“* zu erstellen, einander gegenübergestellt. Die beiden mit dem BiSP-Pen generierten Druck-Zeitreihen unterscheiden sich augenscheinlich stark voneinander. Wir erhalten damit eine Bestätigung der in 2.2 geäußerten Vermutung, daß Schriftbildkopien (Fälschungstyp 3) sich bei einer Online-Analyse eher nicht als gute Fälschungen erweisen werden. Man beachte in diesem Zusammenhang, daß eine gute Fälschung nicht nur eine gute Nachahmung des Schreibstils des Unterschrifteneigners, sondern auch die erfolgreiche *Unterdrückung der charakteristischen Eigenanteile* voraussetzt. Das dies normalerweise nicht ohne weiteres gelingt, wird durch die Beobachtung gestützt, daß sich die Drucksequenzen zweier Kopierversuche, wenn sie beide von der *gleichen* Person erstellt wurden, stark ähnlich sehen (nicht dargestellt).

Als wesentlicher Nachteil bei der Verwendung echter dynamischer Merkmale muß die Notwendigkeit unüblicher Erfassungshardware genannt werden. Es ist in jedem Fall ein besonderes Schreibgerät zu benutzen, und darüber hinausgehend benötigen manche Systeme auch spezielles Papier oder sogar eine komplette Infrastruktur, z.B. ein GPS-System, über welches die absolute Position des Schreibgerätes ermittelt werden kann. Beispiele für solche Systeme werden in [Roh03] beschrieben.

2.3.2.4 Anwendbarkeit dynamisch erfaßter Unterschriften

Einige konkrete Anwendungen der Online-Analyse existieren bereits oder sind zumindest vorstellbar. So verwendet der Paketbringdienst UPS™ seit Jahren ein elektronisches Spezialgerät, mit dem der Empfänger eine Lieferung per Unterschrift zu bestätigen hat. Vor allem *PDAs* und der *Tablet-PC*® sind explizit auf die Eingabe per Stift ausgelegt; für sie erscheint eine unterschriftenbasierte Zugangskontrolle (Entsperren des PDAs bzw. Login am Tablet-PC) naheliegender als ein paßwortbasiertes System.

Eine interessante Nutzungsmöglichkeit stellt die *handschriftliche Unterzeichnung elektronischer Dokumente* dar. Ein mit einem Textprozessor erstelltes Dokument könnte mit Hilfe eines Tablets (oder gewöhnungsbedürftiger mit der Maus) eine Unterschrift erhalten, deren Schriftbild zur optischen Überprüfung, die das Schriftbild repräsentierende Zeitsequenz zur computerbasierten Verifikation dienen könnte. Zum Beispiel steht für das Unterzeichnen von PDF-Dateien mittels Wacom®-Tablet (`www.wacom.com`) mit SigPDF des Herstellers Signature Perfekt (`www.signature-perfect.com`) ein entsprechendes Plugin für den Acrobat Reader® (`www.adobe.com`) zur Verfügung.

Das in 2.3.2.3 genannte Problem der speziellen Erfassungshardware wäre zumindest immer dann ohne großen Belang, wenn die Unterzeichnung an einem ganz bestimmten Ort und unter Einhaltung eines festen Protokolls zu erfolgen hat. Als Beispiel könnte man sich das Eingangsportal zu einem sicherheitsrelevanten Platz wie dem Haupttresor einer Bank vorstellen. Bei der notariellen Beglaubigung eines Dokumentes oder auch bei der Vergabe eines teuren Kredits wird der Kunde im Büro des Sachbearbeiters sitzen, wo ein entsprechendes Schreibgerät samt Infrastruktur bereitstehen könnte. Auch die Akzeptanz des Gerätes durch den Unterzeichner dürfte unter solch streng geregelten Gegebenheiten höher sein, als z.B. beim Unterschreiben einer einfachen Banküberweisung.

2.4 Erfassungssysteme

Die Erfassung der Unterschriftendynamik geschieht, wie schon mehrfach betont, mit einem speziellen Gerät. Ein solches Erfassungsgerät sollte idealerweise Form und Verwendungsweise eines herkömmlichen Schreibstiftes besitzen, um das gewohnte Schreibgefühl nicht zu beeinträchtigen. Aber auch Tablets, wie sie z.B. gerne von Computer-Graphikern zur Dateneingabe verwendet werden, eignen sich zu diesem Zweck. Wir werden uns hier auf die Präsentation des an der FH Regensburg entwickelten kugelschreiberähnlichen „BiSP-Pens“ beschränken, denn die von diesem Gerät aufgezeichneten Daten werden in der vorliegenden Arbeit verwendet. Neben dem BiSP-Pen gibt es eine Reihe weiterer stiftartiger Erfassungssysteme, sogenannte „Smartpens“, welche alle nach unterschiedlichen Methoden arbeiten. In [Roh03] werden verschiedene solcher Schreibgeräte gemeinsam mit anderen für die Erfassung von Unterschriften geeigneten Systemen beschrieben.

2.4.1 Der BiSP-Pen: Architektur und Funktionsweise

Die Bezeichnung „BiSP“ steht für *„Biometrical Smart Pen“*. Damit ist ein als multimodales biometrisches Erfassungssystem geplanter Schreibstift gemeint, der sowohl von seinem Aussehen als auch von seinem Gebrauch her einem Kugelschreiber gleichen soll, und dazu

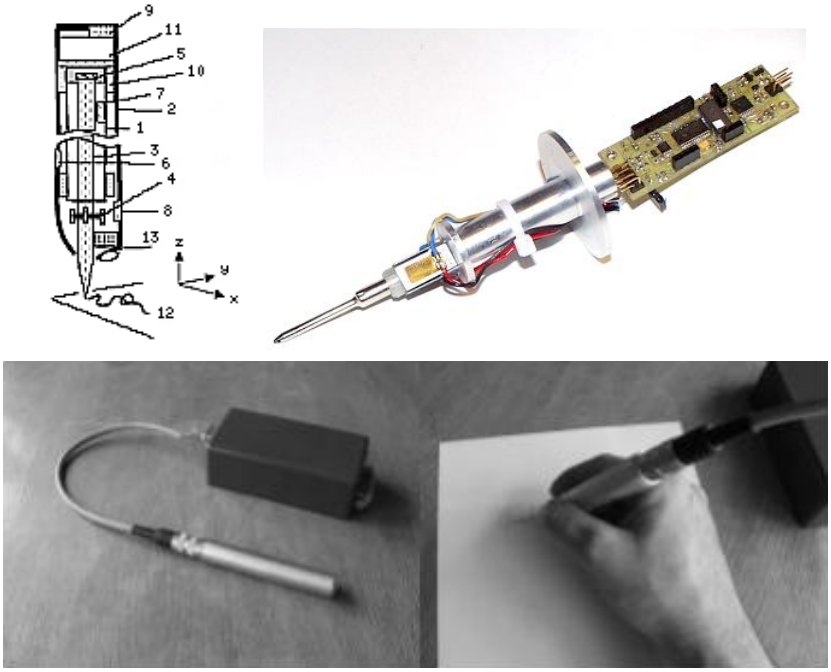

Abbildung 2.4: **Der BiSP-Pen.** links oben: schematische Darstellung des BiSP-Pens (aus [HKS03]), unten: MechPen (alte Version) (aus [MRMK02]), rechts oben: interner Aufbau des MechPens (aus [HKS03]).

insbesondere mit einer herkömmlichen Metallmine ausgestattet sein wird. In seiner endgültigen Form sollen sämtliche zu Beginn von 2.3.2 als Beispiele genannten biometrischen Merkmale während des Schreibens erfaßt werden. Zum Zeitpunkt der Niederschrift dieser Arbeit existieren nach Kenntnis des Autors hierzu vier Prototypen [HKS03][ŠK03][HKS04], deren Funktionalitäten später zu dem in Abb. 2.4, links, skizzierten System integriert werden sollen:

- *OptoPen:* Dieser Stift dient der Erfassung der x(t)- und y(t)-Position zum Zeitpunkt t, d.h. mit ihm läßt sich die *Ortskurve*, und daraus ableitbar die *Schreibgeschwindigkeit* und *Beschleunigung* des Stiftes erfassen. Die Funktionsweise ähnelt derjenigen einer optischen Maus. In der Skizze ist der optische Sensor auf Höhe des Austritts der Mine aus dem Stift (13) angebracht.
- *MechPen:* Der MechPen (Abb. 2.4, unten und oben rechts) besitzt vier Dehnungsmeßstreifen seitlich entlang der Mine (in der Skizze bei 4), mit denen der *Schreibdruck* in x- und y-Richtung erfaßt wird. Am oberen Minenende (5) befindet sich ein Piezo-Element, mit dem die zeitliche Änderung des *Anpreßdrucks* in z-Richtung (senkrecht zur Tischebene) gemessen wird. Die Samplingrate des Stiftes beträgt 500Hz. Die vom Stift erfaßten Signale werden mit einem 10 Bit A/D-Wandler äquidistant quantisiert. Der Dynamikumfang der Sensoren beträgt 5V in jeweils beiden Druckrichtungen bei einer Empfindlichkeit von 2V pro Newton. Es sei erwähnt, daß es mittlerweile eine neue Version des MechPens (nicht abgebildet) gibt.
- *TiltPen:* Der TiltPen erfaßt die Stiftneigung in x- und y-Richtung mittels elektrolytischer Neigungssensoren (nicht in der Skizze dargestellt). Die Samplingrate beträgt 100Hz. Die Neigungsempfindlichkeit wird mit 55mV pro Grad angegeben.
- *MicPen:* Der MicPen erfaßt Schreibgeräusche über Gehäuseresonanzen und elastische Schwingungen der Mine. Ein im Innern des Stiftes angebrachtes Mikrophon (7) ist gegenüber Hintergrundgeräuschen akustisch abgeschirmt (10).

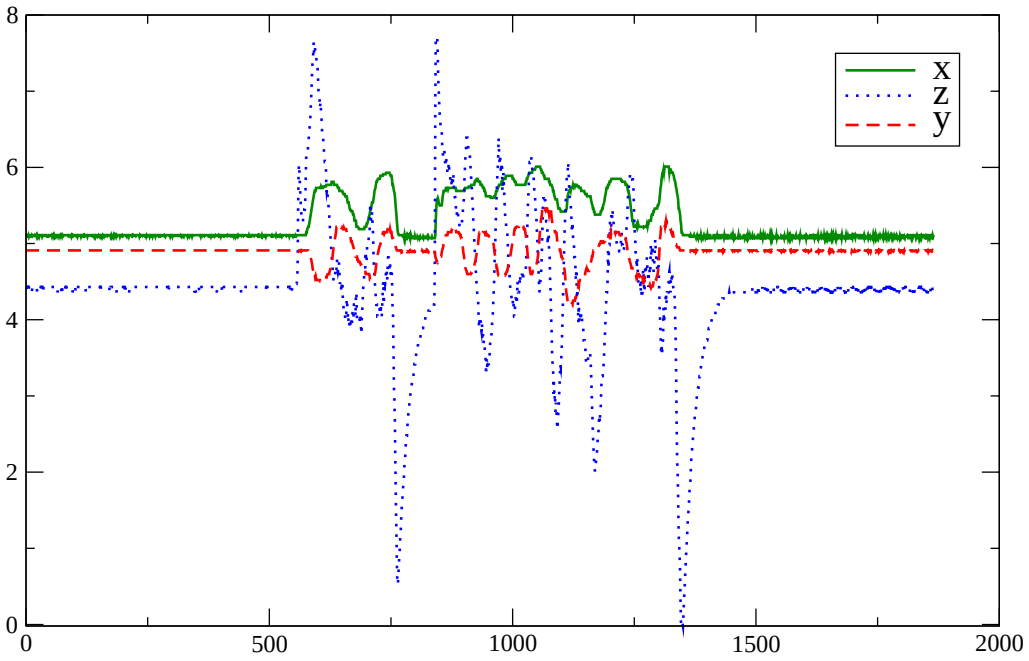

Abbildung 2.5: Beispiel einer mit dem BiSP-Pen erfaßten Unterschrift

Die in dieser Arbeit analysierten Daten wurden mit dem (alten) MechPen erfaßt. Dieser verfügt wie oben dargestellt ausschließlich über Drucksensoren für die x-, y- und z-Richtung. Da wir keinen anderen der vorgestellten Prototypen verwenden, werden wir im weiteren Verlauf stets „BiSP-Pen“ sagen, wenn eigentlich der alte MechPen gemeint ist.

Ein Beispiel für ein mit dem BiSP-Pen erfaßtes Unterschriftensignal stellt Abb. 2.5 dar. Alle drei Drucksignale sind gemeinsam im gleichen Schaubild eingezeichnet. Die Signale nehmen dort Amplitudenwerte im Bereich 0–10V bei einem Basislevel um 5V herum an. Eine detaillierte Betrachtung der mit dem BiSP-Pen erzeugten Unterschriftendaten wird in Kapitel 3 vorgenommen werden.

2.5 Der Modellierungs- und Verifikationsprozeß

Dieser Abschnitt beschreibt die üblicherweise in einem System zur dynamischen Unterschriftenverifikation vorkommenden Prozeßstufen. Es wird dabei exemplarisch auf in der Literatur beschriebene Systeme verwiesen werden, wobei wir uns auf wenige Beispiele beschränken werden. Für einen weitergehenden Überblick über bestehende Ansätze sowie eine umfangreiche Literaturliste sei auf [Roh03] verwiesen. Eine Liste existierender kommerzieller Systeme enthält [Sch04].

2.5.1 Genereller Aufbau eines Verifikationssystems

Abb. 2.6 zeigt den generellen Aufbau eines Verifikationssystems. Der eigentlichen Verifikation (obere Bildhälfte) geht einmalig die Phase des sog. *„Enrollments“* (unten im Bild dargestellt) voraus, bei dem für eine Person ein *Unterschriften-Modell* aus authentifizierten *Trainingsunterschriften* generiert und zusammen mit für die Verifikation notwendigen personenspezifischen Daten in einer Datenbank abgelegt wird. Wir werden uns mit der Modellierung in 2.5.4 beschäftigen.

Für alle vom verwendeten Erfassungsgerät erzeugten Datensequenzen, ob sie als Referenzdaten in das Modell einfließen oder eine zu überprüfende Queryunterschrift darstellen, sind folgende wesentlichen Prozeßstufen durchzuführen:

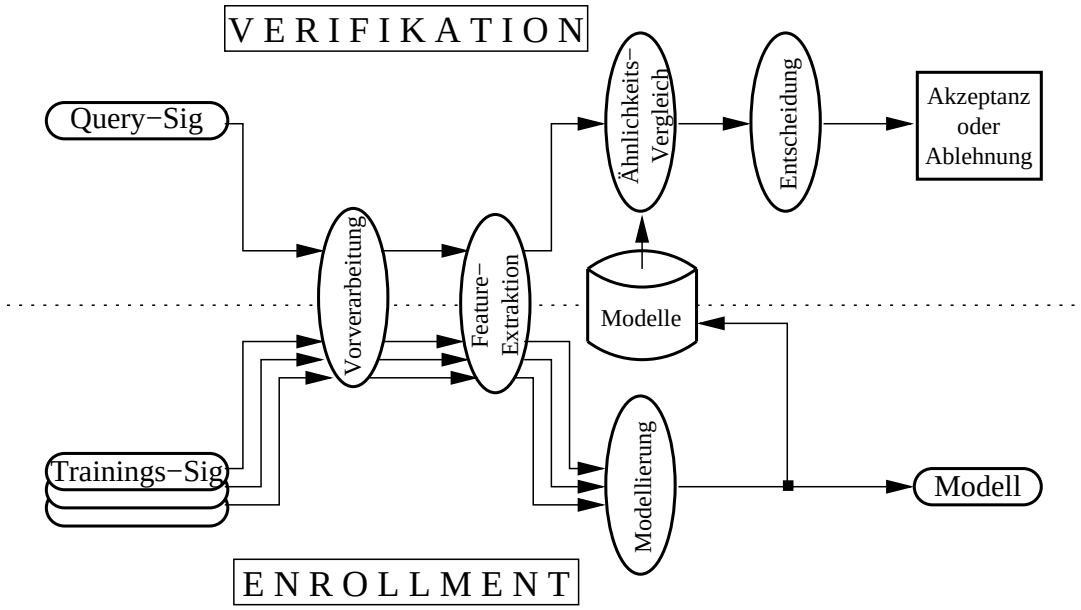

Abbildung 2.6: Genereller Aufbau eines Unterschriften-Verifikationssystems

1. *Datenvorverarbeitung:* Das vom Erfassungsgerät stammende Datensignal wird von störenden Einflüssen befreit und für den späteren Vergleich bzw. für die Modellierung aufbereitet (2.5.2).

2. *Feature-Extraktion:* Jede Datensequenz wird auf das Vorhandensein bestimmter Merkmale („Features“) hin untersucht (2.5.3). Vergleiche werden danach anhand dieser Merkmalsausprägungen durchgeführt, nicht mehr basierend auf den ursprünglichen Datensequenzen.

Die Verifikation findet alsdann in Form eines *Ähnlichkeitsvergleiches* (2.5.5) zwischen der zu überprüfenden *Query-Unterschrift* und dem Modell derjenigen Person statt, von der die Query angeblich stammt. Basierend auf dem ermittelten Ähnlichkeitswert ist anschließend eine *Entscheidung* darüber zu treffen, ob die Query zu akzeptieren oder abzulehnen ist.

2.5.2 Datenvorverarbeitung

Die vom Erfassungsgerät gelieferten „rohen“ Datenzeitreihen müssen zunächst in einer Weise aufbereitet werden, daß ein Mindestmaß an Vergleichbarkeit zwischen zwei Unterschriften gewährleistet werden kann. Es werden hier nur die üblicherweise bei dynamischen Unterschriftendaten auftretenden Probleme in allgemeiner Weise diskutiert. Im Einzelfall können die hier besprochenen Punkte in variierter Form auftreten, oder zusätzliche Probleme hinzukommen.

2.5.2.1 Signalentzerrung und Beseitigung von Störsignalen

Es ist davon auszugehen, daß das eigentliche Nutzsignal nach der Aufzeichnung in verzerrter und von Störsignalen überlagerter Form vorliegt. Es werden in diesem Abschnitt allgemeine Methoden für die Untersuchung von Zeitsequenzen auf Störungen und Verzerrungen hin sowie zu deren Beseitigung angegeben. Der Einfachheit halber werden nur *eindimensionale* Zeitsequenzen betrachtet werden.

2.5.2.1.1 Die Diskrete Fourier Transformation (DFT) Wir gehen in der folgenden Erörterung von Zeitreihen $(x_t)_{0\leq t<T}$ aus, deren Länge T *gerade* sei.[2] Die Koeffizienten der *Diskreten Fourier Transformation (DFT)* besitzen die Form

$$\begin{aligned}\alpha_k &:= \sum_{0\leq t<T} x_t \cdot \cos(k \cdot 2\pi t/T)\,, \quad 0 \leq k = T/2 \\ \beta_k &:= -\sum_{0\leq t<T} x_t \cdot \sin(k \cdot 2\pi t/T)\,, \quad 0 \leq k = T/2\end{aligned} \qquad \textit{(DFT-Koeffizienten)} \qquad (2.1)$$

Formal wird durch eine DFT also ein Vektor $\boldsymbol{x}$ der Länge T auf zwei Vektoren $\boldsymbol{\alpha}$ und $\boldsymbol{\beta}$ der Längen $T/2+1$ abgebildet. Diese Transformation ist umkehrbar, d.h. aus den DFT-Koeffizienten (α_k) und (β_k) läßt sich das Ausgangssignal (x_t) rekonstruieren (*Inverse DFT*):

$$x_t = \sum_{k\geq 0}^{T/2} \left[\alpha_k^* \cdot \cos(t \cdot 2\pi k/T) + \beta_k^* \cdot \sin(t \cdot 2\pi k/T)\right]\,,\ 0 \leq t < T \qquad \textit{(IDFT)} \qquad (2.2)$$

Dabei treten folgende normierte Versionen der DFT-Koeffizienten auf:

$$\alpha_k^* := \begin{cases} \frac{\alpha_0}{T} & : k = 0 \\ \frac{\alpha_k}{T/2} & : 1 \leq k \leq T/2-1 \\ \frac{\alpha_{T/2}}{T} & : k = T/2 \end{cases} \qquad \beta_k^* := -\frac{\beta_k}{T/2}$$

Anschaulich läßt sich (2.2) so deuten, daß eine Zeitsequenz als *additive Überlagerung sinusförmiger Schwingungen* mit bestimmten festen Frequenzen darstellbar ist, wobei die DFT-Koeffizienten i.w. (bis auf gewisse Normierungen) den *Amplituden* dieser Kreisfunktionen entsprechen. Im Prinzip benötigt man für die Darstellung aller möglichen endlichen Zeitreihen Cosinusfunktionen mit *beliebiger Phasenverschiebung*. Aus den trigonometrischen Additionstheoremen erhält man jedoch für eine feste Verschiebungsphase φ bei variablem Winkel ω die Beziehung

$$\begin{aligned}\cos(\omega+\varphi) &= \cos(\omega)\cdot\cos(\varphi) - \sin(\omega)\cdot\sin(\varphi) \\ &=: \alpha_\varphi \cdot \cos(\omega) + \beta_\varphi \cdot \sin(\omega)\end{aligned}$$

mit den Konstanten $\alpha_\varphi := \cos(\varphi)$ und $\beta_\varphi := -\sin(\varphi)$. Ein verschobener Cosinus läßt sich also als gewichtete Summe der *unverschobenen* Cosinus- und Sinusfunktion repräsentieren. Somit beschreibt (2.2) tatsächlich eine allgemeingültige Methode zur Darstellung beliebiger Zeitreihen.

2.5.2.1.1.1 Frequenzgang eines Zeitsignals Wir wollen den *Frequenzgang* eines Zeitsignals berechnen und visualisieren, d.h. wir wollen wissen, mit welcher *Stärke* eine bestimmte Frequenz, repräsentiert durch die sinusförmigen Funktionen mit der entsprechenden Frequenz aus (2.2), im Signal vertreten ist, und mit welcher *Phase* sie gegenüber dem herkömmlichen Cosinus verschoben ist. Es ist bekannt [Smi99], daß die *„DFT-Kernel“*

$$\begin{aligned}\sigma_k(t) &:= \sin(t \cdot 2\pi k/T)\,, \quad 0 \leq k \leq T/2 \\ \zeta_k(t) &:= \cos(t \cdot 2\pi k/T)\,, \quad 0 \leq k \leq T/2\end{aligned} \qquad \textit{(DFT-Kernel)} \qquad (2.3)$$

orthogonale Basisfunktionen für den Vektorraum aller Zeitsequenzen (x_t) der Länge $T/2+1$ darstellen, d.h. wir können nach (2.2) für festes k das Koeffizienten-Paar (α_k^*, β_k^*) als *Rechteckkoordinaten* der cartesischen Zahlenebene auffassen. Hieraus ergeben sich die *„Amplitude“* r_k

[2] Man kann eine Zeitreihe ungerader Länge durch Hinzufügen eines geeigneten Wertes auf die geforderte Form bringen; für Unterschriften stellt dies für gewöhnlich kein Problem dar, da diese ohnehin als Teil einer längeren Aufzeichnung entstanden sind und für gewöhnlich an den Rändern *„DC-Abschnitte“*, d.h. Intervalle konstanten Verlaufs („Direct Current“), aufweisen.

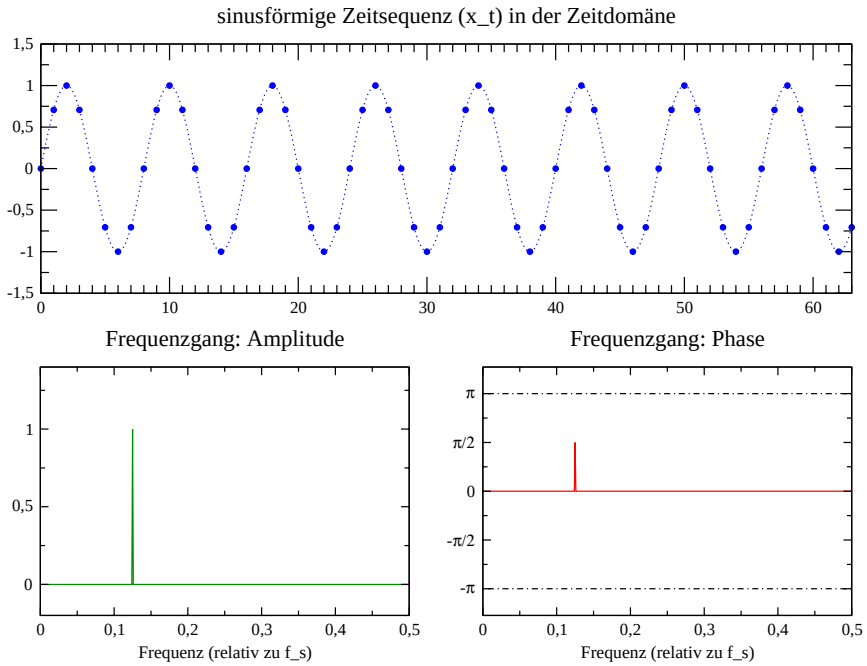

Abbildung 2.7: Frequenzgang einer sinusförmigen Zeitreihe

und die *Phase* φ_k des Frequenzgangs nach Umrechnung in *Polarkoordinaten*:

$$r_k := \sqrt{{\alpha_k^*}^2 + {\beta_k^*}^2} \qquad \text{(Amplitude)}$$
$$\varphi_k := \arctan(\beta_k^*/\alpha_k^*) \quad \text{(Phase)}$$

Abb. 2.7 zeigt den Frequenzgang einer durch $x_t := \sin(t \cdot \pi/4)$, $0 \le t < 64$ definierten Zeitreihe. Wir werden auch sagen, daß (x_t) in der *„Zeitdomäne“*, der Frequenzgang hingegen in der *„Frequenzdomäne“* dargestellt sei. Auf der Abszisse der beiden Schaubilder werden anstelle der Indexe k die *Frequenzen* $f_k := k/T$ als *prozentuale Anteile an der Samplingfrequenz* f_S angegeben. Wegen $0 \le k \le T/2$ beträgt die höchste darstellbare Frequenz somit, unabhängig von der Anzahl an Indexen, stets $0.5 f_S$. Das Schaubild unten links gibt den Verlauf der *Amplitude* r_k wieder. Die Ordinate kann nur nichtnegative Werte darstellen. Da es sich bei (x_t) offenbar um die (auf Länge T ausgeweitete) DFT-Kernelfunktion $\sigma_8(t)$ aus (2.3) handelt, existiert bei der Frequenz $f_8 = 8/64 f_S = 0.125 f_S$ die Amplitude $r = 1$, für alle anderen Frequenzen gilt $r = 0$. Die Ordinate der *Phase* φ_k (unteres rechtes Schaubild) kann allgemein Werte im Bereich $[-\pi, +\pi]$ darstellen.[3] In unserem Fall wird nur bei $f := 0.125 f_S$ ein von 0 verschiedener Wert angenommen: Die Sinus-Funktion entspricht einer um $\pi/2$ verschobenen Cosinus-Funktion.

2.5.2.1.1.2 Hamming-Window Modulation Vor der Durchführung der DFT wird man das Zeitsignal häufig mit einem *Hamming-Window* (w_l) der Länge L (L ungerade) *modulieren* wollen:

$$w_l := \begin{cases} 0.54 - 0.46 \cdot \cos(2\pi l/(L-1)) & : 0 \le l < L \\ 0 & : \text{sonst} \end{cases} \qquad \textit{(Hamming-Window)} \qquad (2.4)$$

[3] Wir verwenden tatsächlich eine Version der Funktion ‘$\arctan(y, x)$’, welche y und x als *zwei getrennte* Argumente entgegen nimmt, da nur eine solche in der Lage ist, zwischen ‘(y, x)’ und ‘$(-y, -x)$’ zu unterscheiden und diesen Kombinationen unterschiedliche Werte im kompletten Bereich $[-\pi, +\pi]$ anstatt lediglich im Bereich $(-\frac{\pi}{2}, +\frac{\pi}{2})$ zuzuordnen. Speziell liefert eine solche Realisierung die Werte $\arctan(1, 0) = \pi/2$, $\arctan(0, 1) = 0$ und $\arctan(0, 0) = 0$. Eine entsprechende Funktion gibt es z.B. mit dem Namen ‘`atan2(·,·)`’ in der Standard-Bibliothek der Programmiersprache C [KR88] sowie zu zahlreichen anderen Sprachen.

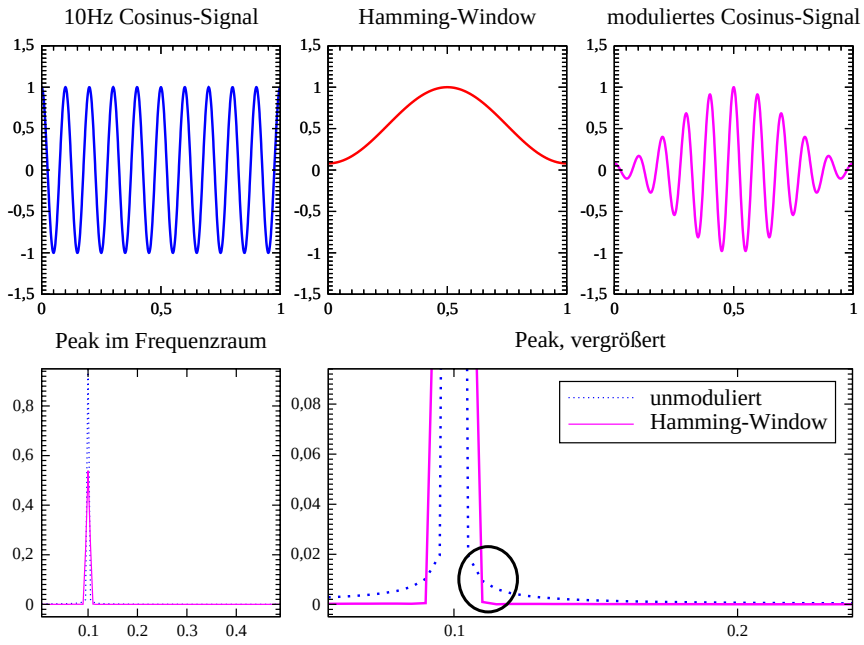

Abbildung 2.8: Modulation mittels Hamming-Window

Mit *„Modulation“* ist dabei die *punktweise Multiplikation* $(z_t) := (x_t \cdot y_t)$ zweier Sequenzen (x_t) und (y_t) gemeint. Abb. 2.8 veranschaulicht dies am Beispiel eines cosinusförmigen Signals (o.l.), welches mit dem Hamming-Window (o.m.) moduliert wurde (o.r.). Der Grund für die Verwendung eines Hamming-Windows ist, daß ein hamming-moduliertes Signal in der Frequenzdomäne oft eine schärfere Abtrennung enger Strukturen von seiner Umgebung erlaubt. In der unteren Hälfte von Abb. 2.8 sehen wir links das obige Cosinus-Signal nach Durchführung der DFT als scharfen Peak dargestellt. Es wurde sowohl unmoduliert als auch hamming-moduliert in die Frequenzdomäne transformiert und nimmt dort in beiden Fällen die gleiche Frequenz-Position ein. Bei genauerer Betrachtung (rechts) fallen allerdings zwei Dinge auf: *Erstens:* Der Peak des unmodulierten Signals *verbreitert* sich an seinem unteren Ende sehr stark, während sich aus der Hamming-Modulation ein Peak von weitgehend unveränderter Breite ergibt. *Zweitens:* Der Peak des modulierten Signals ist im oberen Bereich *breiter* als der des unmodulierten Signals. Man erhält durch Hamming-Modulation also eine größere *Akkuratheit* des Peaks bei zugleich geringerer *Präzision* bzgl. der Frequenzposition. In der Praxis ist dieser Austausch häufig vorteilhaft, da im Falle eines starken Auseinanderlaufens einer an sich engbegrenzten Struktur leichter eine Verschmelzung mit den umgebenden Frequenzdaten stattfinden kann, was dann die Erkennung von interessanten Strukturen in der Frequenzdomäne erschweren kann. Für eine theoretische Begründung des hier dargestellten Sachverhaltes s. [Smi99].

2.5.2.1.1.3 Berechnung der DFT-Koeffizienten Die Koeffizienten der DFT lassen sich grundsätzlich durch direkte Umsetzung der definierenden Gleichungen aus (2.1) algorithmisch berechnen. Der Laufzeitaufwand beträgt hierfür allerdings $\Omega(T^2)$. Unser Anwendungsfall werden Unterschriften-Zeitsequenzen sein, und diese besitzen aufgrund der hohen Samplingrate des BiSP-Pens (vgl. 2.4.1) i.d.R. Längen von einigen *tausend* Samples. Eine effizientere Methode stellt die sog. *Fast Fourier Transform (FFT)* dar, welche in [Smi99] beschrieben wird. Dieser Algorithmus besitzt die Laufzeitkomplexität $\mathrm{O}(T \cdot \log T)$, er setzt allerdings voraus, daß die Länge T einer Zeitsequenz eine *Zweierpotenz* darstellt. Für eine Unterschrift kann dies bedeuten, daß diese möglicherweise mit relativ vielen zusätzlichen konstanten Werten links und rechts des eigentlich interessierenden Signals aufgefüllt werden muß („Padding“). Für die Praxis sollte dies aber generell kein schwerwiegendes Problem darstellen.

2.5.2.1.2 Störsignale Als *Störsignale* werden in dieser Arbeit sämtliche dem eigentlichen Nutzsignal (u_t) *additiv* überlagerten zeitlichen Einflüsse bezeichnet. Sei also (u'_t) das mit dem Erfassungsgerät gemessene Zeitsignal, und seien $(\tilde{u}_t^{(i)})$, $1 \leq i \leq m$, sämtliche Störsignale gleicher Länge, dann gilt

$$(u'_t) = (u_t) + \sum_{1 \leq i \leq m} (\tilde{u}_t^{(i)}) = (u_t + \sum_{1 \leq i \leq m} \tilde{u}_t^{(i)})$$

Als Ziel ergibt sich damit die Abtrennung der Störsignale $(\tilde{u}_t^{(i)})$ vom Ausgangssignal (u'_t) per *Subtraktion*, um das Nutzsignal (u_t) freizulegen, wobei aber i.a. weder Anzahl noch Aufbau der Störungen $(\tilde{u}_t^{(i)})$ bekannt sind.

2.5.2.1.2.1 Beispiel: Resonanzen Ein Beispiel für ein Störsignal sind *Resonanzen*. Jedes physikalische Objekt besitzt bestimmte charakteristische Resonanzfrequenzen, auch „Eigenfrequenzen" genannt. Damit kann es z.B. passieren, daß während des Schreibens einer Unterschrift die Materialresonanzen des Schreibgerätes mit aufgezeichnet werden. Da Resonanzen üblicherweise durch verhältnismäßig klare sinusförmige Schwingungen charakterisiert sind, zeigen sie sich im Frequenzgang als enge Peaks, ähnlich wie in Abb. 2.7 auf S. 15 idealisiert dargestellt, an der Stelle der entsprechenden Resonanzfrequenz; gelegentlich gefolgt von weiteren Peaks geringerer Amplitude bei Vielfachen dieser Grundfrequenz („Oberschwingungen").

Solcherart Resonanzen sollten dann in den Frequenzgängen *sämtlicher* erfaßter Unterschriften auftreten. Ihre Identifikation kann daher evtl. durch das *additive Überlagern* der Frequenzgänge mehrerer Unterschriften *verschiedener* Personen gelingen: Seien $(u_t^{(i)})$, $1 \leq i \leq m$, Unterschriftenproben von m Personen und $(r_{f_k}^{(i)})$ die zu den $(u_t^{(i)})$ gehörenden Amplituden-Frequenzgänge. Dann wird es, sofern sich die Frequenzgänge $(r_{f_k}^{(i)})$ deutlich voneinander unterscheiden, für

$$(r_{f_k}^*) := (\langle r_{f_k}^{(i)} \rangle_i) \tag{2.5}$$

zu einer gewissen *Auslöschung* der Nutzsignalanteile kommen, während die Resonanzanteile auch in $(r_{f_k}^*)$ noch in ursprünglicher Stärke vorhanden sein werden. Ist die Auslöschung weitgehend vollständig, so läßt sich durch $(\hat{r}_{f_k}^{(i)}) := (r_{f_k}^{(i)} - r_{f_k}^*)$ eine Schätzung für den Frequenzgang des Nutzsignals $(\hat{u}_t^{(i)})$ erreichen.

2.5.2.1.2.2 Beispiel: Weißes Rauschen Ein weiteres Beispiel für ein Störsignal ist *weißes Rauschen*, wie es uns z.B. in Form des beim Digitalisieren von Daten entstehenden *Quantisierungsrauschens* (vgl. 2.3.2.1) begegnet. Eine echtes weißes Rauschen produzierende Quelle besitzt die statistischen Eigenschaften eines Zufallsprozesses.

Abb. 2.9 zeigt Amplitude und Phase des Frequenzspektrums einer durch einen algorithmischen Pseudozufallsgenerator erzeugten Datensequenz, welche uniformverteilt Werte im Bereich $[-1, +1]$ enthält. Charakteristisch für weißes Rauschen ist eine über den gesamten Frequenzbereich um einen konstanten Wert c streuende Amplitude. Im gezeigten Beispiel (links) wird eine einzelne stark streuende Datensequenz durch das hellgrün gepunktete Signal dargestellt, das schwächer um den konstanten Wert $c \approx 0.015$ streuende Signal (dunkelgrüne durchgezogene Linie) wurde durch gegenseitige additive Auslöschung mehrerer Zufallssequenzen gemäß (2.5) erzeugt. Die *Phase* weißen Rauschens nimmt zufällige Werte im Bereich $[-\pi, +\pi]$ an (rechtes Schaubild).

Ein das Nutzsignal überlagerndes weißes Rauschen hebt in der Frequenzdomäne das gesamte Amplitudenspektrum des Nutzsignals um einen konstanten Wert an und beschädigt dessen Phasenverlauf. In der Praxis trifft man desöfteren auf die Situation, daß das Nutzsignal vor allem im Hochfrequenzbereich recht schwach ist und dort dann im „Grundrauschen"

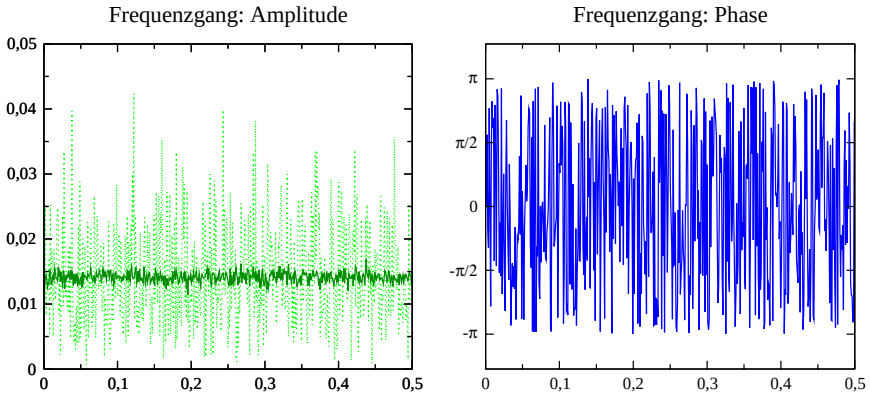

Abbildung 2.9: Frequenzspektrum von weißem Rauschen

verloren geht. Ziel wird es in einem solchen Fall sein, den Rauschpegel im Tieffrequenzbereich abzusenken, um ein besseres Signal/Rausch-Verhältnis zu erlangen, und die hohen Frequenzen *insgesamt* in ihrer Amplitude zu reduzieren, um weitere durch hochfrequente Einflüsse entstehende Störungen des Datenverarbeitungsprozesses zu vermeiden.

2.5.2.1.3 Verzerrungen Als *Verzerrung* bezeichnen wir in dieser Arbeit sämtliche *nicht* durch additive Überlagerung gemäß 2.5.2.1.2 entstandenen Veränderungen am Nutzsignal, also z.B. *Dehnungen* von Teilen des Zeitsignals in Amplitude oder entlang der Zeitrichtung.

In manchen Fällen besitzt das Nutzsignal einen verzerrten *Frequenzgang*. Es wäre z.B. denkbar, daß die Drucksensoren des BiSP-Pens in Abhängigkeit der Druckfrequenz unterschiedliche Werte bei an sich gleichen physikalischen Drücken messen. Eine Verzerrung des Frequenzgangs kann aber auch bewußt herbeigeführt werden: Vor dem Pressen einer Schallplatte werden aus technischen Gründen die tiefen Töne in ihrer Amplitude abgesenkt und die hohen Töne lautstärkemäßig angehoben. Beim Abspielen des Signals wird durch eine geeignete elektronische Schaltung namens „RIAA-Entzerrer“ diese Frequenzgangverzerrung wieder rückgängig gemacht [Gre02].

Mathematisch wird eine Frequenzgangverzerrung erreicht durch *punktweise Multiplikation* (*Modulation*, vgl. 2.5.2.1.1.2) des originalen Frequenzganges mit einer von konstant Eins verschiedenen Funktion. Die *Entzerrung* stellt die entsprechende Rücktransformation durch punktweise *Division* dar. Den auf diese Weise charakterisierten Typ von Verzerrungen nennt man *„lineare“* Verzerrung. Die Beseitigung einer linearen Verzerrung gelingt damit *prinzipiell* dadurch, daß man das verzerrte Zeitsignal mittels einer DFT in die Frequenzdomäne transformiert, es dort in der genannten Weise entzerrt, und am Ende mittels IDFT wieder in die Zeitdomäne abgebildet. Dies läßt sich in der Praxis aufgrund der damit verbundenen Berechnungskomplexität jedoch nur für *kurze* Zeitsequenzen durchführen. Eine Methode zur Entzerrung langer, auch unbeschränkt langer Zeitsignale wird in 2.5.2.1.8 präsentiert werden.

Speziell bei Unterschriften ergibt sich das Problem, daß das Verhältnis zwischen zwei Unterschriftenproben der gleichen Person gerade geprägt ist durch das Vorhandensein lokaler Verzerrungen. Daher empfiehlt es sich, was die Identifikation und Beseitigung von Verzerrungen betrifft, vorsichtig vorzugehen.

2.5.2.1.4 Lineare Filter In den folgenden Abschnitten betrachten wir einen allgemeinen Ansatz, der bei der Beseitigung von Störsignalen und Verzerrungen hilfreich sein kann.

Als *Kernel eines linearen Filters* oder einfach *Filter* bezeichnen wir eine reelle Zahlenfolge $(h_l)_{0\leq l<L} \in \mathbb{R}^L$, $\sum_l h_l = 1$, zusammen mit einer Operation ‘$\star$’, *Konvolution* genannt,

über die die Filterung eines Zeitsignals $(u_t)_{0\leq t<T}$ definiert wird:[4]

$$\begin{aligned}&(u_t)_{0\leq t<T} \star (h_l)_{0\leq l<L} := (\sum_{0\leq l<L} u_{t-l}\cdot h_l)_{0\leq t<T+L-1} \quad \textit{(Konvolution)}\\ &\text{mit } u_t := 0 \text{ für } t<0 \text{ und } t\geq T\end{aligned} \tag{2.6}$$

Das Ergebnis dieser Operation ist also eine Zeitreihe der Länge $T+L-1$. Die Laufzeitkomplexität für die Konvolution beträgt bei direkter Umsetzung von (2.6) offenbar $O(T\cdot L)$. Die Forderung, daß sich die *Filterkoeffizienten* h_l zu 1 aufsummieren müssen, hat verschiedene technische Gründe, und wird für unsere Zwecke keine Einschränkung bedeuten.

Wir werden im Verlauf der Arbeit diverse algebraische Eigenschaften der Konvolution gebrauchen, die sich allesamt durch einfaches Nachrechnen beweisen lassen:

$$\begin{aligned}&[(u_t^{(1)})+(u_t^{(2)})]\star(h_l) = [(u_t^{(1)})\star(h_l)]+[(u_t^{(2)})\star(h_l)] && \textit{(Additivität)}\\ &[c\cdot(u_t)]\star(h_l) = c\cdot[(u_t)\star(h_l)] && \textit{(Skalierbarkeit)}\\ &[(u_t)\star(h_l^{(1)})]\star(h_l^{(2)}) = (u_t)\star[(h_l^{(1)})\star(h_l^{(2)})] && \textit{(Assoziativität)}\\ &(h_l^{(1)})\star(h_l^{(2)}) = (h_l^{(2)})\star(h_l^{(1)}) && \textit{(Kommutativität)}\end{aligned} \tag{2.7}$$

Die *Kommutativität* erlaubt es uns prinzipiell, die zu verwendenden Filter in jeder beliebigen Reihenfolge anzuwenden. Soll auf viele Zeitsignale stets die gleiche Folge von Filtern angewandt werden, so läßt sich durch Konvolution aus allen diesen Filtern ein einziger Gesamtfilter generieren, der dann auf die Zeitsignale anzuwenden ist; durch die *Assoziativität* ist eine solche Vorgehensweise theoretisch abgesichert. Lineare Filter sind ferner unempfindlich gegenüber der Größe der Amplitude (Skalierbarkeit). Außerdem ermöglichen sie durch die *Additivität* folgende Strategie: Ein Signal läßt sich gemäß einer bestimmten Analysestrategie in einzelne *Summandensignale* zerlegen, die dann getrennt gefiltert und auf interessierende Eigenschaften hin untersucht werden können. Die Superposition aller gefilterten Summandensignale führt zum gleichen Ergebnis wie eine Filterung des Gesamtsignals.

2.5.2.1.5 Filter zur Zeitsignal-Glättung Wir beschäftigen uns in diesem Abschnitt mit einer filterbasierten Methode zur Beseitigung von *weißem Rauschen* (vgl. 2.5.2.1.2.2). Abb. 2.10 zeigt ein mit einer Zufallsfolge überlagertes Rechtecksignal (l.o.). Zur Rekonstruktion des ursprünglichen Signals (dicke Linie) werden wir den sog. *Moving-Average-Filter (MA)*

$$(h_l^{\mathrm{MA}}) := (1/L)_{0\leq l<L} \qquad \textit{(Moving-Average-Filter)} \tag{2.8}$$

verwenden, dessen Koeffizienten allesamt den gleichen Wert besitzen (Schaubild l.u.). Faktisch wird durch einen MA-Filter der *Mittelwert* der in der lokalen Umgebung einer Stelle t liegenden Samples u_{t-l} berechnet, was zu einer Auslöschung zufälliger Streuwerte führen kann.

[4] In der Literatur zur Signaltheorie (z.B. [Smi99]) geht man häufig von sog. *linearen Systemen* aus, welche Zeitsignale in linearer Weise transformieren, d.h. es gilt für ein lineares System $S[\cdot]$, Zeitsequenzen $(u_t^{(i)})$ und Konstanten c_i:

$$S[\sum_{1\leq i\leq m} c_i\cdot(u_t^{(i)})] = \sum_{1\leq i\leq m} c_i\cdot S[(u_t^{(i)})]$$

Zusätzlich gilt üblicherweise *Verschiebungsinvarianz*: Aus $(y_t) := S[(u_t)]$ folgt $(y_{t-s})_t = S[(u_{t-s})_t]$, für $s\in\mathbb{Z}$. Wird dem linearen System speziell ein *Impulssignal* $(\delta_{l,0})_l$ (Kroneckersymbol) eingespeist, so liefert es seine sog. *Impulsantwort* $(h_l) := S[(\delta_{l,0})]$ als Ausgabe. Da sich jedes Zeitsignal (u_t) formal darstellen läßt als

$$(u_t) = \sum\nolimits_t u_t\cdot(\delta_{l-t,0})_l$$

so ist durch die Impulsantwort (h_l) das Verhalten des linearen Systems bereits vollständig spezifiziert: Das Ergebnis der Anwendung von $S[\cdot]$ auf (u_t) ist die *Konvolution* von (u_t) mit (h_l), wie sie in (2.6) definiert wird. Es erscheint daher gerechtfertigt, die Zahlenfolge (h_l) *selbst* als Filter zu bezeichnen anstelle eines linearen Systems, welches (h_l) als Impulsantwort besitzt.

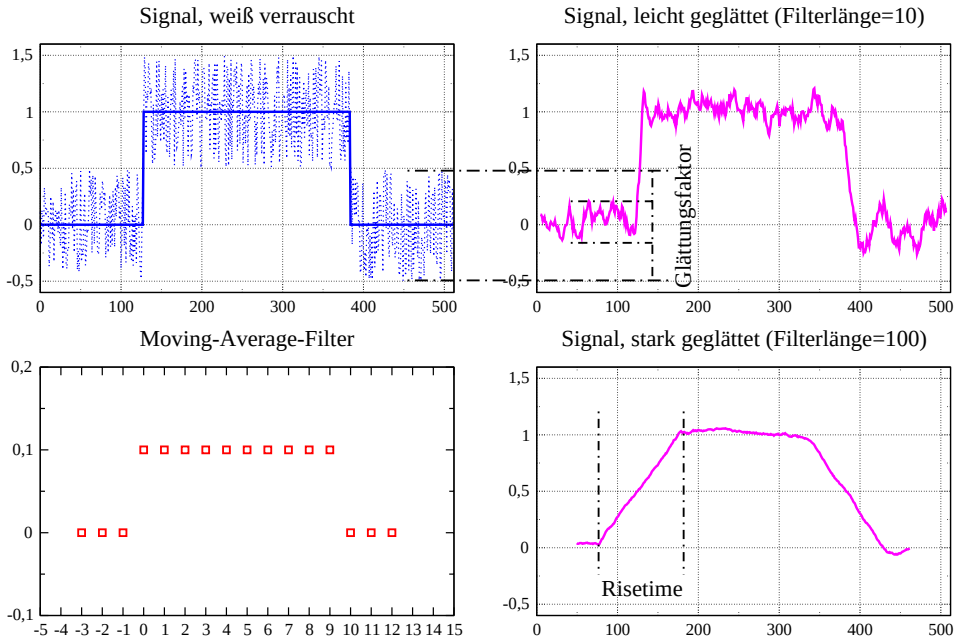

Abbildung 2.10: Zeitsignalglättung mittels Moving-Average-Filter

Dieser Effekt läßt sich am Schaubild r.o. nachvollziehen: Auf das verrauschte Ausgangssignal wurde ein MA-Filter der Länge $L = 10$ angewandt, mit dem Ergebnis, daß sich die Rauschamplitude etwa auf ein Drittel der Ursprungsgröße reduziert hat. Dies ist eine gute Übereinstimmung mit der Theorie [Smi99], wonach die durch einen Moving-Average-Filter der Länge L erbrachte Rauschreduktion bei *weißem Rauschen* proportional zu $\sqrt{L}$ ist.

Im Schaubild r.u. wurde ein MA-Filter der Länge $L = 100$ auf das Ausgangssignal angewandt. Neben der erwarteten starken Glättung fällt hier ein als negativ zu bewertender Effekt auf: Das zu rekonstruierende Signal besitzt als Rechtecksignal steil ansteigende und abfallende Flanken, das geglättete Signal weist jedoch eine deutlich *Trapezform* auf. Quantitativ ausgedrückt beträgt die sog. *„Risetime“*, welche die Anzahl an Zeitindexen zwischen dem Beginn und dem Ende des Anstiegs (oder Abstiegs) einer Flanke bezeichnet, rd. 100 Punkte und besitzt damit die gleiche Länge wie der eingesetzte Filter. Eine feine Struktur im Nutzsignal, deren Größe deutlich unter der Länge des Filterkernels liegt, würde bei der Filterung damit stark verzerrt werden. Man wird daher stets einen Kompromiß zwischen gewünschter Rauschreduktion und zu bewahrender minimaler Strukturgröße eingehen müssen.

Die naive Idee, einen MA-Filter kleiner Länge mehrfach hintereinander auf ein gestörtes Signal anzuwenden, bringt nur eine geringe Verbesserung mit sich, denn nach der einmaligen Anwendung eines MA-Filters besitzt ein ursprünglich weißes Rauschen nicht mehr die statistischen Eigenschaften einer Zufallsfolge. Tatsächlich entspricht die zweimalige Anwendung eines MA-Filters der Länge L aufgrund der *Assoziativität der Konvolution* gemäß (2.7) der einmaligen Filterung des Signals mit einem *Dreieckfilter* der Länge $2L - 1$:

$$(h_l^{\triangle}) := (h_l^{\mathrm{MA}})_{0\le l<L} \star (h_l^{\mathrm{MA}})_{0\le l<L} = \begin{cases} \frac{l+1}{L}\cdot\frac{1}{L} & : 0 \le l \le L-2 \\ \frac{1}{L} & : l = L-1 \\ \frac{2L-1-l}{L}\cdot\frac{1}{L} & : L \le l \le 2L-2 \end{cases} \qquad \textit{(Dreieckfilter)}$$

Für den MA-Filter ist bekannt [Smi99], daß er bei fest vorgegebener Flankensteilheit (ausgedrückt in der Risetime der ansteigenden Flanke eines Rechtecksignals) die größtmögliche Rauschreduzierung unter allen linearen Filtern produziert. Der Dreieckfilter ist dem

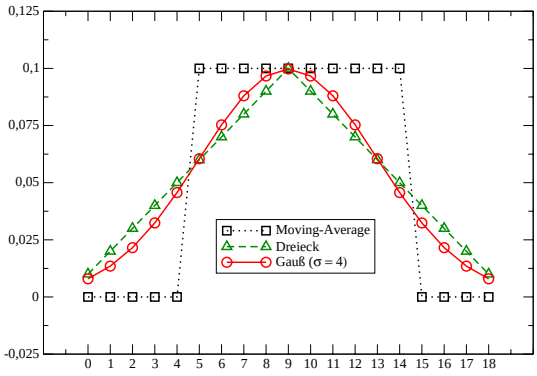

Abbildung 2.11: MA-Filter, Dreieck-Filter und Gauß-Filter im Vergleich

MA-Filter in diesem Punkt also unterlegen, und dies gilt ebenso für den *Gaußfilter*

$$(h_l^{G_\sigma}) := (\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(l-L/2)^2}{\sigma^2}})_{0\leq l<L} \qquad \textit{(Gaußfilter)}$$

welcher in gewisser Weise den Grenzfall der mehrfachen Anwendung des MA-Filters darstellt: Die Konvolution zweier Gaußfilter ergibt ebenfalls einen Gaußfilter. Abb. 2.11 stellt die Formen von MA-Filter, Dreieck-Filter und Gauß-Filter einander gegenüber.

Dennoch verfügt auch der Gaußfilter über gute Glättungseigenschaften, und darüber hinaus besitzt er den Vorteil, daß seine Anwendung zu geringeren Verzerrungen in der *Frequenzdomäne* führt als der MA-Filter [Smi99]. Tatsächlich läßt sich ein Gaußfilter prinzipiell als *Tiefpaßfilter* einsetzen, wir werden im folgenden Abschnitt jedoch einen für diesen Zweck besser geeigneten Spezialfilter betrachten. Der MA-Filter hat wiederum den Vorteil der besonders effizienten Umsetzbarkeit: Anstatt über eine Konvolution gemäß (2.6) läßt sich die Filterung eines Signals $(u_t)_{0\leq t<T}$ über die Rekursionsgleichung

$$y_t := \begin{cases} \frac{1}{L}\cdot u_0 & : t = 0 \\ y_{t-1} + \frac{1}{L}\cdot u_t & : 1 \leq t < L \\ y_{t-1} + \frac{1}{L}\cdot (u_t - u_{t-L}) & : L \leq t < T \\ y_{t-1} - \frac{1}{L}\cdot u_{t-L} & : T \leq t < L+T-1 \end{cases} \qquad \begin{array}{l}\textit{(rekursive Berechnung} \\ \textit{der MA-Filterung)}\end{array}$$

in Zeit $\mathrm{O}(T+L)$ durchführen.

2.5.2.1.6 Tiefpaß-Filterung Die in 2.5.2.1.2.2 erwähnte Absenkung des Pegels hochfrequenten Rauschens läßt sich mittels eines *Tiefpaßfilters* erledigen. Ein *idealer* Tiefpaßfilter würde die in Abb. 2.12 (l.o.) gezeigte Darstellung in der Frequenzdomäne, die sog. *„Frequenzantwort“* $\tilde{H}_{f_C}^{\mathrm{low}}$ des Filters, besitzen:

$$\tilde{H}_{f_C}^{\mathrm{low}}(f) = \begin{cases} 1 & : 0 \leq f < f_C \\ 0 & : f_C \leq f \leq 0.5 \end{cases} \qquad \textit{(idealer Tiefpaß, Frequenzantwort)}$$

Dabei bezeichnet ‘f_C’ die sog. *Cutoff-Frequenz* des Tiefpaßfilters, also diejenige Position in der Frequenzdomäne, ab der alle Frequenzen unterdrückt werden. Die Inverse Fouriertransformierte von $\tilde{H}_{f_C}^{\mathrm{low}}$ heißt *Sinc-Sequenz* (Abb. 2.12, r.o.):

$$\mathrm{sinc}_l := \begin{cases} \frac{\sin(2\pi f_C \cdot l)}{\pi \cdot l} & : l \in \mathbb{Z}\backslash\{0\} \\ 2f_C & : l = 0 \end{cases} \qquad \textit{(Sinc-Sequenz)}$$

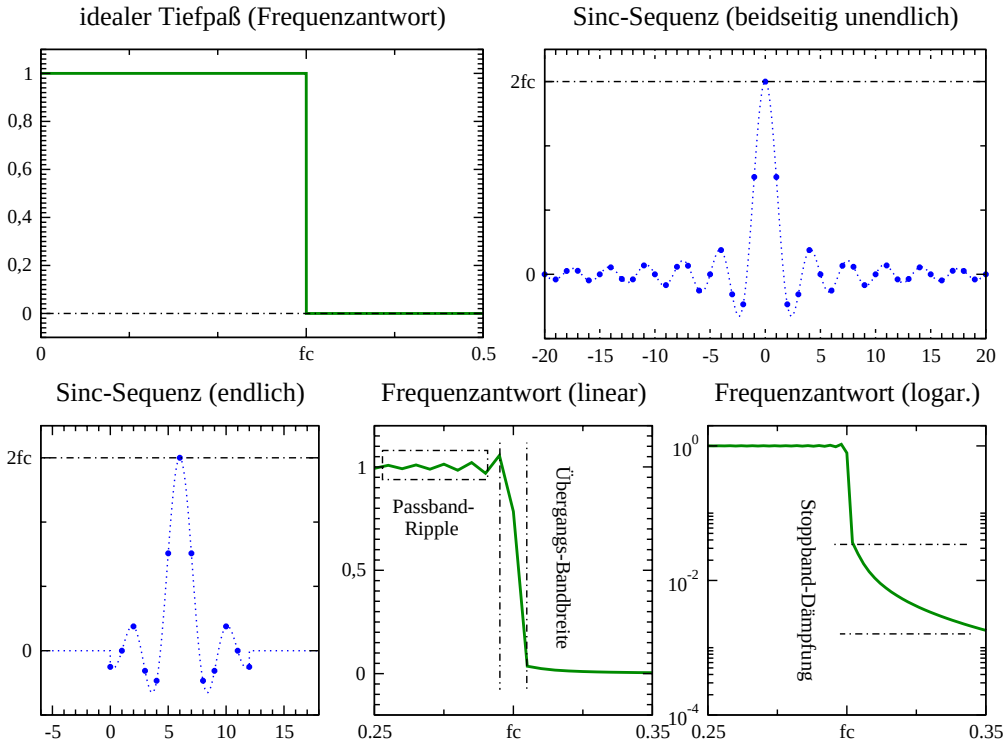

Abbildung 2.12: Tiefpaßfilter: ideal (oben) und real (unten)

Diese Sequenz besitzt eine in positiver und negativer Richtung unendliche Länge. Wir betrachten daher nur den (in horizontaler Richtung verschobenen) Ausschnitt

$$(\hat{h}_l^{\text{low}})_{0\le l\le L} := (\text{sinc}_l)_{-L/2\le l\le L/2} \qquad \textit{(Sinc-Filter)}$$

der Länge $L+1$ als Filterkernel (Abb. 2.12, unten links), dessen Frequenzantwort nun allerdings nicht länger ideal ist. Es treten dabei folgende drei in der Abbildung unten (mitte und rechts) skizzierte Probleme auf:

- *„Paßband-Ripple“ R:* Im Frequenzbereich unterhalb der Cutoff-Frequenz f_C erkennt man Schwingungen, die zu Verzerrungen im zu filternden Signal führen werden.
- *Unvollständige „Stoppband-Dämpfung“ D:* Im Frequenzbereich oberhalb von f_C existieren Wellen, d.h. es werden nach wie vor Bestandteile aus dem eigentlich zu entfernenden Frequenzband im gefilterten Signal übrig bleiben.
- *„Übergangs-Bandbreite“ B:* Das Absinken der Amplitude findet nicht abrupt statt, sondern vollzieht sich über einen bestimmten Frequenzbereich.

Paßband-Ripple und Stoppband-Dämpfung lassen sich durch Modulation des Filterkernels (in der Zeitdomäne) mit dem sog. *Blackman-Window* $(W_l)_{0\le l\le L}$

$$W_l := 0.42 - 0.5\cdot\cos(2\pi l/L) + 0.08\cdot\cos(4\pi l/L) \qquad \textit{(Blackman-Window)}$$

einem mit dem Hamming-Window (2.4), S. 15, verwandten Modulator,[5] reduzieren [Smi99]: Der Paßband-Ripple beträgt danach $R \approx 0.02\%$, die Stoppband-Dämpfung D wird von -21dB ohne Modulation auf -74dB abgesenkt. Die Länge der Übergangsbandbreite B ist dagegen

[5] Das Hamming-Window läßt sich im Prinzip ebenfalls einsetzen, es besitzt aber eine um etwa den Faktor 10 schlechtere Leistung für die hier betrachtete Aufgabe.

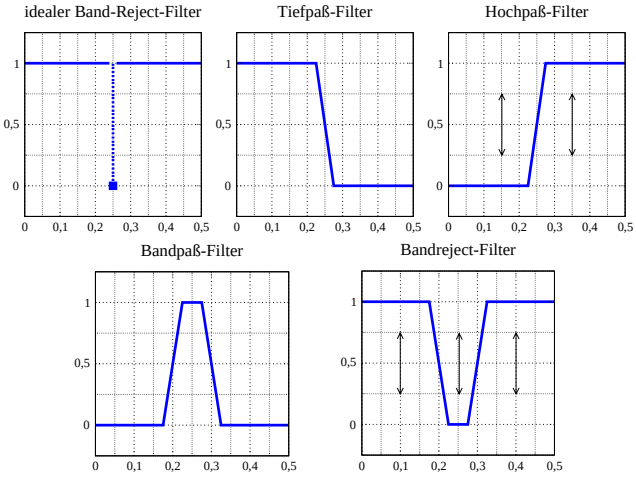

Abbildung 2.13: Konstruktion eines Bandreject-Filters

primär abhängig von der Länge L des Filterkernels (eine gewisse Vergrößerung der Übergangsbandbreite entsteht auch durch den eingesetzten Modulator). B kann als Abstand $|f_r - f_l|$ angegeben werden, wobei f_l die Frequenz bezeichnet, bei der die Amplitude den Wert 0.99 annimmt, und f_r die Stelle, wo der Frequenzgang den Wert 0.01 erreicht. In diesem Fall gilt die ungefähre Beziehung

$$L \approx 4/B \qquad \textit{(Verhältnis von Übergangsbandbreite und Filterlänge)} \tag{2.9}$$

Durch Wahl der Cutoff-Frequenz f_C und der Länge B der Übergangsbandbreite läßt sich damit die Form des blackman-modulierten Sinc-Filters, der sog. *„Windowed-Sinc-Filter“*, bestimmen:

$$h_l^{\text{low}} := \frac{\hat{h}_l^{\text{low}} \cdot W_l}{\sum_{l=0}^{L} \hat{h}_l^{\text{low}} \cdot W_l}, \quad 0 \le l \le L \qquad \textit{(Windowed-Sinc-Filter)} \tag{2.10}$$

Die Summe im Nenner sorgt dafür, daß $\sum_l h_l^{\text{low}} = 1$ gilt (vgl. 2.5.2.1.4).

2.5.2.1.7 Bandreject-Filterung Eine Resonanz-Störung drückt sich im Idealfall durch eine einzelne das Nutzsignal überlagernde sinusförmige Schwingung aus, welche in der Frequenzdomäne als enger Peak bei der Resonanzfrequenz f_R repräsentiert ist (vgl. 2.5.2.1.2.1). Möchte man diesen Peak beseitigen, ohne den Restfrequenzgang unnötig in Mitleidenschaft zu ziehen, so benötigt man im Prinzip einen Filter mit der Frequenzantwort $\tilde{H}^{\text{rej}}(f) := 1 - \delta_{f,f_R}$ (Abb. 2.13 o.l.). Einen Filter endlicher Länge mit dieser Charakteristik gibt es nicht, auf der Grundlage des in 2.5.2.1.6 vorgestellten Tiefpaßfilters, schematisch dargestellt im Schaubild o.m., läßt sich jedoch eine beliebig genaue Näherung für einen solchen Filter konstruieren.

Der erste Schritt besteht in der Generierung eines *Hochpaßfilters* (o.r.). Einen solchen erhält man konzeptionell dadurch, daß man die Frequenzantwort des Tiefpaßfilters entlang der Horizontale $r(f) \equiv 0.5$ spiegelt. Für einen in der Zeitdomäne zur Ordinate *achsensymmetrischen* Tiefpaßfilter $(h_l^{\text{low}})_{-L/2 \le l \le +L/2}$, wie ihn der (evtl. zuvor zu verschiebende) Windowed-Sinc-Filter (2.10) darstellt, erreicht man eine solche *„Spektral-Inversion“* in der Zeitdomäne wie folgt:

$$h_l^{\text{high}} := \begin{cases} 1 - h_l^{\text{low}} & : l = 0 \\ -h_l^{\text{low}} & : \text{sonst} \end{cases} \qquad \textit{(Spektral-Inversion)} \tag{2.11}$$

Eine Begründung für diese Vorgehensweise liefert [Smi99].

Wir können nun als nächsten Zwischenschritt einen *Bandpaßfilter* (h_l^{band}) (Schaubild u.l.) zur *Durchlassung* eines bestimmten Frequenzbereiches konstruieren. Die Entfernung *rechts* vom erlaubten Frequenzband liegender Frequenzen geschieht mit einem Tiefpaßfilter $(h_l^{\text{low},\rho})$ geeigneter Cutoff-Frequenz f_C^ρ und Übergangsbandbreite B^ρ. Die Entfernung der *linken* Frequenzen geschieht mit einem Hochpaßfilter $(h_l^{\text{high},\lambda})$, der aus einem Tiefpaßfilter mit den gewünschten Parametern f_C^λ und B^λ konstruiert wird. Um *beide* Seiten zu entfernen, wenden wir $(h_l^{\text{high},\lambda})$ und $(h_l^{\text{low},\rho})$ hintereinander auf das zu filternde Signal an. Aufgrund der *Assoziativität* der Konvolution aus (2.7), S. 19, ergibt sich damit der Bandpaßfilter als

$$(h_l^{\text{band}}) := (h_l^{\text{high},\lambda}) \star (h_l^{\text{low},\rho})$$

Aus (h_l^{band}) erhalten wir den angestrebten *Bandreject-Filter* (h_l^{rej}), wie er in Abb. 2.13, u.r., skizziert ist, erneut durch Spektral-Inversion durch eine zu (2.11) analoge Transformation.

2.5.2.1.8 Filter zur Dekonvolution In 2.5.2.1.3 wurden *linear verzerrte* Zeitsequenzen (u'_t) diskutiert, die man sich vorstellen kann als Zeitreihen (u_t), deren Frequenzgang punktweise mit einer Verzerrungsfolge $(z_k) \not\equiv (1)$ multipliziert (moduliert) wird. Sofern also (z_k) bekannt ist, läßt sich eine Entzerrung durch Modulation mit der Folge (z_k^{-1}) erreichen. Es läßt sich zeigen [Smi99], daß einer Modulation in der Frequenzdomäne eine *Konvolution in der Zeitdomäne* entspricht. Damit können wir (z_k^{-1}) als *Frequenzantwort* eines Filterkernels (h_l^z) auffassen, welchen wir aus (z_k^{-1}) durch Anwendung einer IDFT gemäß 2.5.2.1.1 erhalten. (h_l^z) läßt sich nun dazu verwenden, aus (u'_t) das eigentliche Nutzsignal (u_t) mittels Konvolution zu rekonstruieren. Man bezeichnet diese Art der Entzerrung als *„Dekonvolution"*. Auf diese Weise lassen sich beliebig lange Zeitsequenzen entzerren, beispielsweise Musiksignale auf Schallplatten durch den in 2.5.2.1.3 erwähnten RIAA-Entzerrer.

2.5.2.1.9 Grenzen linearer Filtertechnik Lineare Filter sind nicht anwendbar bei „nichtlinearen" Verzerrungen, die beispielsweise durch *zeitlich variable* Beeinflussungen der Zeitsequenz oder durch „Gedächtniseffekte", wie man sie als Hysterese von Magnetisierungsvorgängen kennt, entstehen. In manchen dieser Fälle kann man aber zumindest versuchen, die Zeitsequenz in kürzere Segmente zu zerlegen, für welche der Grad der nichtlinearen Änderung vernachlässigbar ist, sodaß man evtl. zumindest eine *stückweise* lineare Filterung durchzuführen in der Lage ist. Andernfalls können nur noch spezialisierte Entzerrungsmethoden helfen, die außerhalb unserer Diskussion stehen.

2.5.2.2 Daten-Zentrierung und Amplituden-Normierung

Die A/D-Wandler des BiSP-Pens (2.4.1) können je nach Konfiguration einem bestimmten physikalischen Druck unterschiedliche Ausgangswerte zuordnen, und auch der Basislevel ist variabel. Gelegentlich verschieben sich zudem, bedingt z.B. durch Temperaturunterschiede, die erzeugten Ausgangsspannungen der Sensoren, sodaß selbst bei sorgfältiger Kalibrierung vor jeder Meßreihe gewisse Unterschiede der Ergebnisse bei an sich gleichen Eingaben niemals völlig auszuschließen sind. Zu diesen äußeren Einflüssen kann hinzukommen, daß ein Schreiber zu unterschiedlichen Zeiten mit unterschiedlichem Druck schreibt, was sich sowohl auf den Grunddruck als auch auf die Stärke der Auslenkung in der Amplitude auswirken kann.

Wir betrachten in diesem Abschnitt der Einfachheit halber nur *eindimensionale* Zeitreihen. Die genannten Szenarien lassen sich dann idealisiert dadurch darstellen, daß zwischen dem eigentlichen Nutzsignal (u_t) und dem tatsächlich aufgezeichneten Signal (u'_t) die folgende Beziehung besteht, mit einer *Skalierung* c und einem *Offset* a:

$$u'_t = c \cdot u_t + a \tag{2.12}$$

Eine solche lineare Transformation ändert die charakteristische Form einer Zeitsequenz nur unwesentlich, und scheint daher wenig bedeutsame Information zu einem die Schreibdynamik auswertenden Verifikationsprozeß beizusteuern. Andererseits kann sie einen solchen erschweren: Bei einem naiven punktweisen Vergleich zweier durch lineare Transformationen verzerrte Sequenzen können beliebig große Fehlerwerte entstehen, denn es gibt mit den Benennungen aus eqrefeq:sig-affin offenbar zu jeder Schranke $S > 0$ Koeffizienten c und a mit $|u'_t - u_t| = |(c-1)u_t + a| > S$, beispielsweise $c := 1$ und $a := S + 1$. Damit kann zumindest bei einer derart einfachen Vergleichsstrategie nur schwer entschieden werden, ob die verglichenen Unterschriften von der gleichen Person stammen oder nicht. In aller Regel wird man die Parameter c und a einer solchen linearen Verzerrung nicht genau kennen, sodaß es in der Praxis nicht möglich ist, das „wesentliche“ Signal über die Rechnung $u_t = (u'_t - a)/c$ zu gewinnen. Es gilt jedoch folgender Satz:

Satz 2.1. *Sei (u_t) eine Zeitsequenz und seien $(u_t^{(i)}) := (c^{(i)} \cdot u_t + a^{(i)})$, $i \in \{1,2\}$, zwei lineare Transformationen von (u_t) gemäß (2.12). Für die beiden Sequenzen*

$$(\hat{u}_t^{(i)}) := (\frac{1}{\sigma^{(i)}} \cdot [u_t^{(i)} - \mu^{(i)}]), \quad i \in \{1,2\}$$

mit

$$\mu^{(i)} := \langle u_t^{(i)} \rangle$$
$$\sigma^{(i)} := \sqrt{\langle (u_t^{(i)} - \mu^{(i)})^2 \rangle}$$

gelten folgende Eigenschaften für alle t und $i \in \{1,2\}$:

1. $\hat{u}_t^{(1)} = \hat{u}_t^{(2)}$,
2. $\hat{\mu}^{(i)} := \langle \hat{u}_t^{(i)} \rangle = 0$,
3. $\hat{\sigma}^{(i)} := \sqrt{\langle (\hat{u}_t^{(i)} - \hat{\mu}^{(i)})^2 \rangle} = 1$,
4. *falls $\mu := \langle u_t \rangle = 0$ und $\sigma := \sqrt{\langle (u_t - \mu)^2 \rangle} = 1$, dann ist $\hat{u}_t^{(i)} = u_t$.*

Beweis. Zu 1: Es ist nach Definition

$$u_t^{(i)} - \mu^{(i)} = c^{(i)} \cdot u_t + a^{(i)} - \langle c^{(i)} \cdot u_t + a^{(i)} \rangle = c^{(i)} \cdot (u_t - \langle u_t \rangle) = c^{(i)} \cdot (u_t - \mu)$$

und damit

$$\hat{u}_t^{(i)} = \frac{1}{\sigma^{(i)}} \cdot [u_t^{(i)} - \mu^{(i)}] = \frac{u_t^{(i)} - \mu^{(i)}}{\sqrt{\langle (u_t^{(i)} - \mu^{(i)})^2 \rangle}} = \frac{c^{(i)} \cdot (u_t - \mu)}{\sqrt{\langle {c^{(i)}}^2 \cdot (u_t - \mu)^2 \rangle}} = \frac{1}{\sigma} \cdot (u_t - \mu)$$

Für $\hat{u}_t^{(1)}$ und $\hat{u}_t^{(2)}$ ergibt sich also der gleiche Wert.

Zu 2:

$$\hat{\mu}^{(i)} = \langle \hat{u}_t^{(i)} \rangle = \langle \frac{1}{\sigma} \cdot (u_t - \mu) \rangle = \frac{1}{\sigma} \cdot (\langle u_t \rangle - \langle \mu \rangle) = \frac{1}{\sigma} \cdot (\mu - \mu) = 0$$

Zu 3:

$$(\hat{\sigma}^{(i)})^2 = \langle (\hat{u}_t^{(i)} - \hat{\mu}^{(i)})^2 \rangle = \langle (\frac{1}{\sigma} \cdot (u_t - \mu) - 0)^2 \rangle = \frac{1}{\sigma} \cdot \langle (u_t - \mu)^2 \rangle = \frac{1}{\sigma} \cdot \sigma = 1$$

Aussage 4 folgt unmittelbar aus dem Beweis zu 1. □

Bei den Zeitreihen $(\hat{u}_t^{(i)})$ handelt es sich um *zentrierte* (der Mittelwert ist 0, Satz 2.1, Aussage 2) und *amplitudennormierte* (die Standardabweichung ist 1, Aussage 3) Versionen der Sequenzen $(u_t^{(i)})$. Aussage 1 ist für unsere Zwecke maßgeblich: Vorangegangene lineare Transformationen werden durch eine derartige Normierung gewissermaßen nivelliert, denn die beiden normierten Sequenzen sind identisch, und aus dem Beweis dieser Teilaussage geht hervor, daß die Abhängigkeit von den Parametern $c^{(i)}$ und $a^{(i)}$ verlorengeht. Allerdings macht der Beweis auch klar, daß i.a. nicht die ursprüngliche Zeitsequenz u_t rekonstruiert wird. Zumindest aber ist dies immer dann der Fall, wenn die zugrundeliegende Sequenz (u_t) bereits zentriert und amplitudennormiert war (Aussage 4), was man in manchen Fällen vielleicht durch geschickte Kalibrierung des Erfassungssystems zumindest näherungsweise erreichen kann. Entscheidend ist aber, daß durch zwei einfach zu bewerkstelligende Transformationen (der algorithmische Aufwand für die Zentrierung und Amplitudennormierung beträgt offenbar $O(T)$) eine gewisse Verbesserung der Vergleichbarkeit für Unterschriften hergestellt wird, wobei die Form der Sequenzen im wesentlichen erhalten bleibt; ob es sich im Ergebnis um die „Originaldaten“ handelt, ist für Fragen der Verifikation von untergeordneter Bedeutung.

2.5.2.3 Sequenzlängenangleich

Zwei Unterschriftensequenzen der gleichen Person werden i.a. eine unterschiedliche Länge, d.h. eine unterschiedliche physikalische Schreibdauer besitzen. Wir betrachten das stark vereinfachte Szenario, wonach für zwei Unterschriften $(u_t^{(1)})_{0\le t\le T_1}$ und $(u_t^{(2)})_{0\le t\le T_2}$ die Sequenz $(u_t^{(2)})$ eine um einen konstanten Faktor γ zeitlich „gedehnte“ Version von $(u_t^{(1)})$ ist, d.h. es gilt $T_2 = [\gamma \cdot T_1]$ und $u^{(2)}_{[\gamma t]} \approx u_t^{(1)}$ für $0 \le t \le T_1$ (wir ignorieren hier die dadurch nicht definierten Zwischenstellen von $(u_t^{(2)})$). Unter diesen Umständen erscheint die Ansicht plausibel, daß $(u_t^{(1)})$ und $(u_t^{(2)})$ sich *nicht wesentlich* voneinander unterscheiden. Die unterschiedlichen Längen T_1 bzw. T_2 könnten aber bei einer Verifikation dazu führen, daß die beiden Unterschriften als unähnlich eingestuft werden. Es werden hier zwei einfache Verfahren vorgestellt, mit denen ein *Sequenzlängen-Angleich* durchgeführt werden kann.

2.5.2.3.1 Resampling Sei $u := (u_t)_{0\le t\le T}$ eine Unterschriftensequenz der Länge $T+1$, und sei T^*+1 die vorgesehene Anzahl an Samples nach dem Längenangleich. Der Wert T^* könnte dabei systemglobal fest vorgegeben oder aber personenspezifisch definiert sein. Wir gehen im Folgenden davon aus, daß die Samples u_t wie im Falle des BiSP-Pens (2.4.1) in *äquidistanten* Zeitabständen aufgezeichnet wurden, und führen ein (ebenfalls äquidistantes) *Resampling* von u durch:

1. **(Interpolation)** Wir assoziieren mit der Zeitreihe u die *Kurve* $f_u : [0,1] \to \mathbb{R}^n$, definiert durch

$$f_u(\alpha) := \begin{cases} u_t & : \alpha = t/T, \text{ für ein } t \in \{0,\dots,T\} \\ I(u,\alpha) & : \text{sonst} \end{cases}$$

   Dabei stellt $I(\cdot,\cdot)$ eine *Interpolationsfunktion* dar, die im einfachsten Fall *linear* ist:

$$I(u,\alpha) := u_t + (\alpha T - t)\cdot(u_{t+1} - u_t), \text{ für dasjenige } t \text{ mit } t \le \alpha T < t+1$$

   Üblich ist aber auch die Anwendung *polinomieller Splines* höheren Grades, um „weichere“ Kurvenverläufe zu erhalten [Sch93].

2. **(Resampling)** Wir gewinnen die längenmodifizierte Sequenz $u^* := (u_t^*)_{0\le t\le T^*}$ aus u durch Setzen von

$$u_t^* := f_u(t/T^*), \text{ für } 0 \le t \le T^*$$

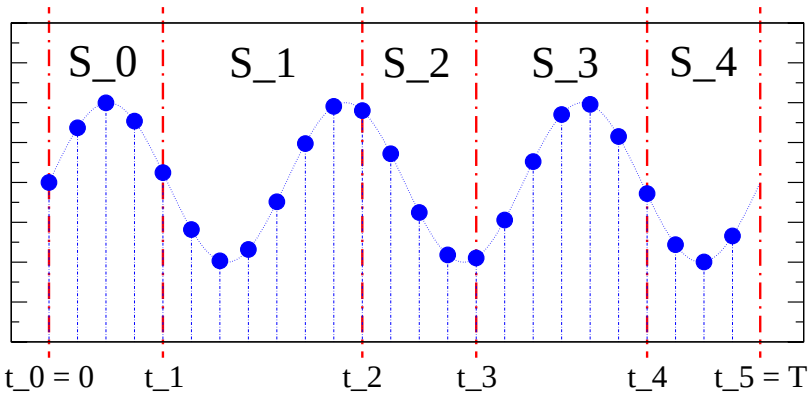

Abbildung 2.14: Segmentierung einer Zeitsequenz (nicht äquidistant)

Offenbar läßt sich dieses Verfahren sowohl für den Fall $T < T^*$ als auch für $T > T^*$ anwenden; für $T = T^*$ ergibt sich $u^* = u$, das Verfahren ist in dieser Hinsicht also „stabil". Jain e.a. [JGC01] verwenden diese Methode in dem von ihnen konzipierten Unterschriftenverifikationssystem.

2.5.2.3.2 Segmentierung Anstatt die Anzahl der Samples einer Sequenz (u_t) zu verändern, kann man alternativ eine Zerlegung von (u_t) in eine gewünschte Anzahl L von *Segmenten*, d.h. aneinander angrenzenden Teilsequenzen durchführen.

Definition 2.1 (Segmentierung). *Sei $(u_t)_{0\le t<T}$ eine Unterschriftensequenz. Eine Folge $s := (t_\lambda)_{1\le\lambda\le L}$, mit $0 \le L \le T$ und $0 < \cdots < t_\lambda < \cdots < t_L < T$, heißt* Segmentierung *von u der Länge L. Mit den Vereinbarungen $t_0 := 0$ und $t_{L+1} := T$ heißt die Folge $S := ((u_t)_{t_\lambda\le t<t_{\lambda+1}})_{0\le\lambda\le L}$ die* Segmentsequenz *von u hinsichtlich s mit den* Segmenten $S_\lambda = (u_t)_{t_\lambda\le t<t_{\lambda+1}}$.

Eine Segmentierung, wie sie schematisch in Abb. 2.14 dargestellt ist, hat gegenüber dem Resampling den Vorteil, daß alle Samples der Ausgangssequenz (u_t) erhalten bleiben, es findet somit im Prinzip kein Informationsverlust statt. Ein Längenvergleich betrifft bei diesem Ansatz aber nun nicht mehr die Länge von (u_t), sondern die Länge der *Segmentsequenz* S, was möglicherweise zu komplizierteren Vergleichsverfahren führt.

Im einfachsten Fall wird eine *äquidistante* Segmentierung mit fest vorgegebener Segmentanzahl L durchgeführt, bei der alle Segmente (nahezu) gleichlang sind: $t_\lambda := [\lambda{\cdot}T/L]$. Für den Fall der eingangs erwähnten Unterschriften $u^{(1)}$ und $u^{(2)}$ mit linearer Dehnung $u^{(2)}_{[\gamma t]} \approx u^{(1)}_t$ wäre eine solche Strategie ausreichend. Allgemeiner anwendbar dürfte eine *„ereignisorientierte"* Segmentierung sein, bei der die Segmentierungsstellen z.B. an *Nulldurchgängen* oder *Maxima* des Sequenzverlaufs gewählt werden. Hiermit ließen sich auch Unterschriften $u^{(1)}$ und $u^{(2)}$ adäquat angleichen, die durch *nichtlineare* zeitliche Dehnungen $\gamma(\cdot)$ auseinander hervorgegangen sind: $u^{(2)}_{[\gamma(t)]} \approx u^{(1)}_t$. In der Realität stehen Unterschriften der gleichen Person aber normalerweise nicht in derart einfach zu beschreibender Weise miteinander in Beziehung, weswegen der Autor in Kapitel 4 ein *adaptives* Segmentierungsverfahren entwickeln wird. Weitere Ansätze zur Segmentierung werden in [LP94] zitiert.

2.5.3 Feature-Extraktion

In aller Regel wird man zwei Unterschriften nicht direkt auf den (vorverarbeiteten) Rohdaten vergleichen, sondern sich stattdessen eine Menge von Kriterien überlegen, gemäß derer man eine Beurteilung durchführen möchte: So wie man Bücher hinsichtlich unterschiedlicher Kriterien miteinander vergleichen kann, beispielsweise bezüglich Textgattung, verwendeter

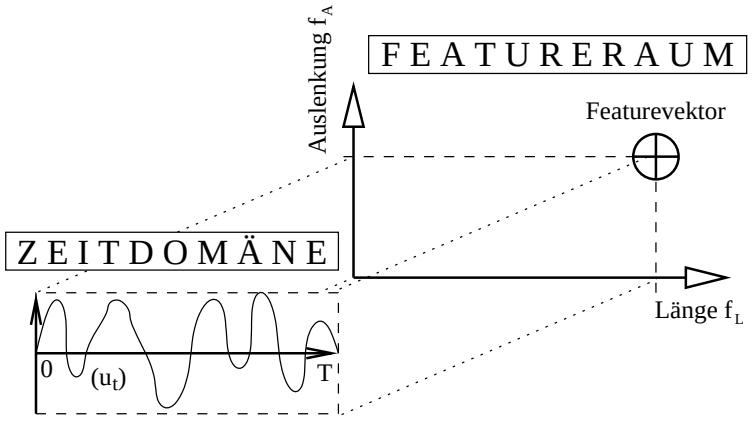

Abbildung 2.15: Globale Features und Featureraum

Sprache oder Farbe des Umschlags (qualitative Merkmale), oder gemäß der Seitenanzahl und der Anzahl an Referenzierungen durch andere Veröffentlichungen (quantitative Merkmale), so ist man auch bei Unterschriften prinzipiell frei hinsichtlich der Wahl geeignet erscheinender *„Features“*, wie wir solcherart Vergleichskriterien von nun an nennen werden.

2.5.3.1 Globale Features

Wir betrachten Mengen $\boldsymbol{f} := \{f_j \mid 1 \leq j \leq n\}$ von Features f_j, sogenannte *„Featuresets“*, wobei wir uns zunächst auf *globale* Features beschränken wollen: Ein *globales* Feature f_j läßt sich als Funktion $f_j : \mathbb{U} \to \mathbb{F}_j$ auffassen, die jeder Unterschrift $u \in \mathbb{U}$ einen Wert $x_j := f_j(u)$ im Wertebereich $\mathbb{F}_j$ zuordnet. Beispiele globaler Features für dynamisch erfaßte Unterschriften $u = (u_t)_{0 \leq t < T}$ sind deren Schreibdauer $f_L(u) := T$ oder die maximale Auslenkung $f_A(u) := \max_t\{u_t\} - \min_t\{u_t\}$ (vgl. Abb. 2.15; in [Roh03] werden weitere Beispiele genannt). Den zum Featureset $\boldsymbol{f}$ gehörenden Gesamtwertebereich $\mathbb{F} := \mathbb{F}_1 \times \cdots \times \mathbb{F}_n$ bezeichnen wir als *„Featureraum“*. Wir werden die Menge $\boldsymbol{f}$ gelegentlich auch als Funktion $\boldsymbol{f} : \mathbb{U} \to \mathbb{F}$ verwenden. [HKS04] beschreibt ein Verfahren, welches dynamisch mit dem BiSP-Pen erfaßte Handschriften auf der Grundlage von mehr als 100 globalen Features analysiert.

2.5.3.2 Vergleich von Unterschriften im Featureraum

Unterschriften sind, wie in Abb. 2.15 zu sehen, bei der Verwendung globaler Features im Featureraum nicht mehr als Zeitsequenzen sondern als Punkte $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$, sogenannte *„Featurevektoren“*, repräsentiert. Die Darstellung im Featureraum $\mathbb{F}$ stellt eine abstrakte Sichtweise dar, die es ermöglicht, Unterschriften ausschließlich hinsichtlich der zuvor durch $\boldsymbol{f}$ festgelegten Kriterien zu vergleichen. Im Idealfall ist das Featureset $\boldsymbol{f}$ so gewählt worden, daß sämtliche Unterschriften einer Person P auf den *gleichen* Punkt $\boldsymbol{x}^{(P)} \in \mathbb{F}$ abgebildet werden, während sich die Bilder der Unterschriften *verschiedener* Personen paarweise unterscheiden. In der Praxis ist allerdings nicht zu erwarten, daß es gelingt ein solch perfektes Featureset $\boldsymbol{f}$ zu generieren. Vielmehr wird man versuchen $\boldsymbol{f}$ so zu wählen, daß zwei Unterschriften der gleichen Person im Featureraum *räumlich eng beieinander* liegen, während die Unterschriften zweier verschiedener Personen einen großen Abstand voneinander haben sollten. Sämtliche Unterschriften einer Person würden also einen *„Punktcluster“* im Featureraum bilden, und die Cluster zweier Personen sollten möglichst klar voneinander getrennte Gebiete einnehmen.

Es sind allerdings nicht alle Featurefunktionen in der Lage, eine derartige „Ähnlichkeitserhaltung“ zu vermitteln; vor allem *ungeordnete Aufzählungsbereiche*, wie z.B. die Farbenmenge $\{\texttt{ROT}, \texttt{GELB}, \texttt{BLAU}\}$, vermögen dies nicht zu leisten, weswegen wir solche Features in dieser Arbeit auch nicht weiter betrachten wollen. Bei *geordneten* Aufzählungbereichen, wie z.B.

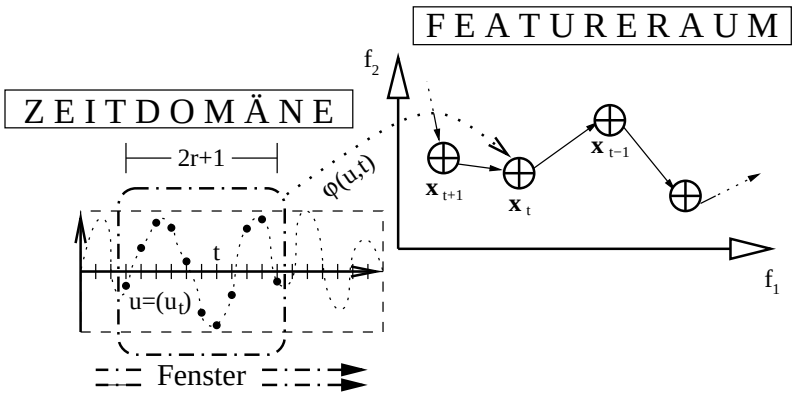

Abbildung 2.16: **Lokale Features und Featurevektor-Zeitsequenzen.** Alle Samples einer Zeitsequenz $u = (u_t)$, welche sich in einem Fenster der Länge $2r + 1$ um eine Zeitstelle t herum befinden, werden durch eine lokale Featurefunktion $\boldsymbol{\varphi}(u, t)$ auf einen Featurevektor $\boldsymbol{x}_t$ im Featureraum abgebildet. Das Fenster überstreicht sukzessive die komplette Zeitreihe u, wodurch im Featureraum eine Zeitsequenz $(\boldsymbol{x}_t)$ entsteht.

Größenbegriffen $\{\texttt{klein}, \texttt{mittel}, \texttt{gross}\}$, gelingt evtl. eine Darstellung durch ganze Zahlen, aber es muß im jeweiligen Anwendungsfall entschieden werden, welchen Zahlen die *vagen* Begriffe „gross“ oder „klein“ entsprechen sollen. Ganze Zahlen, wie sie auch im Beispiel aus 2.5.3.1 zur Messung der Schreibdauer T verwendet wurden, stellen aber auch noch keinen idealen Wertebereich für Featurefunktionen dar, da unser Featureraum idealerweise ein *Vektorraum* sein sollte, um die in der Mathematik üblichen Ähnlichkeitsbegriffe verwenden zu können (vgl. 2.5.5). Wir werden daher von nun an annehmen, daß der Wertebereich unserer Features f_j stets $\mathbb{R}$ ist, und uns Mengen wie $\mathbb{Z}$ (Anzahlen) oder $\mathbb{Q}$ (Fließkommazahlen) entsprechend eingebettet denken.

2.5.3.3 Lokale Features

Die ausschließliche Verwendung globaler Features hat den Nachteil, daß nur *formale* Aspekte, wie beispielsweise die Länge einer Unterschrift, adäquat beurteilt werden können. Für *dynamisch* erfaßte Unterschriften ergibt sich zudem das Problem, daß dadurch *zeitliche Sequenzinformation* und Informationen über die *lokale Dynamik* innerhalb der Zeitreihe verloren gehen.

Ein *„lokales“ Feature* unterscheidet sich von einem globalen dadurch, daß es immer nur *ein* Sample u_t einer Unterschrift u in dessen *lokalem zeitlichen Kontext* erfaßt und auf einen Punkt im Featureraum abbildet. Formal läßt sich eine Menge $\boldsymbol{\varphi} := \{\varphi_1, \ldots, \varphi_n\}$ lokaler Features auffassen als Funktion $\boldsymbol{\varphi} : \mathbb{U} \times \mathbb{Z} \to \mathbb{F}$, welche jedem Sample u_t einer Unterschrift u einen Featurevektor $\boldsymbol{x}_t \in \mathbb{F}$ zuweist: $\boldsymbol{\varphi}(u, t) = \boldsymbol{x}_t$. Dabei läßt sich $\boldsymbol{\varphi}$ prinzipiell durch die mehrfache Anwendung einer Menge *globaler* Features $\boldsymbol{f}$ realisieren vermöge

$$\boldsymbol{\varphi}(u, t) := \boldsymbol{f}(\,(u_{t+\rho})_{-r \le \rho \le +r}\,) \tag{2.13}$$

Durch den *Radius* r wird der besagte lokale Kontext der Stelle t spezifiziert. Anschaulich (vgl. Abb. 2.16) bewegt man dabei ein „Fenster“ der Länge $2r + 1$ über die Zeitsequenz $(u_t)_{0 \le t < T}$ und erhält aus den in diesem Fenster liegenden Samples jeweils einen Vektor $\boldsymbol{x}_t$ im Featureraum. Es ergibt sich mithin erneut eine Zeitsequenz $(\boldsymbol{x}_t)$ gleicher Länge T wie die Ausgangssequenz (u_t),[6] welche die zeitliche Reihenfolge erhält.

[6] Evtl. wird man das linke und rechte Ende der Unterschrift (u_t) gesondert behandeln müssen, da die direkte Anwendung von (2.13) dort formal nicht möglich ist.

Im Prinzip läßt sich wie gesehen jedes globale Feature zur Realisierung eines lokalen Features einsetzen. In der Praxis ist dies allerdings nicht immer sinnvoll. Beispielsweise würde die Anwendung des *Längenfeatures* f_L aus 2.5.3.1 als Ergebnis stets die Länge $2r+1$ des Fensters ergeben. Das Feature f_A, welches die *maximale Auslenkung* innerhalb der gesamten Unterschrift gemessen hatte, ist hingegen auch als lokales Feature von Wert, denn dadurch lassen sich Aussagen über die *lokale Amplitudendynamik* machen. Neben den traditionell zur globalen Messung eingesetzten Features wird es aber auch neue Features geben, die man sinnvollerweise nur für lokale Zwecke einsetzen wird, beispielsweise das *Differenzenfeature* $\varphi_D(u,t) := u_t - u_{t-1}$, welches das *zeitliche Änderungsverhalten* der Unterschrift (u_t) mißt.

2.5.4 Modellgenerierung

Die für die Modellgenerierung bestimmten Unterschriften $u \in \mathbb{U}$ werden von nun an als Sequenzen von *Featurevektoren* $(\boldsymbol{x}_t)_{0 \le t < T}$ im Featureraum $\mathbb{F} = \mathbb{F}_1 \times \cdots \times \mathbb{F}_n$ gemäß 2.5.3.3 betrachtet werden. Die Menge $\mathbb{U}$ wird dabei *sämtliche* Folgen von Vektoren $\boldsymbol{x} \in \mathbb{F}$ enthalten, nicht nur solche, die zu von Menschen geschriebenen Unterschriften gehören. Es ist also $\mathbb{U} = \mathbb{F}^+ = \bigcup_{n=1}^{\infty} \mathbb{F}^n$.

Wir identifizieren eine Person P mit der Menge $P \subseteq \mathbb{U}$ aller von ihr jemals erzeugten Unterschriften, sowie mit der bipolaren Funktion $P : \mathbb{U} \to \{-1, +1\}$, definiert durch

$$P(u) := \begin{cases} +1 & : u \in P \\ -1 & : sonst \end{cases}$$

$P(\cdot)$ ist also die *charakteristische Funktion* der Menge P, und wir werden ohne weitere Ankündigung den Ausdruck 'P' in allen genannten Verwendungsweisen einsetzen. Wir betrachten weiterhin eine Menge $T_P := \{u^{(1)}, \ldots, u^{(m)}\}$ von m *klassifizierten Trainingsmustern*, d.h. für jede Trainingsunterschrift $u^{(i)}$ steht dem zur Modellgenerierung eingesetzten Lernverfahren auch die tatsächliche Klassifikation $P(u^{(i)})$ zur Verfügung. Ziel ist die Generierung eines *Modells* $\mathcal{M}_P$ für P basierend auf T_P.

2.5.4.1 Modelldefinition

Wir werden die Unterschriftenverifikation im weiteren Verlauf als *binäres Klassifikationsproblem* darstellen. Hierzu definieren wir den Begriff des *„Modells"*:

Definition 2.2 (Modell). *Ein* Modell *für die Unterschrift einer Person P ist eine Funktion $\mathcal{M}_P : \mathbb{U} \to \{-1, +1\}$, die für jede Unterschrift $u \in \mathbb{U}$ eine Diagnose '$+1$' (legale Unterschrift) oder '-1' (Fälschung) stellt.*

$\mathcal{M}_P(\cdot)$ ist die charakteristische Funktion der Menge $\mathcal{M}_P := \{u \in \mathbb{U} \mid \mathcal{M}_P(u) = 1\}$; wir werden den Ausdruck '$\mathcal{M}_P$' in beiden Formen verwenden. Bei der Modellgenerierung wird man sich auf drei Hauptaspekte konzentrieren:

1. Man benötigt eine geeignete *Datenstruktur* S, welche alle für die Modellierung relevanten Informationen zur Unterschrift einer Person aus der Trainingsmenge T_P stammend integriert.

2. Man braucht eine *Ähnlichkeitsfunktion* $d : S \times \mathbb{U} \to [0,1]$, die jeder Kombination (s_P, u), bestehend aus einer Unterschrift u und einer konkreten Instanz s_P der Datenstruktur S für eine Person P, einen Ähnlichkeitswert $\alpha \in [0,1]$ zuordnet. Der Wert $d(s_P, u) = 1$ stellt dabei ein Höchstmaß, $d(s_P, u) = 0$ ein Minimum an Ähnlichkeit zwischen der Unterschrift u und den durch s_P repräsentierten Trainingsunterschriften aus T_P dar.

3. Man hat einen *Separierungsthreshold* $\tau_P \in [0,1]$ zu definieren, über den $\mathcal{M}_P$ bestimmt wird durch $\mathcal{M}_P(u) := \mathrm{sgn}(d(s_p,u) - \tau_P)$, mit

$$\mathrm{sgn}(x) := \begin{cases} +1 & : x > 0 \\ -1 & : sonst \end{cases} \qquad \textit{(Vorzeichen-Funktion)} \tag{2.14}$$

Es läßt sich nun der *Trainingsfehler*

$$\varepsilon_{T_P} := \frac{1}{m} \sum_{\substack{u \in T_P \\ \mathcal{M}_P(u) \neq P(u)}} 1 \qquad \textit{(Trainingsfehler)} \tag{2.15}$$

für dieses Modell $\mathcal{M}_P$ und die Trainingsmenge T_P ermitteln, d.h. der prozentuale Anteil an durch das Modell $\mathcal{M}_P$ falschklassifizierten Trainingsunterschriften $u^{(i)}$.

2.5.4.2 Ein Beispielmodell

Wir geben ein einfaches Beispiel eines Modells an für den Fall, daß Unterschriften $u \in \mathbb{U}$ durch *globale* Features gemäß 2.5.3.1 repräsentiert sind. Wir betrachten also Featurevektoren $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{F}$, d.h. wir beschränken uns formal gesehen auf Zeitsequenzen $(\boldsymbol{x}_t)$ der Länge 1. Die Menge $T_P := \{u^{(1)}, \ldots, u^{(m)}\}$ der Trainingsunterschriften bestehe ausschließlich aus Unterschriften von P, wir verwenden in diesem Beispiel für die Modellierung also *keine* Fälschungen.

2.5.4.2.1 Die Datenstruktur 'S' Die Datenstruktur S besteht, neben den Informationen zur Identität der Person P, aus den beiden Vektoren $\boldsymbol{\mu} := (\mu_1, \ldots, \mu_n)^\top \in \mathbb{F}$ und $\boldsymbol{\sigma} := (\sigma_1, \ldots, \sigma_n)^\top \in \mathbb{F}$. Die Idee ist, daß eine Instanz $s_P \in S$ durch $\boldsymbol{\mu}$ die „wahre" Unterschrift der Person P repräsentiert, während durch $\boldsymbol{\sigma}$ die „personenspezifische Variabilität" der Unterschrift in sämtlichen Features erfaßt wird. $\boldsymbol{\mu}$ definieren wir über den *arithmetischen Mittelwert* der Trainingsunterschriften in jedem Feature f_j:

$$\mu_j := \langle x_j^{(i)} \rangle_i \,, \quad 1 \leq j \leq n$$

Unsere Approximation der „wahren" Unterschrift von P entsteht hier gewissermaßen *„rein adaptiv"*: Wir greifen nicht auf irgendwie geartete komplexe Theorien über den Aufbau von Unterschriften zurück, sondern gewinnen unsere Hypothese durch eine einfache statistische Anpassung an das vorhandene Datenmaterial. Völlig theoriefrei arbeiten aber auch wir nicht: Unser Kriterium für die Wahl einer geeigneten Prognose ist die *Minimierung des quadratischen Fehlers* in Bezug auf sämtliche Trainingsunterschriften, denn genau dies leistet der verwendete Mittelwert:

$$\partial_y \sum_{i=1}^{m} (x_j^{(i)} - y)^2 = \sum_{i=1}^{m} -2 \cdot (x_j^{(i)} - y) = 2 \cdot \left[my - \sum_{i=1}^{m} x_j^{(i)} \right] = 0 \iff y = \frac{1}{m} \cdot \sum_{i=1}^{m} x_j^{(i)}$$

Als Variabilitätsmaß für $\boldsymbol{\sigma}$ verwenden wir die komponentenweisen *Standardabweichungen* über die Elemente der Trainingsmenge:

$$\sigma_j := \sqrt{\left\langle (x_j^{(i)} - \mu_j)^2 \right\rangle_i} \,, \quad 1 \leq j \leq n$$

2.5.4.2.2 Die Ähnlichkeitsfunktion 'd' Die Ähnlichkeitsfunktion d definieren wir als

$$d(s_P, u) := d(s_P, (x_1, \ldots, x_n)^\top) := \exp\left[- \left(\rho^{-1} \cdot \langle \frac{x_j - \mu_j}{\sigma_j} \rangle_j \right)^2 \right]$$

Dabei können wir den Quotienten $\delta_j := (x_j - \mu_j)/\sigma_j$ als *relativen Fehler* des Featurewertes x_j in Bezug auf die Modell-Struktur ansehen. Für *Eigenunterschriften* wird sich hierfür zumeist ein *kleiner* Zahlenwert um Eins herum ergeben, denn wir hatten mit der Standardabweichung σ_j die „personenspezifische Variabilität" bezogen auf das j-te Feature gemessen. Für *Fremdunterschriften* läßt sich hingegen ohne genauere Kenntnis der Features f_j keine Aussage über deren zu erwartende Fehlerabweichung treffen. Man darf aber vermuten, daß diese häufig höher liegen wird als bei Originalunterschriften. Die Skalierung mit σ^{-1} gewährleistet ferner, daß im *mittleren Fehler* $\bar{\delta} := \langle \delta_j \rangle_j$ alle Beiträge δ_j in etwa gleichberechtigt vertreten sind (vgl. dies mit der in 2.5.2.2 geführten Diskussion über die *Amplitudennormierung*). Da wir kein spezifisches Wissen über die verwendeten Features haben, nehmen wir an, daß alle Eigenunterschriften $\boldsymbol{x}$ der Person P *zufällig* um die „wahre" Unterschrift $\boldsymbol{\mu}$ herum streuen, was die Wahl einer *gaußischen* Ähnlichkeitsfunktion nahelegt. Offenbar liegt damit $d(s_P, u)$ stets im Intervall $[0, 1]$. Mit dem Parameter $\rho > 0$ läßt sich die *„Toleranz"* ggü. Unterschriften $\boldsymbol{x}$ mit großem mittleren Fehler $\bar{\delta}$ einstellen: Ist ρ groß, so erhalten auch noch Unterschriften mit relativ starker Abweichung vom Mittelvektor $\boldsymbol{\mu}$ hohe Ähnlichkeitsbewertungen.

2.5.4.2.3 Der Separierungsthreshold 'τ_P' Der Separierungsthreshold τ_P wird sinnvollerweise nicht statisch festgelegt, sondern als *Systemparameter* eingesetzt, um die Akzeptanzrate des Verifikationssystems an die jeweiligen Erfordernisse anzupassen. Wir werden auf dieses Thema ausführlich in 2.6 zu sprechen kommen.

2.5.4.3 Andere Modellierungsmethoden

Diverse Modellierungsstrategien wurden in der Vergangenheit untersucht. [Roh03] zitiert Arbeiten, in denen über die Modellierung von Unterschriften mittels *Hidden Markov Models (HMM)* berichtet wird. [LP94] gibt einen Überblick über Systeme, welche *Neuronale Netze* einsetzen. Speziell wird in [MRMK02] die Anwendung eines Netzes vom Typ *ART-2* angegeben.

2.5.4.4 Overfitting und Validierung

Das adaptive Erlernen eines Modells anhand von Trainingsunterschriften kann zu einer *Überanpassung* (engl. *„Overfitting"*, vgl. [Bra95]) an die Daten der Trainingsmenge T_P führen. Ein Indiz dafür, daß ein Modellierungsprozeß zu Overfitting neigt, ergibt sich aus dem Vergleich des Trainingsfehlers ε_{T_P} gemäß (2.15), S. 31, und dem in analoger Weise für Unterschriften $u \notin T_P$ definierten *Generalisierungsfehler* ϵ_P (vgl. 2.6.1.2): Fällt ε_{T_P} signifikant geringer aus als ϵ_P, so ist vermutlich Overfitting aufgetreten. Der eigentliche Zweck des Trainings, aus wenigen realistischen Beispieldaten eine möglichst repräsentative Hypothese zu gewinnen, wurde hier verfehlt.

Algorithmus 2.1 Training mit Validierung

generiere Ausgangsstruktur $s_P^{(0)} \in S$ für P
$\lambda := 0$
repeat
  $\lambda := \lambda + 1$
  generiere Struktur $s_P^{(\lambda)} \in S$ für P
until $\mathrm{e}_V(s_P^{(\lambda)}) > \mathrm{e}_V(s_P^{(\lambda-1)})$

Um dem Effekt des Overfittings während der Trainingsprozedur entgegenzuwirken, verwenden wir eine Teilmenge V der Trainingsmenge T_P als *„Validierungsdaten"*: Das Erlernen des Modells geschieht nun nur noch anhand der verbleibenden Trainingsmenge $T'_P := T_P \backslash V$. In Algorithmus 2.1 werden wie üblich für ein adaptives Lernverfahren sukzessive neue Mo-

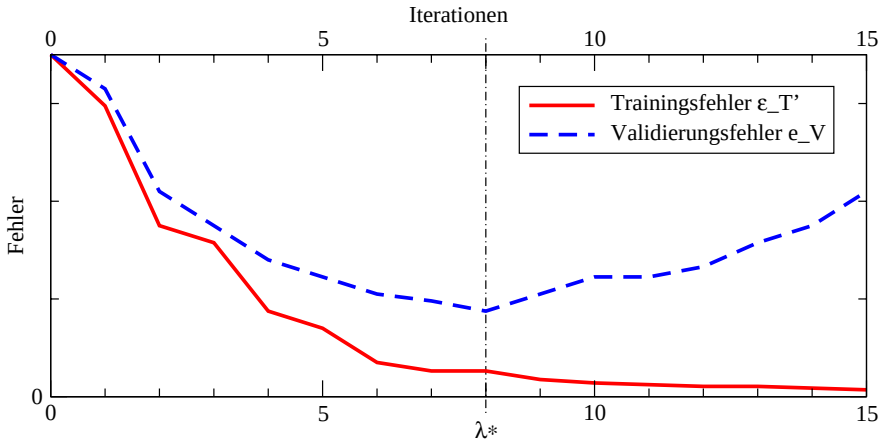

Abbildung 2.17: Trainingsfehler vs. Validierungsfehler

dellinstanzen $s_P^{(\lambda)}$ generiert. Für die nicht zum eigentlichen Training verwendete Validierungsmenge V wird nach jedem Iterationsschritt λ der *„Validierungsfehler“*

$$\mathrm{e}_V(s_P) := \langle 1 - d(s_P, \nu) \rangle_{\nu \in V} \qquad \textit{(Validierungsfehler)} \tag{2.16}$$

mit der Ähnlichkeitsfunktion d berechnet und mit dem entsprechenden Wert des vorangegangenen Schrittes $\lambda - 1$ verglichen.

Abb. 2.17 stellt den typischen Ablauf eines solchen Trainings mit Validierung anhand zweier Kurven dar: Eine Kurve beschreibt den Verlauf des Trainingsfehlers $\varepsilon_{T'_P}$, während die andere Kurve den Validierungsfehler e_V darstellt. Zunächst sinken beide Fehlerwerte für mehrere Modellierungsiterationen λ, was für eine zunehmende Verbesserung des Modells spricht. Von der mit ‘λ^*’ gekennzeichneten Iteration an wächst der Validierungfehler aber wieder, während der Trainingsfehler weiter sinkt. Dies deutet darauf hin, daß die Modellstrukturen $s_P^{(\lambda)}$ mit $\lambda > \lambda^*$ sich zu stark an die Trainingsdaten T'_P anpassen. Damit erscheint es sinnvoll, wie in Algorithmus 2.1 an der Stelle λ^* das Training abzubrechen und $\mathcal{M}_P^{(\lambda^*)}$ als Ergebnismodell auszugeben.

2.5.4.5 Boosting

Ein großes praktisches Problem für die Unterschriftenverifikation ist, daß man für das Enrollment i.a. nur über eine eng begrenzte Anzahl von Trainingsdaten verfügt, insbesondere was die Anzahl an *legalen* Unterschriften betrifft, denn man kann normalerweise nicht von einer zu modellierenden Person große Mengen an Schreibproben einfordern. Wir wollen hier eine Methode namens *„Boosting“* [Sch01] betrachten, welche für eine feste Menge $T_P := \{u^{(1)}, \ldots, u^{(m)}\}$ von klassifizierten Trainingsdaten und einen gegebenen Modellierungsalgorithmus A ein Modell erzeugt, dessen Trainingsfehler ε_{T_P} manchmal unter demjenigen liegen kann, den ein durch A erzeugtes Modell $\mathcal{M}_P$ für eine Person P besitzt, ohne daß dabei die Gefahr von Overfitting gemäß 2.5.4.4 besteht. Es wird dabei keine Modifikation am Verhalten von A oder an den von A generierten Modellen durchgeführt.

Als Beispiel für das Boostingprinzip wollen wir *„Ada-Boost“* [FS97] in Algorithmus 2.2 betrachten, bei dem r Teilmodelle $\mathcal{M}_P^{(k)}$ mit dem Lernalgorithmus A konstruiert und zum Schluß zu einem Gesamtmodell integriert werden (die Forderung, daß r ungerade ist, hat technische Gründe, auf die hier nicht eingegangen werden soll). Dabei wird A nicht die eigentliche Trainingsmenge T_P vorgelegt, sondern jeweils durch zufälliges Ziehen mit Zurücklegen ausgewählte abgeleitete Trainingsmengen $T_P^{(k)}$. Die Auswahl geschieht relativ zu einer Zufallsverteilung D_k dergestalt, daß die Vielfachheit einer Trainingsunterschrift u in $T_P^{(k)}$ in etwa

Algorithmus 2.2 Ada-Boost

Require: $r \in \mathbb{N}$ ungerade

bilde uniforme Startverteilung auf T_P: $D_1(u^{(i)}) := \frac{1}{m}$, $1 \leq i \leq m$

for $k := 1$ to r **do**

bilde Menge $T_P^{(k)}$: m mal Ziehen mit Zurücklegen aus T_P gemäß Verteilung D_k

generiere Modell $\mathcal{M}_P^{(k)}$ durch Anwendung des Modellgenerators A auf $T_P^{(k)}$

setze $\alpha_k := \frac{1}{2}\ln(\frac{1-\varepsilon_k}{\varepsilon_k})$, mit Trainingsfehler $\varepsilon_k := \sum_{u \in T_P} D_k(u)$

bilde Verteilung $D_{k+1}(u^{(i)}) := \varphi_k(u^{(i)}) / \sum_{u \in T_P} \varphi_k(u)$, mit

$$\varphi_k(u) := D_k(u) \cdot \exp(-P(u) \cdot \mathcal{M}_P^{(k)}(u) \cdot \alpha_k)$$

end for

Ergebnismodell: $\mathcal{M}_P(u) := \mathrm{sgn}(\mathcal{M}_P^*(u))$ mit der Vorzeichenfunktion $\mathrm{sgn}(\cdot)$ und mit

$$\mathcal{M}_P^*(u) := \sum_{1 \leq k \leq r} \frac{\alpha_k}{\sum_{1 \leq \lambda \leq r} \alpha_\lambda} \cdot \mathcal{M}_P^{(k)}(u)$$

seine Wahrscheinlichkeit $D_k(u)$ widerspiegeln wird. Für jedes durch A generierte Modell $\mathcal{M}_P^{(k)}$ wird der zugehörige Trainingsfehler $\varepsilon_k := \sum_{u \in T_P} D_k(u)$ ermittelt. Es sei verdeutlicht, daß der Trainingsfehler hier *nicht* auf der abgeleiteten Trainingsmenge $T_P^{(k)}$, sondern auf der Ausgangsmenge T_P berechnet wird: Wir wollen die Leistung der Modelle $\mathcal{M}_P^{(k)}$ hinsichtlich der gleichen Referenzmenge beurteilen, nicht hinsichtlich der Spezialmengen, die im Prinzip noch nicht einmal über gemeinsame Elemente verfügen müssen.

Die Zufallsverteilung D_{k+1} wird durch Modifikation von D_k derart bestimmt, daß die neue Trainingsmenge $T_P^{(k+1)}$ zu einem großen Teil aus solchen Mustern $u \in T_P$ bestehen wird, die durch das Vorgängermodell $\mathcal{M}_P^{(k)}$ *mißklassifiziert* wurden: Wenn wir annehmen, daß $\mathcal{M}_P^{(k)}$ einen Trainingsfehler von $\varepsilon_k \leq 1/2$ aufweist, so gilt $\alpha_k := \frac{1}{2}\ln(\frac{1-\varepsilon_k}{\varepsilon_k}) > 0$,[7] sodaß der Faktor $\exp(-\mathcal{M}_P^{(k)} \cdot P(u) \cdot \alpha_k)$ für $\mathcal{M}_P^{(k)} \neq P(u)$ groß wird ($\mathcal{M}_P^{(k)} \cdot P(u) = -1$), während er bei korrekten Diagnosen klein wird. Dabei werden Fehlklassifizierungen umso stärker „belohnt“, je kleiner der Trainingsfehler ε_k ausfällt (α_k wird groß). Algorithmus A darf sich also im k+1-ten Schritt weitgehend auf die jeweils verbliebenen „schwierigen“ Trainingsunterschriften konzentrieren und für diese ein speziell angepaßtes Modell schaffen. Aber um eine Überanpassung auf sehr wenige Muster zu vermeiden, sollte das Mengenverhältnis zwischen falsch und korrekt klassifizierten Mustern in $T_P^{(k+1)}$ halbwegs ausgewogen sein: Die im Algorithmus verwendete Funktion $\varphi_k(\cdot)$ ist gerade so gewählt, daß die im k-ten Schritt fehlklassifizierten Muster einen Gesamtbeitrag zu D_{k+1} von genau 50% leisten [Sch01], sprich

$$\mathrm{Ws}_{D_{k+1}}[\mathcal{M}_P^{(k)}(u) \neq P(u)] := \sum_{\substack{u \in T_P \\ \mathcal{M}_P^{(k)}(u) \neq P(u)}} D_{k+1}(u) = 0.5$$

Das Ergebnismodell $\mathcal{M}_P$ setzt sich aus den einzelnen Teilmodellen $\mathcal{M}_P^{(k)}$ zusammen, indem es deren Diagnose $\mathcal{M}_P^{(k)}(u)$ für eine vorgelegte Unterschrift $u \in \mathbb{U}$ nach „Zuverlässigkeit“ gewichtet aufsummiert: Ein zuverlässiges Modell zeichnet sich hierbei durch einen geringen Trainingsfehler ε_k und damit durch einen hohen Wert für α_k aus. Die Verwendung der Vorzeichenfunktion ‘$\mathrm{sgn}(\cdot)$’ (vgl. (2.14), S. 31) gewährleistet, daß das Ergebnis entweder $+1$ oder -1 ist.

[7] Den „Idealfall“ ‘$\varepsilon_k = 0$’ müssen wir hier gesondert behandeln, da α_k hierfür nicht definiert ist. Man wird hier eine pragmatische Lösung wählen, z.B. indem man $\varepsilon_k^* := \max\{\varepsilon_k, \frac{1}{m}\}$ anstelle von ε_k verwendet.

Wir betrachten nun den speziellen Fall, daß sämtliche vom Lernalgorithmus A generierten Teilmodelle $\mathcal{M}_P^{(k)}$ einen Trainingsfehler $\varepsilon_k < 1/2 - \gamma$ aufweisen, mit einer Konstante $\gamma > 0$, und zeigen, daß sich unter diesen Umständen der *Gesamttrainingsfehler*

$$\varepsilon^* := \mathrm{Ws}_{U(T_P)}[\mathcal{M}_P(u) \cdot P(u) < 0] := \frac{1}{m} \sum_{\substack{u \in T_P \\ \mathcal{M}_P(u) \neq P(u)}} 1$$

(mit der Gleichverteilung $U(T_P)$ auf der Trainingsmenge) tatsächlich reduziert.

Satz 2.2. *Sei T_P eine Menge klassifizierter Trainingsmuster und A ein Lernalgorithmus, mit dessen Hilfe r Teilmodelle $\mathcal{M}_P^{(k)}$ mit Trainingsfehler $\varepsilon_k < 1/2 - \gamma$, für $\gamma > 0$, durch Algorithmus 2.2 erzeugt werden. Dann gilt für jedes $\theta \in \mathbb{R}$*

$$\mathrm{Ws}_{U(T_P)}\left[P(u) \cdot \sum_{1 \leq k \leq r} \alpha_k \, \mathcal{M}_P^{(k)}(u) \leq \theta \cdot \sum_{1 \leq k \leq r} \alpha_k\right] \leq \left(\sqrt{(1-2\gamma)^{1-\theta} \cdot (1+2\gamma)^{1+\theta}}\right)^r$$

mit der Gleichverteilung $U(T_P)$ auf den Trainingsmustern und den Werten $\alpha_k := \frac{1}{2}\ln(\frac{1-\varepsilon_k}{\varepsilon_k})$.

Beweis. [Sch01] □

Offenbar gilt mit den Benennungen aus Algorithmus 2.2

$$P(u) \cdot \sum_{1 \leq k \leq r} \alpha_k \, \mathcal{M}_P^{(k)}(u) \leq \theta \cdot \sum_{1 \leq k \leq r} \alpha_k \iff P(u) \cdot \mathcal{M}_P^*(u) \leq \theta$$

sodaß mit $\theta := 0$ und der Tatsache, daß

$$P(u) \cdot \mathcal{M}_P(u) = -1 \iff P(u) \cdot \mathcal{M}_P^*(u) \leq 0$$

nunmehr aus Satz 2.2 für den Gesamttrainingsfehler ε^* folgt:

$$\varepsilon^* = \mathrm{Ws}_{U(T_P)}[P(u) \cdot \mathcal{M}_P(u) < 0] \leq \left(\sqrt{(1-2\gamma) \cdot (1+2\gamma)}\right)^r = \left(\sqrt{1-4\gamma^2}\right)^r =: (\rho(\gamma))^r$$

mit $\rho(\gamma) := 1 - 4\gamma^2 < 1$, d.h. ε^* strebt mit wachsender Anzahl r an Iterationen exponentiell gegen 0.

Trotz des sinkenden Trainingsfehlers neigt Ada-Boost aber *nicht* zu Overfitting. Stattdessen läßt sich zeigen, daß die Anwendung dieser Methode auch eine Verbesserung des *Generalisierungsfehlers* mit sich bringt. Eine mathematische Behandlung dieses Aspekts erbringt [Sch01].

Die offensichtlichen Nachteile von Ada-Boost sind der erhöhte Zeitbedarf zur Generierung der r Teilmodelle $\mathcal{M}_P^{(k)}$, der Platzbedarf für deren Speicherung im Gesamtmodell $\mathcal{M}_P$ und die Dauer zur Berechnung aller Teilergebnisse $\mathcal{M}_P^{(k)}(u)$. Dabei lassen sich aber zumindest die beiden genannten Laufzeitprobleme durch Parallelisierung der Teilmodellbildung und der Teilergebnisberechnung in den Griff bekommen, zumal in der Praxis die Anzahl r der Teilmodelle wegen des exponentiellen Absinkens des Trainingsfehlers vermutlich nicht allzu hoch gewählt werden muß, um gute Ergebnisse zu erhalten.

2.5.5 Vergleichsmethoden

Dieser Abschnitt hat den Vergleich zweier in einem Featureraum $\mathbb{F}$ repräsentierter Unterschriften zum Gegenstand, und damit die Kernaufgabe des *Verifikationsprozesses* gemäß 2.5.1. Ziel ist die Bestimmung einer geeigneten *Ähnlichkeitsfunktion* $d(\cdot)$, wie sie bei der Modelldefinition in 2.5.4.1 eingeführt wurde.

2.5.5.1 Vergleiche zwischen Featurevektoren

Ein Vorteil globaler Features ist, daß durch sie repräsentierte Unterschriften $u \in \mathbb{U}$ als *einzelne Punkte* $\boldsymbol{x}$ im Featureraum $\mathbb{F}$ repräsentiert werden (2.5.3.2). Da wir zudem angenommen hatten, daß für die Wertebereiche $\mathbb{F}_j$ aller unserer Features f_j stets $\mathbb{F}_j = \mathbb{R}$ gilt, haben wir mit $\mathbb{F} = \mathbb{R}^n$ einen *Vektorraum* über $\mathbb{R}$ vorliegen, für dessen Elemente wir *Normen* definieren können [SW90].

Definition 2.3 (Norm). *Sei* $(V, +)$ *ein Vektorraum über einem Körper* $(\mathbb{K}, +, \cdot)$. *Eine* Norm $\|\cdot\|$ *ist eine Funktion* $\|\cdot\| : V \to \mathbb{K}$ *mit folgenden Eigenschaften für beliebige* $\boldsymbol{x}, \boldsymbol{x}' \in V$ *und* $c \in \mathbb{K}$*:*

$$\begin{aligned}
&\|\boldsymbol{x}\| \geq 0 \quad \textit{und} \quad \|\boldsymbol{x}\| = 0 \iff \boldsymbol{x} = \mathbf{0} \\
&\|c \cdot \boldsymbol{x}\| = |c| \cdot \|\boldsymbol{x}\| \\
&\|\boldsymbol{x} + \boldsymbol{x}'\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{x}'\|
\end{aligned}$$

Aus einer Norm $\|\cdot\|$ für den Featureraum $\mathbb{F}$ läßt sich eine *Metrik* $\mathrm{d}_{\|\cdot\|} : \mathbb{F}^2 \to \mathbb{R}$ gewinnen:

$$\mathrm{d}_{\|\cdot\|}(\boldsymbol{x}, \boldsymbol{x}') := \|\boldsymbol{x} - \boldsymbol{x}'\| \qquad \textit{(Metrik)} \tag{2.17}$$

Deren wesentliche Eigenschaften sind gegeben durch

$$\begin{aligned}
&\mathrm{d}_{\|\cdot\|}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{d}_{\|\cdot\|}(\boldsymbol{x}', \boldsymbol{x}) && \text{(Symmetrie)} \\
&\mathrm{d}_{\|\cdot\|}(\boldsymbol{x}, \boldsymbol{x}') \geq 0 \quad \text{und} \quad \mathrm{d}_{\|\cdot\|}(\boldsymbol{x}, \boldsymbol{x}) = 0 \iff \boldsymbol{x} = \boldsymbol{x}' && \text{(positive Definitheit)} \\
&\mathrm{d}_{\|\cdot\|}(\boldsymbol{x}, \boldsymbol{x}'') \leq \mathrm{d}_{\|\cdot\|}(\boldsymbol{x}, \boldsymbol{x}') + \mathrm{d}_{\|\cdot\|}(\boldsymbol{x}', \boldsymbol{x}'') && \text{(Dreiecksungleichung)}
\end{aligned}$$

Sobald wir eine Metrik für $\mathbb{F}$ besitzen, liegt es nahe, den Grad der *Ähnlichkeit* zwischen zwei Unterschriften u und u' aus $\mathbb{U}$ quantitativ über die Entfernung $\mathrm{d}_{\|\cdot\|}(\boldsymbol{x}, \boldsymbol{x}')$ zwischen den entsprechenden Featurevektoren $\boldsymbol{x}$ und $\boldsymbol{x}'$ in $\mathbb{F}$ auszudrücken:[8]

$$\mathrm{sim}_{\mathbb{U}}(u, u') := \mathrm{d}_{\|\cdot\|}(\boldsymbol{f}(u), \boldsymbol{f}(u')) \qquad \textit{(Ähnlichkeitsbewertung)}$$

Wir können uns damit in dieser Erörterung auf die Diskussion geeigneter Normen konzentrieren.

Es kommen verschiedene Normen für die Unterschriftenverifikation in Frage. Da wir wie erwähnt $\mathbb{F} = \mathbb{R}^n$ annehmen, ist die Verwendung der *Euklidischen Norm*

$$\|\boldsymbol{x}\|_2 := \sqrt{\sum_{1 \leq j \leq n} x_j^2} \qquad \textit{(Euklidische Norm)} \tag{2.18}$$

naheliegend, mit welcher man in geometrischen Anwendungen üblicherweise den Abstand eines Punktes $\boldsymbol{x}$ zum Ursprung $\mathbf{0}$ mißt. Alternativ kann man die sog. L_1*-Norm*

$$\|\boldsymbol{x}\|_1 := \sum_{1 \leq j \leq n} |x_j| \qquad \textit{(L_1-Norm)}$$

verwenden, welche evtl. bei diskretwertigen Features zu bevorzugen ist und sich etwas effizienter berechnen läßt.

Die genannten Normen lassen *sämtliche* Features in das Gesamtergebnis mit einfließen. Eine Strategie könnte hingegen sein, lediglich diejenigen Features mit den minimalen oder

[8] Die in 2.5.4.1 betrachteten Ähnlichkeitsfunktionen $d(\cdot, \cdot)$ hatten den Wertebereich $[0, 1]$. Wir ersparen uns hier eine Diskussion darüber, wie man den Bereich $\mathbb{R}_0^+$ in geeigneter Weise auf $[0, 1]$ transformieren kann. Ein Beispiel war für das Modellierungsbeispiel in 2.5.4.2.2 angegeben worden.

maximalen Beiträgen zu verwenden, um unempfindlicher gegenüber „Ausreißerwerten“ in einer der beiden Richtungen zu sein. Den Extremfall dieses Ansatzes stellt die *Maximumsnorm* dar

$$\|\boldsymbol{x}\|_\infty := \max_{1\leq j\leq n}(|x_j|) \qquad \textit{(Maximumsnorm)}$$

bei der stets nur der Betrag des *größten* Featurewertes zur Bewertung verwendet wird. Mehr Flexibilität bringt die Verwendung eines *k-Majoritäts-Bewerters*

$$\sum_{j=1}^{k}|x_{\pi(j)}|\,, \quad \pi\in S_n \text{ mit } x_{\pi(1)}>\cdots>x_{\pi(n)} \qquad \textit{(Majoritätsbewerter)}$$

welcher sich als L_1-Norm für den Teilraum der Features mit den k größten Werten interpretieren läßt.

2.5.5.2 Vergleich von Vektor-Sequenzen

Bei der Verwendung *lokaler* Features entstehen *Folgen* von Featurevektoren (vgl. 2.5.3.3). Die in diesem Abschnitt vorgestellte Methode führt den Vergleich zweier Vektorsequenzen $u=(\boldsymbol{x}_t)_{0\leq t<T}$ und $u'=(\boldsymbol{x}'_t)_{0\leq t<T'}$ auf den Vergleich einzelner *Paare von Vektoren* gemäß 2.5.5.1 zurück. Es wird dabei versucht zwei Sequenzen dergestalt aneinander auszurichten, daß sich „zueinander passende“ Vektoren gegenüberliegen. Aus den einzelnen Distanzen der so entstandenen Vektorpaare kann dann eine *Gesamtähnlichkeit* $\mathrm{sim}(u,u')$ für die beiden Unterschriftensequenzen berechnet werden, die, so die Hoffnung, für zwei Unterschriften der *gleichen* Person regelmäßig größer ausfallen wird als für zwei Fremdunterschriften. Dieses Verfahren des *„Sequence-Alignment“* besitzt neben seiner Nützlichkeit für die Unterschriftenverifikation (Referenzen in [LP94]) auch Anwendung in verschiedenen anderen Bereichen, u.a. in der Bioinformatik, wo es zum Vergleich von Gensequenzen eingesetzt wird [Pev00].

2.5.5.2.1 Sequenz-Alignment Die folgende Diskussion orientiert sich an [Pev00]. Wir legen als erstes fest, was wir unter einem vektorweisen aneinander Ausrichten, einem sog. *„Alignment“*, verstehen wollen.

Definition 2.4 (Alignment). *Gegeben sei eine Folge $(\boldsymbol{x}_t)_{0\leq t<T}$ sowie ein nicht in $(\boldsymbol{x}_t)$ auftretendes Element ‘$\perp$’. Als* Dehnung *der Länge $L\geq T$ von $(\boldsymbol{x}_t)$ bezeichnen wir eine Folge $(\hat{\boldsymbol{x}}_t)_{0\leq t<L}$, die aus $(\boldsymbol{x}_t)$ durch Einfügung von $(L-T)$-vielen Instanzen von $\perp$ an beliebigen Positionen entstanden ist, d.h. $(\hat{\boldsymbol{x}}_t)$ ist von der Form*

$$(\hat{\boldsymbol{x}}_t)_{0\leq t<L}=\perp\ldots\perp\boldsymbol{x}_0\perp\ldots\perp\boldsymbol{x}_t\perp\ldots\perp\boldsymbol{x}_{T-1}\perp\ldots\perp$$

Ein Alignment *A zweier Folgen $(\boldsymbol{x}_t)_{0\leq t<T}$ und $(\boldsymbol{x}'_t)_{0\leq t<T'}$ der Länge $L\geq\max\{T,T'\}$ ist eine Paarfolge $((\hat{\boldsymbol{x}}_t,\hat{\boldsymbol{x}}'_t))_{0\leq t<L}$, deren Glieder durch Dehnungen $(\hat{\boldsymbol{x}}_t)$ von $(\boldsymbol{x}_t)$ und $(\hat{\boldsymbol{x}}'_t)$ von $(\boldsymbol{x}'_t)$ der Länge L mit dem gemeinsamen Füllelement $\perp$ gegeben sind, wobei in A keine Paare ‘$(\perp,\perp)$’ auftreten.*

Abb. 2.18 stellt ein Alignment graphisch für zwei Folgen binärer Tripel dar, wobei versucht wird, *identische* Vektoren einander zuzuordnen. Eine für unsere Zwecke wesentliche Folgerung von Def. 2.4 ist, daß bei der Bildung der Vektorpaare keine *Überkreuzungen* auftreten dürfen, daß also z.B. die Paare $(\boldsymbol{x}_{t_1},\boldsymbol{x}'_{t_2})$ und $(\boldsymbol{x}_{t_2},\boldsymbol{x}'_{t_1})$ nicht gemeinsam im Alignment vorkommen. Zwei Unterschriften wollen wir nämlich nur dann als ähnlich ansehen, wenn sie aus möglichst vielen einander ähnlichen Segmenten in der (weitgehend) gleichen relativen Reihenfolge bestehen. Dies ist der Grund, warum die beiden mit ‘*’ markierten Vektoren nicht miteinander verknüpft wurden. Wir werden es allerdings in gewissen Grenzen tolerieren, daß gelegentlich ein Segment nur in einer der beiden Sequenzen auftritt, was sich in obiger Definition

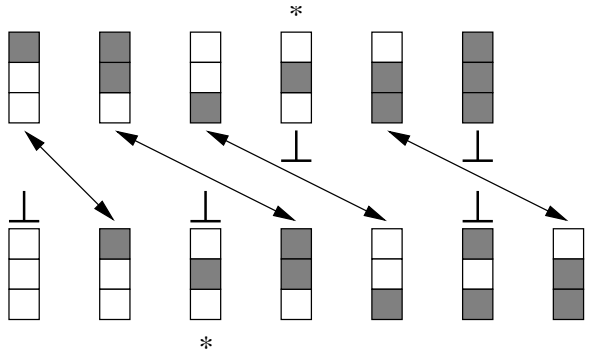

Abbildung 2.18: Alignment zwischen zwei Folgen

durch erlaubte Paare der Form '$(\boldsymbol{x}_t, \perp)$' bzw. '$(\perp, \boldsymbol{x}'_t)$' ausdrückt, und im Schaubild durch die Markierung '$\perp$' anstelle eines abgehenden Verknüpfungspfeils. Dies ist allein schon deswegen nötig, weil die Ausgangssequenzen $(\boldsymbol{x}_t)_{0\leq t<T}$ und $(\boldsymbol{x}'_t)_{0\leq t<T'}$ von unterschiedlicher Länge sein können ($T \neq T'$). Aber auch die Zulassung längerer Teilabschnitte $(\boldsymbol{x}_t, \perp), \ldots, (\boldsymbol{x}_{t+k}, \perp)$, $k \geq 1$, erscheint angebracht: Sollten manche Unterschriftenproben nämlich größere „Schmutzbereiche" aufweisen, so sollten sich dort nicht erzwungenermaßen Partnervektoren anfinden, welche eigentlich besser zu anderen Stellen passen würden. Explizit ausgeschlossen sind jedoch Paare der Form '$(\perp, \perp)$', denn für ein Unterschriften-Alignment macht es nach Ansicht des Autors keinen Sinn, Lücken zwischen zwei Segmenten miteinander zu vergleichen, d.h. solche Paare könnten stets ohne Verlust weggelassen werden. Da somit also in jedem Paar eines Alignments A mindestens ein Segmentglied aus $(\boldsymbol{x}_t)$ oder $(\boldsymbol{x}'_t)$ auftreten muß, ergibt sich für die Länge L eines Alignments die Beziehung

$$\max(T, T') \leq L \leq T + T'$$

Für die Paare $(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}'_t)$ eines Alignments kann man nun einen *Einzelfehler*

$$\epsilon_t := \epsilon(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}'_t) := \mathrm{d}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}'_t) \qquad \textit{(Einzelfehler)} \tag{2.19}$$

angeben, mit einer geeignet zu wählenden Metrik d im Sinne von 2.5.5.1, sofern es sich bei $\hat{\boldsymbol{x}}_t$ und $\hat{\boldsymbol{x}}'_t$ in beiden Fällen um Glieder aus den Ausgangssequenzen handelt. Gesondert zu behandeln sind die Paare '$(\boldsymbol{x}_t, \perp)$' bzw. '$(\perp, \boldsymbol{x}'_t)$', die wir *„Gap-Paare"* nennen wollen. Geeignete Werte für den *„Gap-Fehler"* $\epsilon_\perp$ sind problemspezifisch zu wählen, zwei Überlegungen dürften aber beim Vergleichen von Unterschriften zumeist zutreffen: *Erstens* erscheint es sinnvoll, auch für Gap-Paare einen Fehler $\epsilon_t > 0$ zu definieren, da es inhaltlich betrachtet i.d.R. als Nachteil gewertet werden kann, wenn es zu einem Vektor einer Sequenz keinen passenden Partner gibt. *Zweitens* erscheint es aus Symmetriegründen sinnvoll, den Paaren $(\boldsymbol{x}, \perp)$ und $(\perp, \boldsymbol{x})$ den *gleichen* Fehlerwert ϵ zuzuordnen, da sich durch Vertauschen zweier Sequenzen u und u' keine Veränderung des Alignments und damit der Ähnlichkeit ergeben sollte.

Der *Gesamtfehler* ϵ_A eines Alignments A zweier Sequenzen $(\boldsymbol{x}_t)$ und $(\boldsymbol{x}'_t)$ ergibt sich nun im einfachsten Fall als Kombination der Einzelfehler ϵ_t:

$$\epsilon_A := \langle \epsilon_t \rangle_t \qquad \textit{(Gesamtfehler)}$$

Ein *minimales Alignment* $A_{\min}$ ist ein Alignment mit *minimalem Gesamtfehler* $\epsilon_{\min}$ über alle möglichen Alignments beliebiger Länge L für $(\boldsymbol{x}_t)$ und $(\boldsymbol{x}'_t)$.

2.5.5.2.2 Dynamic-Time-Warping (DTW) Für eine systematische Bestimmung des minimalen Fehlers $\epsilon_{\min}$ gehen wir von den *Prefixsequenzen* $(\boldsymbol{x}_t)_{0\leq t\leq i}$ und $(\boldsymbol{x}'_t)_{0\leq t\leq j}$, $0 \leq i < T$ und $0 \leq j < T'$, aus und bestimmen für diese Zeitreihen ein minimales Alignment $A^{i,j}_{\min}$ mit dem zugehörigen Gesamtfehler $\epsilon^{i,j}_{\min}$. Wir haben drei Fälle zu unterscheiden [Pev00]:

1. $A^{i,j}_{\min}$ endet mit dem Paar $a_l := (\boldsymbol{x}_i, \boldsymbol{x}'_j)$. Dann stellt der Prefix $A^{i-1,j-1} := (a_0, \ldots, a_{l-1})$ ein Alignment für die Prefixsequenzen $(\boldsymbol{x}_t)_{0\le t\le i-1}$ und $(\boldsymbol{x}'_t)_{0\le t\le j-1}$ dar. $A^{i-1,j-1}$ muß ein *minimales* Alignment sein, sonst ließe sich ein Alignment $\tilde{A}^{i-1,j-1}$ zu $(\boldsymbol{x}_t)_{0\le t\le i-1}$ und $(\boldsymbol{x}'_t)_{0\le t\le j-1}$ finden mit $\epsilon_{\tilde{A}^{i-1,j-1}} < \epsilon_{A^{i-1,j-1}}$, und damit wäre $\epsilon_{\tilde{A}^{i-1,j-1}} + \epsilon(a_l) < \epsilon_{A^{i-1,j-1}} + \epsilon(a_l) = \epsilon^{i,j}_{\min}$, im Widerspruch zur Annahme, daß $A^{i,j}_{\min}$ bereits ein minimales Alignment war. Es ist also:

$$A^{i,j}_{\min} = (\, A^{i-1,j-1}_{\min}, (\boldsymbol{x}_i, \boldsymbol{x}'_j)\,)$$

2. Das letzte Paar von $A^{i,j}_{\min}$ ist $a_l := (\boldsymbol{x}_i, \perp)$. Dann geht a_l ein minimales Alignment für die Sequenzen $\boldsymbol{x}_{i-1}$ und $\boldsymbol{x}'_j$ voraus:

$$A^{i,j}_{\min} = (\, A^{i-1,j}_{\min}, (\boldsymbol{x}_i, \perp)\,)$$

3. Eine analoge Überlegung wie zuvor führt für ein Schlußpaar $a_l = (\perp, \boldsymbol{x}'_j)$ auf

$$A^{i,j}_{\min} = (\, A^{i,j-1}_{\min}, (\perp, \boldsymbol{x}'_j)\,)$$

Zur Bestimmung von $A^{i,j}_{\min}$ werden wir nun denjenigen dieser drei Fälle auswählen, für den der Gesamtfehler minimiert wird, der sich damit, unter Hinzunahme zweckmäßig erscheinender Rekursionsverankerungen, wie folgt berechnet:

$$\epsilon^{i,j}_{\min} := \begin{cases} 0 & : i = j = 0 \\ \sum_{\lambda=1}^{i} \epsilon(\boldsymbol{x}_\lambda, \perp) & : j = 0, i \ge 1 \\ \sum_{\lambda=1}^{j} \epsilon(\perp, \boldsymbol{x}'_\lambda) & : i = 0, j \ge 1 \\ \min\left\{ \epsilon^{i-1,j-1}_{\min} + \epsilon(\boldsymbol{x}_i, \boldsymbol{x}_j), \epsilon^{i-1,j}_{\min} + \epsilon(\boldsymbol{x}_i, \perp), \epsilon^{i,j-1}_{\min} + \epsilon(\perp, \boldsymbol{x}_j) \right\} & : \text{sonst} \end{cases}$$

Das gesuchte minimale Alignment $A_{\min}$ für die Unterschriften u und u' und der zugehörige Gesamtfehler $\epsilon_{\min}$ ergeben sich nach diesen Vorarbeiten als Spezialfall:

$$A_{\min} = A^{T,T'}_{\min}$$
$$\epsilon_{\min}(u, u') = \epsilon^{T,T'}_{\min}$$

Algorithmus 2.3 Dynamic-Time-Warping (DTW)

lege Tabelle E der Größe $(T+1)\cdot(T'+1)$ an
$E_{0,0} := 0$
for $i = 1$ to T **do**
  $E_{i,0} := E_{i-1,0} + \epsilon(\boldsymbol{x}_i, \perp)$
end for
for $j = 1$ to T' **do**
  $E_{0,j} := E_{0,j-1} + \epsilon(\perp, \boldsymbol{x}_j)$
end for
for $i = 1$ to T **do**
  for $j = 1$ to T' **do**
    $E_{i,j} := \min\{\, E_{i-1,j-1} + \epsilon(\boldsymbol{x}_i, \boldsymbol{x}_j),\ E_{i-1,j} + \epsilon(\boldsymbol{x}_i, \perp),\ E_{i,j-1} + \epsilon(\perp, \boldsymbol{x}_j)\,\}$
  end for
end for
Ausgabe: $E_{T,T'}$

Die hier vorgestellte systematische Charakterisierung des minimalen Alignments läßt sich durch den *„Dynamic-Time-Warping" (DTW)* genannten Algorithmus 2.3 effizient umsetzen.

Der Algorithmus arbeitet korrekt, da in jedem Schritt nur solche Zellen $E_{i,j}$ ausgelesen werden, deren Inhalt bereits zuvor berechnet wurde. Die Laufzeit bestimmt sich über die beiden verschachtelten FOR-Schleifen und beträgt daher $O(T \cdot T')$. Der Speicherbedarf ist durch die Größe der verwendeten Tabelle E gegeben und beträgt somit $O(T \cdot T')$.

Angegeben ist in Algorithmus 2.3 nur, wie der Fehler $\epsilon_{\min}(u, u')$ für zwei Unterschriften u und u' berechnet wird. Um auch das *Alignment* $A_{\min}$ zu bestimmen, muß man sich zusätzlich mit jedem Tabelleneintrag $E_{i,j}$ merken, über welche der drei Zellen $E_{i-1,j-1}$, $E_{i,j-1}$ und $E_{i-1,j}$ der geringste Fehlerwert berechnet wurde. Man startet dann zum Schluß bei der Zelle $E_{T,T'}$ und traversiert die Tabelle entlang der gespeicherten Verweise bis zur Zelle $E_{0,0}$. Die auf diesem Weg besuchten Indexpaare (i, j) ergeben (nach Invertierung der Folge) das gesuchte minimale Alignment. Die Laufzeitkomplexität verbleibt bei $O(T \cdot T')$.

2.5.5.2.3 Platzeffiziente Ähnlichkeitsberechnung Für die Bestimmung der Ähnlichkeitsbewertung zweier Unterschriften u und u' mittels Sequence-Alignment genügt es, den minimalen Fehler $\epsilon_{\min}(u, u')$ zu ermitteln; eine zusätzliche Berechnung des minimalen Alignments selbst, wie zum Ende des vorangegangenen Abschnitts besprochen, ist hierfür nicht nötig. Unter diesen Umständen läßt sich eine von Algorithmus 2.3 verschiedene Berechnungsmethode angeben, welche – bei gleichbleibender Laufzeit – mit Platz $O(T + T')$ auskommt. Ein entsprechender Algorithmus wird in [Pev00] beschrieben.

2.6 Evaluation von Verifikationssystemen

In diesem Abschnitt beschäftigen wir uns mit der Frage, wie die Leistungsfähigkeit eines Verifikationssystems für Unterschriften, wie es in 2.5 skizziert worden ist, beurteilt werden kann. Wir werden als erstes auf diverse Qualitätsmaße und deren Messung zu sprechen kommen, und danach in 2.6.2 die Auswahl einer geeigneten Testdatenmenge diskutieren.

2.6.1 Leistungsmessung

Unterschriftenverifikationssysteme lassen sich als spezielle *Diagnosesysteme* auffassen:[9] Jede überprüfte Unterschrift $u \in \mathbb{U}$ kann bezogen auf eine Person P entweder *legal* ($P(u) = 1$) sein oder eine *Fälschung* ($P(u) = -1$) darstellen. Aufgabe des Systems ist die Stellung einer Diagnose $\mathcal{M}_P(u) = 1$ oder $\mathcal{M}_P(u) = -1$ mittels des für P generierten Modells $\mathcal{M}_P$ (vgl. 2.5.4.1). Ein leistungsstarkes System wird sich dadurch auszeichnen, daß es selten eine unkorrekte Diagnose $\mathcal{M}_P(u) \neq P(u)$ stellt. In diesem Abschnitt werden verschiedene Maße vorgestellt, mit denen sich die Leistung eines Verifikationssystems quantitativ bestimmen läßt.

2.6.1.1 Fehlerraten

Ziel ist eine möglichst hohe *Spezifität* $\mathrm{SPC}_P := \mathrm{Ws}(\mathcal{M}_P(u) = 1 | P(u) = 1)$ (der Prozentsatz an für P legalen Unterschriften u, für die das System mit Hilfe des Modells $\mathcal{M}_P$ die Diagnose „legal“ stellt), sowie eine hohe *Sensitivität* $\mathrm{SNS}_P := \mathrm{Ws}(\mathcal{M}_P(u) = -1 | P(u) = -1)$ (der Anteil an Fälschungen, die als solche vom System erkannt werden), bei gleichzeitig niedriger *False Rejection Rate* $\mathrm{FRR}_P := \mathrm{Ws}(\mathcal{M}_P(u) = -1 | P(u) = 1)$ (wie viele legale Unterschriften irrtümlicherweise als Fälschung eingestuft werden) und geringer *False Acceptance Rate* $\mathrm{FAR}_P := \mathrm{Ws}(\mathcal{M}_P(u) = 1 | P(u) = -1)$ (wie viele Fälschungen als legal akzeptiert werden).

[9] Dieser Abschnitt orientiert sich an der gängigen Literatur über Bayes'sche Entscheidungstheorie, beispielsweise [DH73].

Aus der Definition der *bedingten Wahrscheinlichkeit*

$$\mathrm{Ws}(A|B) := \frac{\mathrm{Ws}(A,B)}{\mathrm{Ws}(B)} \qquad \textit{(bedingte Wahrscheinlichkeit)}$$

für Ereignisse A und B kann man ersehen, daß

$$\mathrm{SPC}_P = \mathrm{Ws}(\mathcal{M}_P(u) = 1 | P(u) = 1) = 1 - \mathrm{Ws}(\mathcal{M}_P(u) = -1 | P(u) = 1) = 1 - \mathrm{FRR}_P$$
$$\mathrm{SNS}_P = \mathrm{Ws}(\mathcal{M}_P(u) = -1 | P(u) = -1) = 1 - \mathrm{Ws}(\mathcal{M}_P(u) = 1 | P(u) = -1) = 1 - \mathrm{FAR}_P$$

Wir kommen also mit *zwei* der vier definierten Qualitätsmaße aus und werden daher in Zukunft nur noch die Raten 'FRR_P' und 'FAR_P' betrachten. Daß nicht sogar *ein einziges* dieser Maße ausreicht leuchtet ein, denn es ist mit obigen Definitionen vereinbar, daß ein Verifikationssystem bei gegebener niedriger FAR entweder auch eine niedrige FRR (ein gutes System) oder eine hohe FRR (ein weniger gutes System) aufweisen kann.

2.6.1.1.1 Messung der Fehlerraten Wir betrachten hier lediglich die *personenspezifischen* Fehlerraten FAR_P und FRR_P. In der Praxis sind eher die *systemweiten* Fehlerraten $\mathrm{FAR} := \langle \mathrm{FAR}_P \rangle_P$ und $\mathrm{FRR} := \langle \mathrm{FRR}_P \rangle_P$ von Interesse, die sich aber durch Mittelwertbildung über die personenspezifischen Fehlerraten einer hinreichend umfangreichen und repräsentativ gewählten Probantenmenge approximieren lassen.

Wir nehmen an, daß uns eine Menge $T_P := T_P^+ \dot\cup T_P^-$ von *Testunterschriften* $u \in \mathbb{U}$ zur Verfügung steht, wobei $\forall u \in T_P^+ : P(u) = 1$ und $\forall u \in T_P^- : P(u) = -1$ gelte. Wir ermitteln die personenspezifischen Fehlerraten approximativ wie folgt:

$$\mathrm{FRR}_P \approx \frac{|\{u \in T_P^+ \mid \mathcal{M}_P(u) = -1\}|}{|T_P^+|}$$
$$\mathrm{FAR}_P \approx \frac{|\{u \in T_P^- \mid \mathcal{M}_P(u) = 1\}|}{|T_P^-|}$$

Die formale Ähnlichkeit dieser beiden Berechnungsformeln darf allerdings nicht über den Umstand hinwegtäuschen, daß die hier genannte Approximation von FAR_P als problematisch anzusehen ist. Während man nämlich einer Menge T_P^+ vernünftigerweise zutrauen darf, daß sie die Unterschrift der Person P in ausreichender Weise statistisch repräsentiert, sofern T_P^+ nur hinreichend viele, und ggf. auch noch über einen längeren Zeitraum erfaßte Unterschriftenproben von P enthält, so ist überhaupt nicht klar, wie eine statistisch repräsentative Menge von *Fälschungen*, d.h. eine geeignete Menge T_P^- für P auszusehen hat. Wir werden auf dieses Problem in 2.6.2 zu sprechen kommen.

2.6.1.1.2 Einstellung der Fehlerraten Den idealen Wert $\mathrm{FRR}_P = 0$ erreicht man trivialerweise, indem man für *sämtliche* Unterschriften $u \in \mathbb{U}$ die Diagnose $\mathcal{M}_P(u) := 1$ stellt; für ein derart naives Modell gilt aber offenbar $\mathrm{FAR}_P = 1$. Die gleiche Überlegung gilt umgekehrt ebenso. In der Praxis wird man es i.d.R. nicht schaffen, für beide Fehlerraten zugleich den Wert 0 zu erreichen, vielmehr wird mit der Erniedrigung der einen Rate zumeist eine Erhöhung der anderen einhergehen. Bei der Konstruktion eines Verifikationssystems wird man daher bemüht sein, einen Parameter $\alpha \in [0,1]$ zu definieren, der es erlaubt, das Verhältnis zwischen FAR_P und FRR_P möglichst stufenlos zu regulieren: Bei $\alpha = 0$ soll $\mathrm{FRR} \approx 0$ gelten, bei $\alpha = 1$ entsprechend $\mathrm{FAR} \approx 0$.[10]

Durch Variation von α läßt sich dann die Systemperformanz an unterschiedliche Anforderungen anzupassen. So wird man z.B. für ein Zugangssystem zu einem Hochsicherheitstrakt nur eine sehr geringe Falschakzeptanzrate zu tolerieren bereit sein, und dafür gelegentliche

[10] Das Zeichen '$\approx$' soll hier dem Umstand Rechnung tragen, daß es für manche Verifikationssysteme evtl. nicht möglich ist, völlige Fehlerlosigkeit zu erreichen.

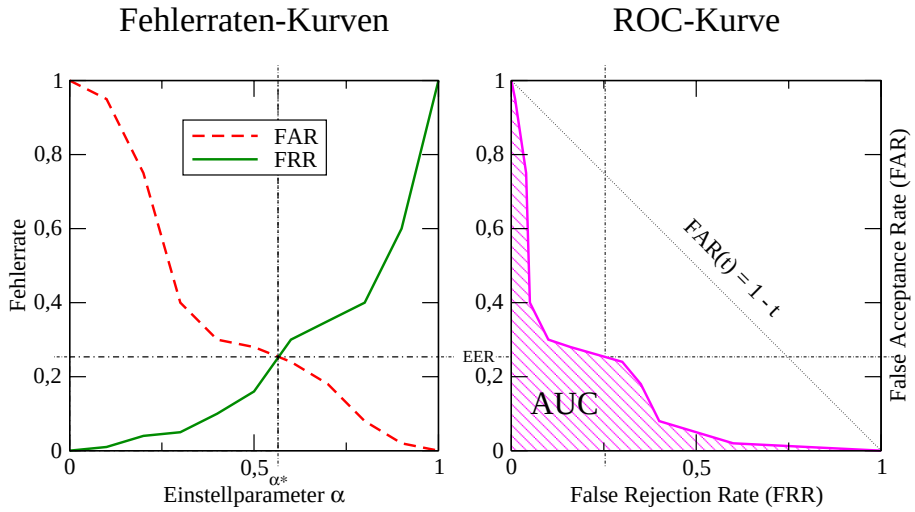

Abbildung 2.19: Fehlerratenkurven und ROC-Kurve

Wiederholungen der Authentifizierungsversuche auch bei zugelassenen Personen in Kauf nehmen. Ein Paketzustelldienst wird hingegen seine Kunden ungern verprellen wollen, und da zudem die Wahrscheinlichkeit, in der Wohnung des auf dem Paket angegebenen Adressaten einen Betrüger anzutreffen, verschwindend gering ist, wird hier eine zu einem niedrigen Wert von FRR_P führende Einstellung von α im Vordergrund stehen.

Technisch gesehen bietet es sich an, den Wert des Einstellparameters α eng an die Größe des *Separierungsthresholds* τ_P des Modells $\mathcal{M}_P$, wie er in 2.5.4.1 eingeführt wurde, zu koppeln, denn dieser Threshold bestimmt, wie ähnlich (ausgedrückt über die Ähnlichkeitsfunktion d) eine vorliegende Unterschrift u zur internen Repräsentation s_P des Modells zu sein hat, damit sie akzeptiert wird.

2.6.1.1.3 Darstellung des Fehlerraten-Verhältnisses Die Werte FAR_P und FRR_P lassen sich in Abhängigkeit vom Einstellparameter α in einem Funktionsdiagramm, wie in Abb. 2.19 (links) gezeigt, darstellen. Die Fehlerratenkurven $\mathrm{FAR}_P(\alpha)$ und $\mathrm{FRR}_P(\alpha)$ werden normalerweise monoton verlaufen, und besitzen dann (mindestens) einen Schnittpunkt an einer Stelle α^*_P, für den $\mathrm{FRR}_P(\alpha^*_P) = \mathrm{FAR}_P(\alpha^*_P) =: \mathrm{EER}_P$ gilt. Den Wert EER_P bezeichnet man als *„Equal-Error-Rate“*. Er wird gelegentlich dazu verwendet, die Leistung eines Verifikationssystems mit nur *einem* Zahlenwert auszudrücken.

Eine weitere Visualisierungsmöglichkeit bietet die sog. ROC-*Kurve* („Receiver Operating Characteristic“). Hier wird das Verhältnis der beiden Fehlerraten zueinander dargestellt, indem die FAR_P als Funktion der FRR_P aufgezeichnet wird. Abb.2.19, rechts, zeigt die ROC-Kurve für die Fehlerratenkurven des linken Schaubilds. Ein ideales Verifikationssystem ist offenbar dadurch gekennzeichnet, daß der Funktionsverlauf konstant bei $\mathrm{FAR}_P \equiv 0$ liegt. Das schlechteste denkbare Verifikationssystem, ein „Zufallssystem“, wird hingegen durch die Diagonale $\varphi(t) = 1 - t$ charakterisiert: Wenn $\mathrm{Ws}[\mathcal{M}_P(u) = -1] = t$ gilt, und die Diagnosen stochastisch unabhängig von der wahren Klassifikation $P(u)$ von u erstellt werden, so ist

$$\mathrm{FRR}_P = \mathrm{Ws}[\mathcal{M}_P(u) = -1 | P(u) = 1] = \mathrm{Ws}[\mathcal{M}_P(u) = -1] = t$$

und

$$\mathrm{FAR}_P = \mathrm{Ws}[\mathcal{M}_P(u) = 1 | P(u) = -1] = \mathrm{Ws}[\mathcal{M}_P(u) = 1] = 1 - \mathrm{Ws}[\mathcal{M}_P(u) = -1] = 1 - t$$

Werden für ein Modell $\mathcal{M}_P$ dennoch Werte oberhalb dieser Diagonalen geliefert, d.h. gibt es Paare $(\mathrm{FRR}_P, \mathrm{FAR}_P)$ mit $\mathrm{FAR}_P > 1 - \mathrm{FRR}_P$, so kann man folgende „Korrektur“ durchführen: Wir fassen $\mathcal{M}_P$ zunächst auf als parametrisierte Folge $\mathcal{M}_P = (\mathcal{M}^\alpha_P)_\alpha$, mit dem Einstellparameter α, und es sei $(\mathrm{FRR}, \mathrm{FAR})_\alpha$ das durch α bestimmte Fehlerratenpaar in der

ROC-Kurve. Wir erzeugen eine neue Modellfolge $\hat{\mathcal{M}}_P$ durch

$$\hat{\mathcal{M}}_P^\alpha(u) := \begin{cases} -\mathcal{M}_P^\alpha(u) & : \text{ falls } \mathrm{FAR}_P(\alpha) > 1 - \mathrm{FRR}_P(\alpha) \\ \mathcal{M}_P^\alpha(u) & : \text{ sonst} \end{cases}$$

Eine solche partielle Invertierung der Modellfolge ist stets in mechanischer Weise durchführbar, ohne daß Informationen über die interne Modellierungsstrategie bekannt sein müßten. Die Leistungsfähigkeit des Systems wird durch diese Operation aber nie verringert. Wir können daher o.B.d.A. annehmen, daß die ROC-Kurve eines Verifikationssystems *niemals oberhalb* der Diagonalen $\varphi(t) = 1 - t$ verläuft.

Möchte man die Leistung eines Systems basierend auf der ROC-Kurve mit nur *einer Zahl* darstellen, so kann man dazu den sog. AUC-*Wert*, die „*Area Under Curve*“, verwenden, d.h. die Fläche unterhalb des Funktionsverlaufes von $\mathrm{FAR}_P(\cdot)$:

$$\mathrm{AUC}_P := \int_0^1 \mathrm{FAR}_P(\alpha)\, d\alpha$$

Der Wert von AUC_P liegt dabei im Bereich $[0, \frac{1}{2}]$; leistungsfähige Systeme haben i.d.R. niedrige AUC-Werte.

Der Einstellparameter α tritt nicht mehr explizit im Diagramm auf, man benötigt ihn aber, um eine ROC-Kurve für eine Person P experimentell zu bestimmen: Dazu gibt man eine aufsteigende Folge $0 =: \alpha_0 < \alpha_1 < \cdots < \alpha_n := 1$ von Werten für α vor, und bestimmt die jeweiligen Paare $(\mathrm{FRR}_P(\alpha_j), \mathrm{FAR}_P(\alpha_j))$, welche man dann im Diagramm aufträgt.

2.6.1.2 Generalisierungsfehler

Die beiden Fehlerraten FAR und FRR sind für sich betrachtet nur von beschränkter Aussagekraft, wenn es um die Frage geht, wie leistungsfähig ein System in der praktischen Anwendung sein wird. Von größerem Interesse hierfür könnte der *Generalisierungsfehler* des Modells sein:

$$\epsilon_P := \mathrm{Ws}[\mathcal{M}_P(u) \neq P(u)] \qquad \textit{(Generalisierungsfehler)}$$

Zwischen dem Generalisierungsfehler ϵ_P und den beiden Fehlerraten FAR_P und FRR_P besteht allerdings ein Zusammenhang:

$$\begin{aligned} \epsilon_P &= \mathrm{Ws}[\mathcal{M}_P(u) \neq P(u)] \\ &= \mathrm{Ws}[\mathcal{M}_P(u) = 1, P(u) = -1] + \mathrm{Ws}[\mathcal{M}_P(u) = -1, P(u) = 1] \\ &= \mathrm{Ws}[\mathcal{M}_P(u) = 1 | P(u) = -1] \cdot \mathrm{Ws}[P(u) = -1] \\ &\quad + \mathrm{Ws}[\mathcal{M}_P(u) = -1 | P(u) = 1] \cdot \mathrm{Ws}[P(u) = 1] \\ &= \mathrm{FAR}_P \cdot \mathrm{Ws}[P(u) = -1] + \mathrm{FRR}_P \cdot (1 - \mathrm{Ws}[P(u) = -1]) \end{aligned} \tag{2.22}$$

Dabei läßt sich $\mathrm{FRD}_P := \mathrm{Ws}[P(u) = -1]$ aus (2.22) interpretieren als die Wahrscheinlichkeit, mit der ein *Betrugsversuch* (engl.: „fraud“) stattfindet, denn die der Wahrscheinlichkeitsbestimmung zugrundeliegende (hier nie explizit genannte) Verteilung D gibt an, mit welcher Wahrscheinlichkeit eine Unterschrift u dem Verifikationssystem zur Prüfung vorgelegt wird. Damit ist allerdings der Wert von FRD_P – und damit implizit auch der Wert von ϵ_P – inhärent abhängig vom jeweiligen Szenario: Im Beispiel des Paketdienstes aus 2.6.1.1.2 war mit einer sehr geringen Betrugswahrscheinlichkeit zu rechnen, d.h. es gilt $\mathrm{FRD}_P \approx 0$, womit ϵ_P gemäß (2.22) gegeben ist durch

$$\epsilon_P = \mathrm{FRD}_P \cdot \mathrm{FAR}_P + (1 - \mathrm{FRD}_P) \cdot \mathrm{FRR}_P \approx 0 \cdot \mathrm{FAR}_P + 1 \cdot \mathrm{FRR}_P \approx \mathrm{FRR}_P$$

In diesem Fall ist die Falschakzeptanzrate nahezu irrelevant, und wir können den Parameter α in Hinblick auf einen geringen Wert von FRR_P wählen – ein Ergebnis, zu dem wir bereits zuvor durch informelle Überlegungen gekommen waren.

2.6.1.3 Konfidenz

Befindet sich ein Verifikationssystem im Einsatz, so kennt man nur die Diagnosen $\mathcal{M}_P(u)$ zu einer Unterschrift u. Wir werden daher zwei weitere Bewertungsmaße für das Modell $\mathcal{M}_P$ betrachten, welche wir *„positive Konfidenz“*

$$\mathrm{CNF}^+{}_P := \mathrm{Ws}[P(u) = 1 |\, \mathcal{M}_P(u) = 1] \qquad \textit{(„positive Konfidenz“)}$$

und *„negative Konfidenz“*

$$\mathrm{CNF}^-{}_P := \mathrm{Ws}[P(u) = -1 |\, \mathcal{M}_P(u) = -1] \qquad \textit{(„negative Konfidenz“)}$$

nennen wollen. Intuitiv kann man diese beiden stochastischen Größen als Antwort auf die Frage ansehen, wie *vertrauenswürdig* eine Diagnose $\mathcal{M}_P(u)$ des Systems ist. Über den *Satz von Bayes* [Fel68]

$$\mathrm{Ws}[A_i|B] = \frac{\mathrm{Ws}[B|A_i] \cdot \mathrm{Ws}[A_i]}{\sum_{1\le\lambda\le n} \mathrm{Ws}[B|A_\lambda] \cdot \mathrm{Ws}[A_\lambda]} \qquad \textit{(Satz von Bayes)}$$

für Ereignisse A_i, $1 \le i \le n$, und B ergibt sich für $\mathrm{CNF}^+{}_P$:

$$\begin{aligned}
\mathrm{CNF}^+{}_P &= \mathrm{Ws}[P(u) = 1 |\, \mathcal{M}_P(u) = 1] \\
&= \frac{\mathrm{Ws}[\mathcal{M}_P(u) = 1 | P(u) = 1] \cdot \mathrm{Ws}[P(u) = 1]}{\mathrm{Ws}[\mathcal{M}_P(u) = 1 | P(u) = 1] \cdot \mathrm{Ws}[P(u) = 1] + \mathrm{Ws}[\mathcal{M}_P(u) = 1 | P(u) = -1] \cdot \mathrm{Ws}[P(u) = -1]} \\
&= 1/\left[1 + \frac{\mathrm{Ws}[\mathcal{M}_P(u) = 1 | P(u) = -1]}{\mathrm{Ws}[\mathcal{M}_P(u) = 1 | P(u) = 1]} \cdot \frac{\mathrm{Ws}[P(u) = -1]}{\mathrm{Ws}[P(u) = 1]}\right] \\
&= 1/\left[1 + \frac{\mathrm{FAR}_P}{1 - \mathrm{FRR}_P} \cdot \frac{\mathrm{FRD}_P}{1 - \mathrm{FRD}_P}\right]
\end{aligned}$$

Mit den Bezeichnungen $\eta_P := \frac{\mathrm{FAR}_P}{\mathrm{FRR}_P}$ und $\phi_P := \frac{\mathrm{FRD}_P}{1-\mathrm{FRD}_P}$ können wir damit schreiben:

$$\mathrm{CNF}^+{}_P = \left[1 + \frac{\phi_P}{\mathrm{FAR}_P^{-1} - \eta_P^{-1}}\right]^{-1} \tag{2.24a}$$

Analog erhält man:

$$\mathrm{CNF}^-{}_P = \left[1 + \frac{\phi_P^{-1}}{\mathrm{FRR}_P^{-1} - \eta_P}\right]^{-1} \tag{2.24b}$$

Die beiden Konfidenzbegriffe lassen sich also rechnerisch auf die *systemspezifischen* Fehlerraten FAR_P und FRR_P sowie die *situationsspezifische* Betrugswahrscheinlichkeit FRD_P zurückführen.

Wir betrachten als Beispiel ein Unterschriftenverifikationssystem mit einer durchschnittlichen Equal Error Rate $\mathrm{EER} = \langle \mathrm{EER}_P \rangle_P$ von 1% ($\mathrm{FAR} = \mathrm{FRR} = 10^{-2}$, $\eta = 1$), was nach heutigen Maßstäben als sehr gut angesehen werden müßte. Das System soll in einer Umgebung eingesetzt werden, in dem unter 1000 Unterschriften nur eine Fälschung zu erwarten ist ($\mathrm{FRD} = 10^{-3}$, $\phi = 1/(10^3 - 1)$) – sofern Unterschriftenfälschungen im Finanzwesen nicht deutlich häufiger auftreten als Kreditkartenbetrügereien [BLH99], so ist dies ein durchaus realistisches Szenario. Es ergibt sich mit (2.24a) ein positiver Konfidenzwert CNF^+ von nahezu 1, d.h. man darf der Diagnose, daß es sich bei der vorgelegten Unterschrift u um ein Original handelt, entsprechend großes Vertrauen schenken. Ganz anders sieht es jedoch für die negative Konfidenz aus: Hier liefert (2.24b) einen Wert für CNF^- von lediglich etwa 9%, d.h. auch bei einer Diagnose „$\mathcal{M}(u) = -1$“ erscheint es vernünftiger zu glauben, daß die Unterschrift in Wahrheit legal ist. Dies ist ein typisches Problem bei Diagnosesystemen, welches immer dann auftritt, wenn die Fehlerraten FAR und FRR deutlich größer sind als die Fehlerauftrittswahrscheinlichkeit FRD.

2.6.2 Diskussion zur Auswahl der Testdaten

Unser Ziel ist die Bestimmung einer möglichst repräsentativen Menge von *Testunterschriften* für die statistische Leistungsmessung gemäß 2.6.1. Ein wichtiger Grundsatz lautet, daß die Menge der Testdaten sich nicht mit der Menge der Trainingsdaten überschneiden darf, denn ein zu testendes Modell hat sich zwangsläufig an die Trainingsdaten angepaßt, womit das Ergebnis verfälscht würde (vgl. die Diskussion in 2.5.4.4 zum Thema „Overfitting"). Übrigens dürfen mit der gleichen Begründung auch keine *Validierungsdaten* zum Testen eingesetzt werden, denn diese Teilmenge der Trainingsdaten diente der Vermeidung von Overfitting und damit der Modellgenerierung.

Wir gehen an dieser Stelle davon aus, daß es uns gelungen ist, von einer hinreichend großen Anzahl an Personen jeweils eine ausreichend große Menge an Originalunterschriften (positive Beispiele) zu erhalten. Damit bleibt als wesentliche Frage, von welcher Art die *negativen Beispiele* sein sollten. Wir diskutieren dazu wieder die in 2.1.2 vorgestellten Fälschungstypen, wobei wir uns hier auf die beiden für die Praxis am meisten relevanten Typen, nämlich *Fremdunterschriften* und *Schriftbildkopien*, beschränken werden (vgl. hierzu auch die Diskussionen in 2.1.2 und 2.2).

2.6.2.1 Testen mit Fremdunterschriften (Fälschungstyp 1)

Obwohl Fremdunterschriften eigentlich den naivsten Fälschungstyp darstellen, werden sie sehr häufig zum Testen eingesetzt. Das hat zum einen praktische Gründe, denn wenn man von r Probanten je m Unterschriftenproben besitzt, so stehen zum Testen für jedes Modell $(r-1)\cdot m$ Fälschungen zur Verfügung. Darüberhinaus scheint die Objektivität und Repräsentativität durch Fremdunterschriften noch am ehesten gewährleistet zu sein, denn die Qualität der Fälschung hängt hier nicht von einer aufwendigen Lernprozedur und dem Talent des Fälschers ab. Es ist allerdings zu erwarten, daß das Meßergebnis bei Verwendung von Fremdunterschriften i.a. zu gut ausfällt, also lediglich eine obere Schranke für die tatsächliche Leistung des Systems darstellt.

Als Alternative bietet sich noch das Testen mit *modifizierten Fremdunterschriften* an. Im einfachsten Fall wird man dazu die Unterschrift mit einer gewissen „Rauschsequenz" additiv überlagern, oder neue Unterschriften durch Kombination von Einzelunterschriften bilden. Hinsichtlich der Frage nach Repräsentativität erscheinen solche Ansätze allerdings problematisch. Außerdem könnten sie bei solchen Systemen ins Leere greifen, welche intern während der Datenvorverarbeitung das Rauschen beseitigen (vgl. 2.5.2.1), oder bei der Modellierung ohnehin mehrere Trainingsunterschriften zusammenfassen, wie wir es im Modellierungsbeispiel aus 2.5.4.2 getan haben.

2.6.2.2 Testen mit Schriftbild-Kopien (Fälschungstyp 3)

Die meisten realen Unterschriftenfälschungen dürften Schriftbildkopien sein, sodaß dieser Fälschungstyp bei der Evaluation eines Systems besonders große Aufmerksamkeit verdient. Für durchschnittlich gute Kopien bedarf es vermutlich weder allzu großer Übung noch besonderen Talents, und es ist anzunehmen, daß die meisten in der Realität auftretenden kopierenden Fälschungen von *Laien* auf diesem Gebiet erstellt wurden, sodaß der Aufbau eines statistisch repräsentativen Bestands an qualifizierten Fälschungen dieser Art nicht unmöglich erscheint. Als Schwierigkeit ergibt sich, daß dazu im Prinzip jeder Proband die Unterschrift jedes anderen mindestens einmal fälschen sollte, was nicht nur ein organisatorisches Problem darstellt, sondern auch noch die Bereitschaft aller Beteiligten erfordert, seine eigene Unterschrift für Fälschungsversuche durch zumeist fremde Personen zur Verfügung zu stellen.

Kapitel 3

Eigenschaften der verwendeten Daten

Die in dieser Arbeit verwendeten Daten wurden während des Sommers 2003 durch mehrere am BiSP-Projekt beteiligte Arbeitsgruppen erfaßt. Jeder Probant hatte (1) seine Unterschrift abzugeben, (2) den Namen „*Dog Müller*“ in seiner eigenen Schrift ohne Vergleichsvorlage zu schreiben (Blindfälschung, Typ 2) und (3) das Schriftbild einer vorgegebenen Unterschrift „*Dog Müller*“ (s. Abb. 2.3, S. 9)[1] nachzuzeichnen (kopierende Fälschung, Typ 3). Jede dieser Aufgaben wurde während einer Sitzung zehnmal hintereinander durchgeführt, woraus sich drei Dateien mit jeweils 10 Datensätzen ergaben. Diese Dateien wurden später in je zehn Teildateien zerlegt, die jeweils eine einzelne Schriftprobe repräsentieren, und in dieser Form den Teilnehmern des BiSP-Projektes zur Verfügung gestellt.

Erfaßt wurden Schreibproben von insgesamt 70 Probanten, von diesen nahmen 40 Personen an 10 Sitzungen, 10 Personen an mindestens 4 Sitzungen, 1 Person an 2 Sitzungen und 19 Personen an genau einer Sitzung teil. Für die Zwecke des Autors sind lediglich die *Unterschriften* von Interesse. Von 4480 ursprünglich erfaßten Exemplaren verbleibt nach der Entfernung einiger offensichtlich defekter Datensätze eine Gesamtheit von 4464 Unterschriften, welche dieser Arbeit als Datenmaterial zugrunde liegen. Die Daten wurden vom Autor vor ihrer Verwendung in den Experimenten anonymisiert.

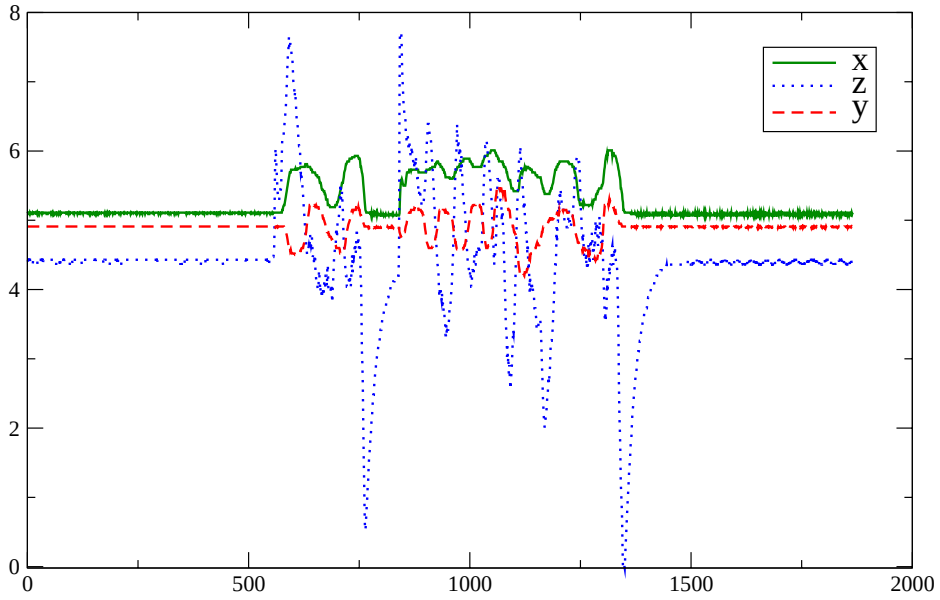

Abbildung 3.1: Unterschriftensequenz des BiSP-Pens direkt nach Erfassung

[1] Hinter der Unterschrift „*Dog Müller*“ steht keine reale Person; sie wurde von einem hier nicht zu nennenden Teilnehmer des BiSP-Projektes ausschließlich für die genannte Feldstudie angefertigt.

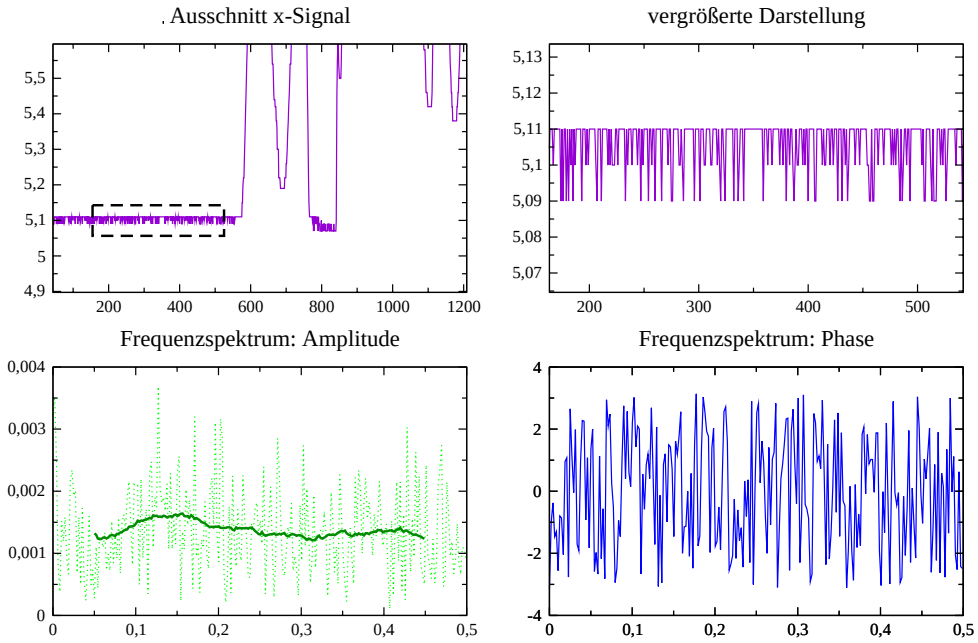

Abbildung 3.2: Verrauschtes Komponentensignal

3.1 Eigenschaften in der Zeitdomäne

Abb. 3.1 zeigt eine mit dem BiSP-Pen aufgezeichnete nicht bearbeitete Unterschriftensequenz. Das eigentliche Signal startet nach etwa 500 Samples und besitzt eine Laufzeit von rund 1000 Samples (2000 ms). Zu Beginn und am Ende ist das Signal weitgehend konstant („DC-Anteile“), das Erfassungssystem befindet sich dort im „Leerlauf“.

3.1.1 Basislevel und Dynamikbereich

Die DC-Anteile der drei Signalkomponenten in Abb. 3.1 sind von 0 verschieden und stimmen, bedingt u.a. durch ungenaue Kalibrierung und durch einen langfristigen Drift[2], auch nicht exakt überein. Der eigentlich vorgesehene Basiswert der drei Signale betrug 5V, die Signale konnten Werte zwischen 0V und 10V annehmen (vgl. 2.4.1); dieser Wertebereich wird in unserem Beispiel nur von der z-Komponente weitgehend vollständig ausgenutzt.

3.1.2 Störung durch Rauschüberlagerung

Auffällig ist ein gewisses „Vibrieren“ des Signalverlaufs, das vor allem im Leerlauf gut zu sehen ist, aber auch während der Aufzeichnung auftritt. Diese in Abb. 3.2 (oben) für das x-Signal in zwei Vergrößerungsstufen dargestellte Störung trat bei allen an der Erfassung beteiligten Arbeitsgruppen häufig auf, sodaß zu vermuten ist, daß es sich um eine Eigenschaft des BiSP-Pens handelt. Im vorliegenden Beispiel wird der DC-Anteil um ein bis zwei hundertstel Volt variiert, also um *1 bis 2 Quantisierungsstufen*. In Einzelfällen traten stärkere Störungen bis hin zu *8 Quantisierungsstufen* auf. Die dargestellten Schaubilder legen die Vermutung nahe, daß es sich bei der Störung um *weißes Rauschen* gemäß 2.5.2.1.2.2 handelt, vielleicht hervorgerufen durch elektronisches Rauschen im A/D-Wandler des BiSP-Pens: Die Sprünge scheinen weitgehend zufällig aufzutreten (Bilder oben), die Amplitude des Frequenzspektrums (links unten) scheint über weite Bereiche um einen konstanten Wert herum zu variieren, und die Phase (rechts unten) scheint zufällige Werte im Bereich $[-\pi, +\pi]$ anzunehmen. Ein statistischer Test für diese Vermutungen wurde nicht durchgeführt.

[2]Während einer einzelnen Erfassung trat dieser Drift nur gelegentlich und dann in geringem Maße auf.

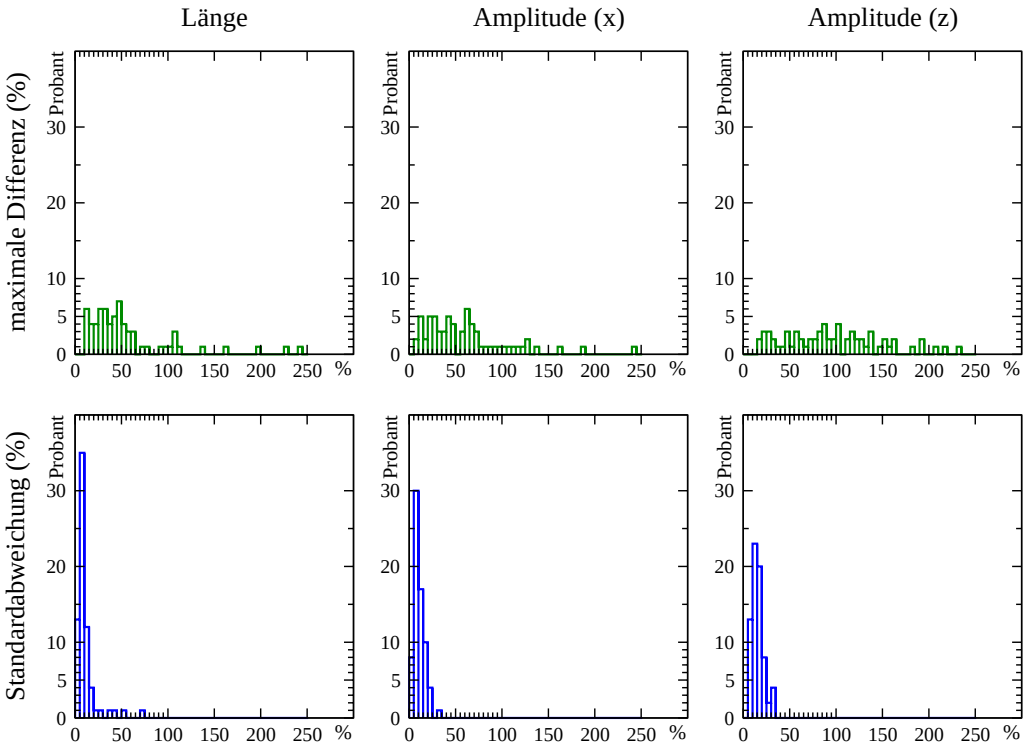

Abbildung 3.3: Personenspezifische Variabilitäten bei Länge und Amplitude

3.1.3 Längen- und Amplituden-Unterschiede

Abb. 2.2 in 2.3.2.2 auf S. 8 hatte bereits deutlich gemacht, daß sich die Unterschriften zweier Personen hinsichtlich Länge (Schreibdauer) und Amplitude (Druck) deutlich voneinander unterscheiden können. Die Unterschiede zwischen den Unterschriften der *gleichen* Person stellen sich als personenabhängig heraus.

Abb. 3.3 stellt im Histogramm links oben die Verteilung der *prozentualen maximalen Längendifferenzen* für alle Probanten dar. Für jede Person P wurde dessen längste Unterschrift $u^{P,\max}$ und kürzeste Unterschrift $u^{P,\min}$ ermittelt und deren *relativer* Längenunterschied nach der Formel $100 \cdot (|u^{P,\max}| - |u^{P,\min}|)/|u^{P,\min}|$ bestimmt ('$|u|$' steht für die Länge einer Unterschrift u).[3] Die Abszisse ist in Abschnitte zu je 5% unterteilt, die Ordinate gibt die Anzahl an Probanten an. Man erkennt an dem Schaubild, daß die Schreibdauer für zwei Unterschriften sich bei einem Großteil der Personen um maximal 50% unterscheidet, jedoch gibt es auch Probanten mit einem Längenunterschied von über 200%. Es stellt sich heraus, daß sehr lange Unterschriften oft aus den ersten Schreibversuchen mit dem BiSP-Pen stammen: Unter den 51 Probanten, welche an mehr als einer Sitzung teilgenommen haben, stammte in 18 Fällen ($\approx 35\%$) die längste Unterschrift aus der ersten Erfassungsserie. Eine visuelle Untersuchung der längsten Unterschriften ergab ferner, daß diese gelegentlich durch sehr langes Absetzen des Stiftes zustande gekommen waren, vermutlich zwischen dem Schreiben zweier Namensteile.

Im Histogramm links unten wird die *relative Standardabweichung* der Längendifferenzen nach der Formel $100 \cdot \sqrt{\langle(|u^{P,(i)}| - \mu^P)^2\rangle_i}/\mu^P$ angegeben, wobei $\mu^P := \langle u^{P,(i)}\rangle_i$ die mittlere Länge aller Unterschriften von P ist. Die Längenabweichungen der Unterschriften $u^{P,(i)}$ des Probanten P werden also bezogen auf deren *mittlerer* Länge μ^P dargestellt, denn langen Unterschriften wird man fairerweise eine absolutgesehen größere Variabilität zugestehen als kurzen Unterschriften. Gemäß dieser Sichtweise streuen die Längenunterschiede der meisten Probanten maximal um 25%, wobei das Maximum der Verteilung bei etwa 10% liegt.

[3] Eine Methode zur Feststellung der Unterschriftenlänge wird in 4.1.4 vorgestellt.

Die restlichen vier Histogramme stellen in ähnlicher Weise die maximalen und durchschnittlichen Variabilitäten einer Person hinsichtlich der *Amplitude* dar, wobei sich hier auf die Darstellung der x- und der z-Komponente beschränkt wurde. Die Berechnung der statistischen Werte erfolgt analog zu der für Längendifferenzen, allerdings wurden die Signale zuvor zentriert und auf *Amplituden-Standardabweichung* hin normiert (vgl. 2.5.2.2). Es werden hier also nicht die Unterschiede in den absoluten Spannungswerten angegeben, sondern die Unterschiede in der *Dynamik* des Signals.

Bei den maximalen Amplituden-Unterschieden zeigt sich eine noch stärkere Streuung als bei den Längenunterschieden; speziell bei der z-Komponente ergibt sich eine Verteilung, die in weiten Bereichen der Gleichverteilung ähnelt. Man beachte allerdings, daß die Amplitudenwerte hier durch die Zeitstelle maximalen Drucks, d.h. durch einen einzigen Druckwert bestimmt werden. Gelegentlich lassen sich bei Unterschriften einzelne Stellen extremen Drucks beobachten, welche für den dynamischen Verlauf der Gesamtunterschrift nicht unbedingt repräsentativ erscheinen. Die Histogramme für die Standardabweichungen relativieren dieses Bild denn auch wieder, denn es zeigt sich, daß selten mit einer durchschnittlichen Abweichung von mehr als 25% zu rechnen ist.

3.1.4 Stift-Rotation

Für die Erfassung der in dieser Arbeit verwendeten Daten war die Position eines der vier seitlichen Drucksensoren des BiSP-Pens (vgl. 2.4.1) an der Außenwand des Stiftes markiert worden und sollte beim Schreiben in Schreibrichtung nach rechts zeigen. Damit sollte vermieden werden, daß der Stift beim Schreiben eine beliebige Anfangsrotation aufweist. Eine neuere Version des BiSP-Pens ist nicht mehr kreisrund, sondern dergestalt asymmetrisch geformt, daß ein stets gleichartiges Halten des Schreibgerätes zumindest für die gleiche Person in gewissen Grenzen erzwungen wird. Im Idealfall erspart man sich auf diese Weise einen Vorverarbeitungsschritt zum *Rotationsausgleich*. Wir wollen uns dennoch ansehen, in welcher Weise sich die Rotation des Stiftes um die Längsachse auf die erfaßten Daten auswirkt, da eine spezifische Ausrichtung des Stiftes im Normalfall nicht vorausgesetzt werden kann. Im Übrigen scheinen sich auch bei der Erfassung der in dieser Arbeit betrachteten Daten nicht immer alle Probanten an die Vorgaben gehalten zu haben.

Für die folgende theoretische Erörterung fassen wir die x_t- und y_t-Samples einer Unterschrift zu Paaren (x_t, y_t) zusammen. Seien $((x_t, y_t))_t$ und $((x'_t, y'_t))_t$ zwei solche Paarsequenzen, wobei beide Sequenzen die gleiche Unterschrift repräsentieren mit dem Unterschied, daß $((x'_t, y'_t))_t$ im Vergleich zu $((x_t, y_t))_t$ durch Rotation des Smartpens um einen Winkel ω^* erzeugt wurde. Beträgt beispielsweise $\omega^* = 90°$, so gilt $x'_t = y_t$ und $y'_t = -x_t$. In allgemeiner Form läßt sich das Verhältnis zwischen den Paaren (x_t, y_t) und (x'_t, y'_t) über *Polarkoordinaten* beschreiben:[4]

1. Das Paar (x_t, y_t) wird in Polarkoordinaten (r_t, ω_t) umgerechnet:[5]

$$r_t := \sqrt{x_t^2 + y_t^2}$$
$$\omega_t := \arctan(y_t / x_t)$$

2. Eine Rotation um den Winkel ω^* wird durchgeführt:

$$\tilde{r}_t := r_t$$
$$\tilde{\omega}_t := \omega_t + \omega_t^*$$

3. Es findet eine Rückführung in cartesische Koordinaten statt:

$$\tilde{x}_t := \tilde{r}_t \cdot \cos(\tilde{\omega}_t)$$
$$\tilde{y}_t := \tilde{r}_t \cdot \sin(\tilde{\omega}_t)$$

[4] Diesen Hinweis verdankt der Autor seinem Betreuer PD Dr. R. Brause.
[5] Beachte die Bemerkung zur verwendeten arctan-Funktion auf S. 15.

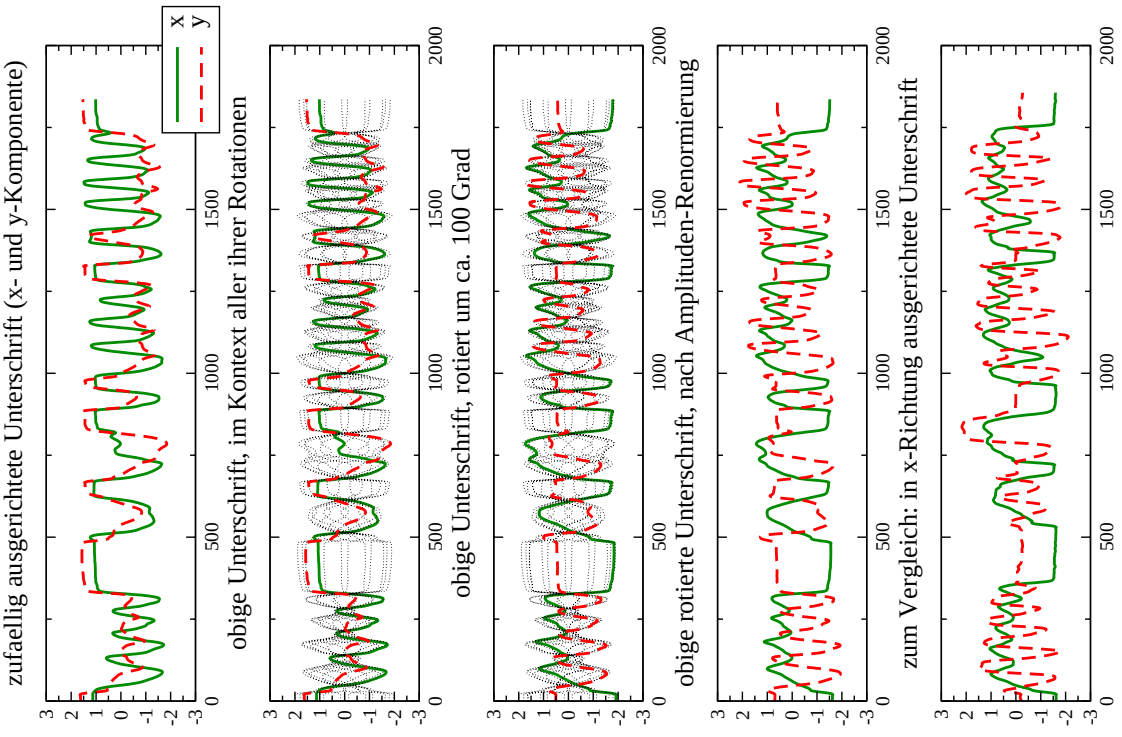

Abbildung 3.4: Gegeneinander rotierte Unterschriften einer Person

Es gilt nun offenbar

$$(\tilde{x}_t, \tilde{y}_t) = (x'_t, y'_t) \tag{3.1}$$

Abb. 3.4 stellt die soeben vorgestellte Methode zum *manuellen Rotationsangleich* dar am Beispiel zweier Unterschriften der gleichen Person, die mit einem ungefähr um eine viertel Drehung zueinander rotierten Stift erfaßt wurden (oberstes und unterstes Schaubild).[6] Das zweite Bild von oben stellt die erste Unterschrift gemeinsam mit mehreren durch Anwendung der Polarkoordinatenmethode berechneten Drehungen ihrer selbst dar (feingepunktete Linien). Das mittlere Bild gibt den gleichen „Rotationskontext“ wieder, wobei hier die Drehung um ca. 100° hervorgehoben wird; das vierte Bild zeigt dieses neue Signal ohne Rotationskontext. Zwischen der gedrehten ersten und der unrotierten zweiten Unterschrift im letzten Bild bestehen augenscheinlich Ähnlichkeiten.

3.1.5 Stift-Neigung

Bei der Betrachtung der x-Komponente der in Abb. 3.5 dargestellten Unterschrift fällt auf, daß diese über weite Strecken *negative* Werte (relativ zum Leerlaufsignal; die DC-Anteile aller Komponentensignale wurden zur besseren Veranschaulichung angeglichen) annimmt. Dieses Phänomen kann nicht allein durch linksgerichtete Schreibbewegungen erklärt werden, wie sie z.B. beim Schreiben von Schleifen („e“, „l“, etc.) auftreten. Da die Hauptschreibrichtung zumindest bei deutschsprachigen Schreibern von links nach rechts verläuft, müßte der Anteil an positiven x-Werten hoch sein. Der Linksdruck entsteht vielmehr durch eine *Links-Neigung* des Stiftes. Dies konnte experimentell bestätigt werden, indem der BiSP-Pen ohne Schreibbewegung in vertikaler Richtung auf die Schreibfläche gepreßt und dann in negativer x-Richtung abgekippt wurde. Tatsächlich wurde das in Abb. 3.5 dargestellte Signal von einem dem Autor namentlich bekannten Linkshänder erstellt, welcher den Stift die meiste Zeit über deutlich nach links geneigt hält.

[6] Die in diesem Beispiel betrachteten Unterschriften stammen nicht aus dem zu Beginn des Kapitels vorgestellten Datenbestand.

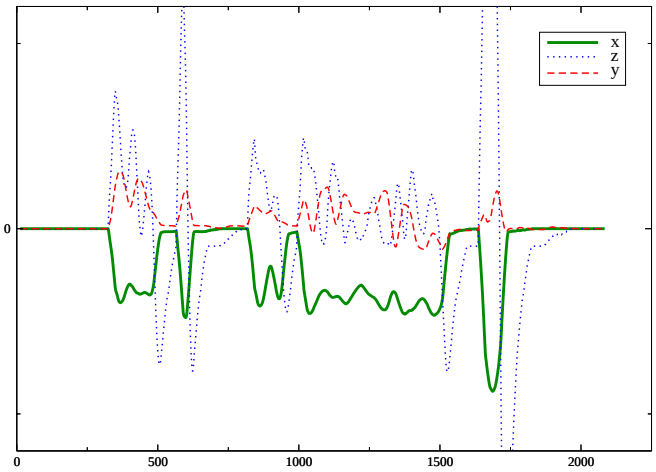

Abbildung 3.5: Negative x-Werte durch Linksneigung

3.2 Eigenschaften in der Frequenzdomäne

3.2.1 Frequenzgänge unterschiedlicher Signalkomponenten

Abb. 3.6 stellt die drei Komponentensignale der Zeitsequenz aus Abb. 3.1 in der Frequenzdomäne dar. Vor der Fouriertransformation gemäß 2.5.2.1.1 wurde das Zeitsignal mit dem *Hamming-Window* (2.4), S. 15, moduliert, um evtl. vorhandene feine Strukturen im Frequenzgang besser herauszuarbeiten. Der Amplitudenwert bei der Frequenzstelle 0 wurde nach der DFT jeweils auf 0 gesetzt, da der ursprüngliche Wert den DC-Anteil des Signals angibt, welcher von der Kalibrierung des Erfassungssystems abhängt und daher für unsere Zwecke keine relevante Information enthält. Die Amplitude des Frequenzgangs (gepunktete Kurve) ist stark verrauscht, daher wurde eine zweite Kurve (durchgezogen) angegeben, welche durch Mittelung der Frequenzgänge nach (2.5), S. 17, von 50 Unterschriften der gleichen Person, für die zuvor ein Längenangleich durch *Resampling* im Sinne von 2.5.2.3.1 stattgefunden hatte, erzeugt wurde.

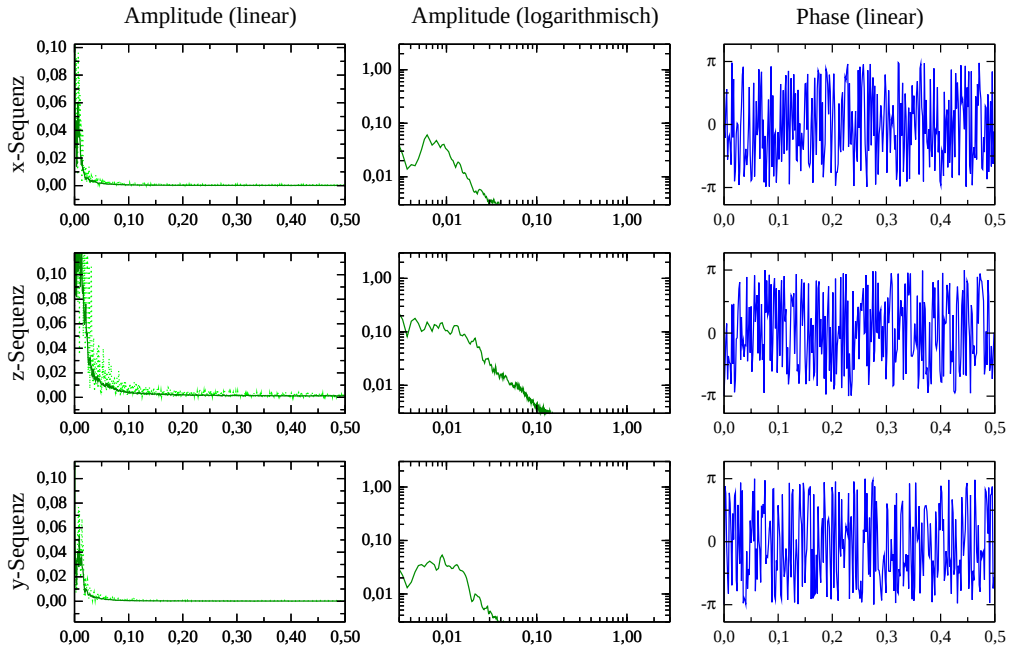

Abbildung 3.6: Frequenzspektren der Komponenten einer Unterschriftensequenz

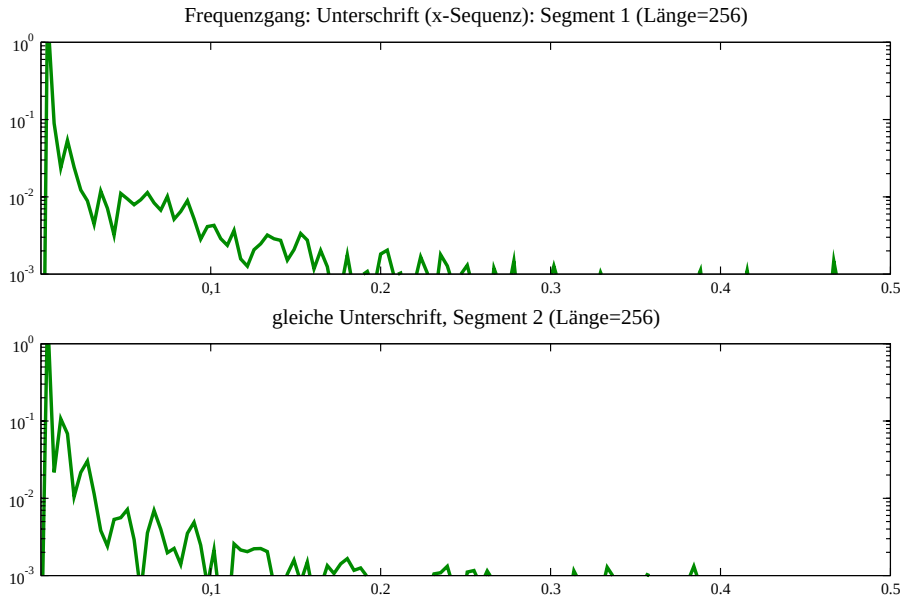

Abbildung 3.7: Untersuchung: Zeitabhängigkeit des Frequenzspektrums

Die *Phasen* der Signalkomponenten (rechte Bilder) lassen nach Ansicht des Autors keine spezifische Struktur erkennen; das Aussehen ähnelt für alle drei Signalkomponenten demjenigen der Phase des weißen Rauschens in Abb. 2.9 auf S. 18. Wir werden uns daher nicht weiter mit der Phase des Frequenzgangs auseinandersetzen.

Die *Amplitude* aller drei Komponenten beginnt im niederfrequenten Bereich mit hohen Werten und fällt dann rasch ab (linke Bilder). Für alle Komponenten sieht man, daß sich bei einem Zehntel der Samplingfrequenz f_S (≈ 50 Hz) das Signal nicht mehr vom Grundrauschen unterscheiden läßt. Der Verlauf der Amplitude im Frequenzbereich unterhalb $0.1 f_S$ läßt sich anhand der doppeltlogarithmischen Darstellungen (mitte) beurteilen. Demgemäß erscheint der Verlauf für die x- und die y-Komponente sehr ähnlich, die z-Komponente weicht hierin ein wenig von den beiden anderen ab, wobei aber das Grundmuster erhalten bleibt. Der Anstieg der Amplitude zu Beginn des Verlaufs der x- und y-Komponente befindet sich bei unter $0.01 f_S$ und läßt sich somit vermutlich als Artefakt des Entfernens der Null-Frequenz (s.o.) erklären; daß dieses Phänomen bei der z-Komponente nicht allzu deutlich hervortritt, könnte auf ein Darstellungsproblem zurückzuführen sein. Keine der drei Amplitudenverläufe scheint über besonders markante strukturelle Eigenschaften, z.B. in Form deutlich hervortretender Peaks bei bestimmten Frequenzen, zu verfügen.

3.2.2 Zeitabhängigkeit des Frequenzgangs

Für Abb. 3.7 wurden aus dem Zeitsignal einer Unterschrift zwei sich nicht überlappende Abschnitte der Länge 256 entnommen und in die Frequenzdomäne transformiert. Ziel ist es, durch Vergleichung dieser beiden Zeitabschnitte festzustellen, ob der Frequenzgang einer Unterschrift während des Schreibens variabel ist. Aus der vorangegangenen Beobachtung, wonach keine relevanten Frequenzanteile oberhalb von $0.1 f_S$ auftreten, ergibt sich, daß die kürzesten in Unterschriften auftretenden informationstragenden Strukturen eine Dauer von minimal 0.02 s haben können. Damit erscheinen die hier verwendeten Abschnitte, welche eine Länge von ca. 0.5 s repräsentieren, im Prinzip sehr lang für eine verläßliche Analyse. Eine weitere Verkürzung der Abschnitte würde eine solche Analyse jedoch praktisch unmöglich machen: Bereits bei der hier gewählten Abschnittslänge von 256 Zeit-Samples entfallen auf den interessierenden Bereich zwischen 0 und $0.1 f_S$ in der Frequenzdomäne lediglich rd. 25 Frequenz-Samples.

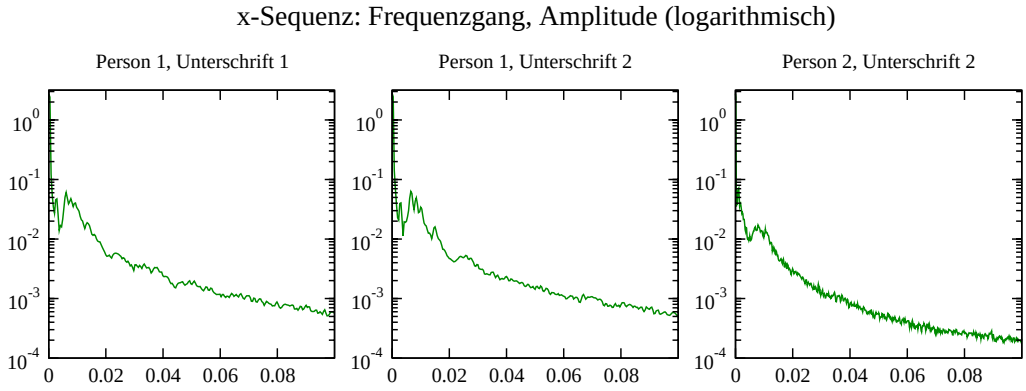

Abbildung 3.8: Frequenzspektrum verschiedener Unterschriften

In beiden Schaubildern fällt der Amplitudenwert zügig, und erreicht bei etwa $0.1 f_S$ einen Wert von unter 1/100-tel des Ausgangswertes bei ca. 1 Hz. Der grundsätzliche Verlauf entspricht dem der drei Komponentensequenzen in Abb. 3.6 aus 3.2.1. Die teilweise starken Schwankungen der Amplitude im niederfrequenten Bereich repräsentieren nicht notwendigerweise tatsächlich existierende Phänomene: Im Gegensatz zu Abb. 3.6 wurde keine Mittelung mehrerer Frequenzverläufe zu deren Glättung gemäß (2.5) auf S. 17 durchgeführt, denn hierzu hätten zueinander passende Zeitverlaufs-Abschnitte gleicher Länge aus verschiedenen Unterschriften der gleichen Person ermittelt werden müssen, was in der Praxis nur schwer durchführbar erscheint. Eine Glättung des Frequenzgangs mittels Moving-Average-Filter im Sinne von 2.5.2.1.5 ist ebenfalls praktisch ausgeschlossen aufgrund der geringen Anzahl an Frequenz-Samples (s.o.). Damit sind charakteristische Unterschiede zwischen den beiden Amplitudenverläufen zumindest nicht ohne Weiteres auszumachen, eher scheint das gezeigte Beispiel darauf hinzudeuten, daß der Frequenzgang nicht in bedeutsamer Weise von der Zeit abhängig ist.[7]

3.2.3 Frequenzgänge bei verschiedenen Unterschriften

Die wesentliche Form des Frequenzspektrums bleibt auch erhalten, wenn man zwei *verschiedene* Unterschriften der *gleichen* Person vergleicht (Abb. 3.8, links und mitte). Es ergibt sich somit aus dem bisher Gesehenen, daß der Frequenzgang für verschiedene Unterschriften einer Person einigermaßen stabil ist, und damit auf den ersten Blick „charakteristisch“ für diese Person zu sein scheint. Betrachten wir nun allerdings in Abb. 3.8, mitte und rechts, die Amplitudenverläufe zweier Unterschriften *verschiedener* Personen, so sehen wir erneut einen i.w. gleichen Grundverlauf. Zwar erkennt man im niederfrequenten Bereich gewisse Unterschiede zumindest in den absoluten Amplitudenwerten, jedoch ist es fraglich, ob anhand dieser Differenzen eine sichere Unterscheidung der Unterschriften zweier Personen möglich ist.

Der Autor vermutet nach dieser Erörterung, daß ein Vergleich zweier Unterschriften anhand der jeweiligen DFT-Koeffizienten zu keinen besonders guten Ergebnissen führen dürfte. Es erscheint vielmehr einleuchtend, daß die inhaltliche Information einer Unterschrift in erster Linie im *Zeitverlauf* codiert ist, und vom Frequenzspektrum lediglich eine für Unterschriften allgemeine „formale“ Information zu erwarten ist. Aber selbst wenn es im Frequenzspektrum bisher nicht entdeckte hinreichend charakteristische Merkmale geben sollte, so können diese sich wohl nur im niederfrequenten Bereich unterhalb $0.1 f_S$ befinden, und für die Analyse dieses Bereiches dürfte die durch die Samplingrate des BiSP-Pens vorgegebene Auflösung kaum ausreichend sein.

[7] Die hier gemachte Beobachtung rechtfertigt übrigens rückwirkend erst die Vorgehensweise im vorangegangen Abschnitt: Wir hatten dort die Fouriertransformation jeweils auf die *kompletten* Signalkomponenten angewandt, was i.a. nur dann zu zuverlässigen Resultaten führt, wenn der Frequenzgang *zeitinvariant* ist.

Kapitel 4

Beschreibung des eigenen Verfahrens

In diesem Kapitel wird das vom Autor neu entwickelte Verfahren zur Modellierung und Verifikation der Unterschriftendynamik im Detail beschrieben. Im Verlauf der Diskussion werden verschiedene *Systemparameter* definiert. Die für den Betrieb des Verifikationssystems gewählten Werte für diese Parameter werden in abgesetzten „ Boxen “ dargestellt.

4.1 Datenvorverarbeitung

Dieser Abschnitt behandelt die vom Autor eingesetzten Methoden zur Datenvorverarbeitung im Sinne des Grundlagenabschnitts 2.5.2. Die auf diese Weise aufbereiteten Unterschriften dienen als Grundlage sowohl des Enrollments in 4.2 als auch der Verifikation in 4.4.

4.1.1 Glättung des Signalverlaufs

In 3.1.2 waren wir auf ein oft deutliches, das Nutzsignal überlagerndes, vermutlich weißes Rauschen aufmerksam geworden. Dieses Störsignal scheint alle möglicherweise sonst noch vorhandenen Störanteile zu dominieren, sodaß wir uns hier auf dessen Beseitigung konzentrieren können. Wir verwenden hierfür den Moving-Average-Filter (2.8) aus 2.5.2.1.5, S. 19. Für die Bestimmung einer geeigneten Kernellänge L_{MA} orientieren wir uns an zwei Beobachtungen:

- Es lassen sich gemäß 3.2 Nutzsignal-Frequenzen bis maximal $0.1 f_S$ feststellen. Dies bedeutet, daß in der Zeitdomäne mit informationstragenden Strukturbreiten bis hinab zu 10 Punkten zu rechnen ist.
- Für das betrachtete Rauschen wurden nach 3.1.2 Stärken von bis zu 8 Quantisierungsstufen gemessen. In den meisten Fällen beträgt die Stärke allerdings nur 1–2 Stufen.[1]

Wir müssen die Länge unseres Filters dergestalt wählen, daß das Rauschen einerseits hinreichend stark reduziert wird, um folgende Bearbeitungsstufen nicht zu beeinträchtigen, andererseits aber sollte möglichst wenig Strukturinformation verloren gehen. Wir wählen als Filterlänge

$$\boxed{L_{\mathrm{MA}} := 9} \qquad \textit{(Filterlänge MA-Filter)} \tag{4.1}$$

Damit bleiben wir im Größenbereich der kleinsten auftretenden Strukturen, sollten aber gemäß 2.5.2.1.5 zugleich in der Lage sein, das Rauschen auf ca. $1/\sqrt{L_{\mathrm{MA}}} = 1/3$ der Ausgangsstärke zu reduzieren, was nach obiger Beobachtung in den meisten Fällen eine Reduktion auf weniger als eine Quantisierungsstufe zur Folge hätte.

[1] Bezogen auf den maximal möglichen Bereich von $2^{10} = 1024$ Stufen beim BiSP-Pen bedeuten selbst 8 Quantisierungsstufen weniger als 1%. Allerdings nutzen die Signale diesen Bereich nur selten vollständig aus und nehmen die meiste Zeit über erheblich niedrigere Werte an (vgl. 3.1.3). Das hier betrachtete Rauschen kann also nicht von vornherein als unproblematisch abgetan werden.

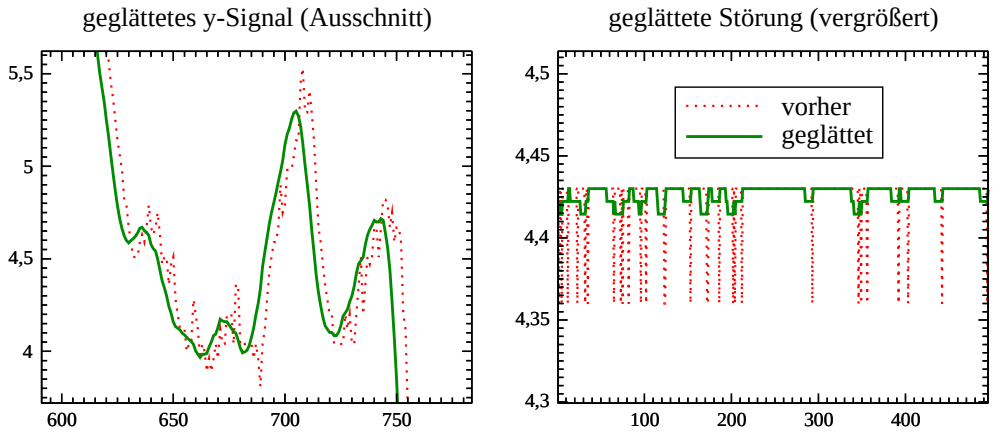

Abbildung 4.1: Auswirkung eines Moving-Average-Filters der Länge 9

Abb. 4.1 zeigt die Auswirkungen des von uns gewählten Filters auf ein reales Signal. Die rechte Seite zeigt einen Ausschnitt des DC-Anteils mit besagtem starken Rauschen. Die Amplitude des Ergebnissignals (grün durchgezogen) ist hier tatsächlich auf weniger als 1/3 des Ausgangssignals (rot gepunktet) reduziert worden.[2] Für das Zeitsignal läßt sich eine weitere in 2.5.2.1.5 genannte Eigenschaft des MA-Filters bestätigen, nämlich daß eine gute Signalglättung mit dem weitgehenden Erhalt der ursprünglichen Flankensteilheit einhergeht.

4.1.2 Tiefpaßfilterung

Wie in 3.2 beobachtet besitzt der Frequenzgang von Unterschriften schreiberunabhängig nur im Bereich unterhalb von $0.1 f_S$ nennenswerte Nutzinformation, bei höheren Frequenzen läßt sich der Amplitudenverlauf nicht vom Grundrauschen unterscheiden. Wir können auf die Unterschriftensequenzen daher eine *Tiefpaßfilterung* anwenden, um hochfrequente Einflüsse als potentielle Störungen zu eliminieren. Dazu werden wir den *Windowed-Sinc-Filter* (2.10), wie er in 2.5.2.1.6 beschrieben wurde, einsetzen, mit der *Cutoff-Frequenz*

$$\boxed{f_C := 0.1 f_S} \qquad \textit{(Cutoff-Frequenz Tiefpaßfilter)} \tag{4.2}$$

Als *Übergangsbandbreite* B wählen wir $B := 0.025 f_S$, was ausreichend schmal für unsere Zwecke erscheint. Dies führt gemäß (2.9) auf S. 23 auf einen Filterkernel der Länge

$$\boxed{L_{\mathrm{WSinc}} := 160} \qquad \textit{(Filterlänge Tiefpaßfilter)} \tag{4.3}$$

Abb. 4.2 zeigt die Anwendung des Filters auf ein reales Signal. Das Schaubild links oben zeigt die Frequenzdomäne unterhalb von $0.1 f_S$. Das Frequenzspektrum in diesem Bereich ist vor (rot gepunktet) und nach (grün durchgezogen) der Filterung nahezu identisch. Die Auswirkungen des Filters sind im Schaubild rechts oben zu sehen. Durch den Filter werden die Amplituden der Frequenzen jenseits der Cutoff-Frequenz f_C wie vorgesehen um mehrere Zehnerpotenzen im Vergleich zum Ausgangsfrequenzgang reduziert. Auch die Übergangsbreite B entspricht recht gut den Vorgaben. In den unteren Schaubildern erkennt man die Auswirkungen des Tiefpaßfilters auf das *Zeitsignal*: Das gefilterte Signal wirkt abgerundeter als das Ausgangssignal. Eine deutliche Verzerrung des grundsätzlichen Signalverlaufs ist mit bloßem Auge nicht auszumachen.

[2] Dies ist ein weiteres Indiz für die in 3.1.2 formulierte Vermutung, daß es sich bei der beobachteten Störung wirklich um weißes Rauschen handelt.

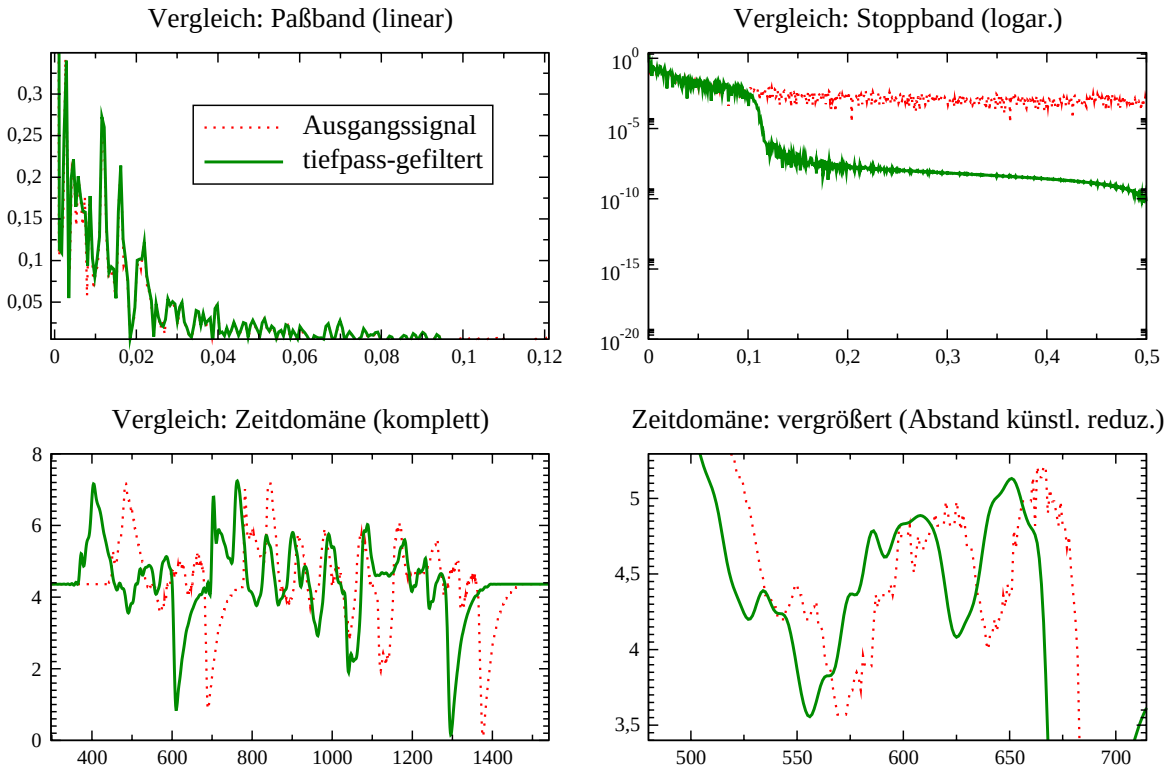

Abbildung 4.2: Auswirkung eines Windowed-Sinc-Filters der Länge 160 mit $f_C = 0.1$

4.1.3 Problemdiskussion: Resonanzen

Das Frequenzspektrum der uns vorliegenden Unterschriften hatte in 3.2.1 keinen deutlichen Hinweis auf das Vorhandensein von Störresonanzen, wie sie in 2.5.2.1.2.1 diskutiert wurden, ergeben. Insofern erübrigt sich die Anwendung von Methoden zu deren Beseitigung an dieser Stelle. Würde das Verifikationssystem auf anderes Datenmaterial zu portieren sein, so hätte man ggf. über den Einsatz eines *Bandreject-Filters* gemäß 2.5.2.1.7 nachzudenken.

4.1.4 Beseitigung („Trimmen“) der Signal-Ränder

Alle Einzelsequenzen besitzen am linken und rechten Rand einen mehr oder minder langen DC-Anteil (vgl. Abb. 3.1 auf S. 46), welcher sich störend auf die weiteren Prozeßstufen auswirken kann. Beispielsweise beeinflußt er das Ergebnis einer Datenzentrierung gemäß 2.5.2.2: Sei $(u'_t) := (\tilde{u}^l_t) \circ (u_t) \circ (\tilde{u}^r_t)$ eine Unterschriftenprobe der Länge T mit den DC-Rändern $(\tilde{u}^l_t)$ und $(\tilde{u}^r_t)$ und der „eigentlichen“ Unterschrift (u_t). Die Länge von (u_t) betrage αT, mit einer Konstanten $\alpha < 1$, und es sei $\mu := \langle u_t \rangle$ der für die Zentrierung relevante Mittelwert. Die Länge der Ränder $(\tilde{u}^l_t)$ und $(\tilde{u}^r_t)$ zusammengenommen beträgt somit $(1-\alpha)T$, den konstanten Wert ihrer Samples $\tilde{u}^{l/r}_t$ bezeichnen wir mit d. Der Gesamtmittelwert $M := \langle u'_t \rangle$ der Sequenz ergibt sich damit zu

$$M = (1-\alpha) \cdot d + \alpha \cdot \mu = \alpha \cdot (\mu - d) + d$$

Falls $\alpha \ll 1$, d.h. falls der relative Anteil der Ränder $(\tilde{u}^l_t)$ und $(\tilde{u}^r_t)$ an der Gesamtsequenz (u'_t) hoch ist, dann ist der tatsächlich gemessene Mittelwert M in starkem Maße von d abhängig und weicht evtl. deutlich vom eigentlich zu bestimmenden Mittelwert μ ab. Speziell für $d = 0$ ergibt sich $M = \alpha\mu$, sodaß eine den weiteren Prozeßstufen vorangeschaltete Verschiebung der DC-Komponente für kleines α keinen Gewinn verspricht.

Für das Entfernen bzw. das *„Trimmen“* der Signalränder der Unterschriftensequenz ergibt sich das Problem, daß die DC-Anteile verrauscht sein können, wie in 3.1.2 gezeigt. Die in 4.1.1

angegebene Methode zur Signalglättung hatte sich zwar als leistungsstark herausgestellt, eine perfekte Begradigung der DC-Anteile wird man von ihr aber nicht erwarten dürfen. Algorithmus 4.1 liefert eine Heuristik zur Bestimmung einer Trimm-Stelle für das Entfernen des *linken* Randes einer *einzelnen* Komponentensequenz. Als linke Trimmstelle für eine *komplette* Unterschrift wird diejenige linke Trimmstelle aller Komponentensequenzen mit *maximalem* Index verwendet. Eine *rechte* Trimmstelle läßt sich durch Anwendung des Algorithmus auf eine *invertierte* Komponentensequenz bestimmen.

Algorithmus 4.1 Trimmen des linken Randes einer Komponentensequenz $(x_t)_{0 \leq t \leq T}$

Require: Fenstergröße W_{Trim}
Require: Offset O_{Trim}
Require: Threshold τ_{Trim}
  bestimme maximal mögliche Auslenkung $A := \max_t(x_t) - \min_t(x_t)$ des Signals
  schätze DC-Wert ab: $\hat{d} := \frac{1}{W_{\text{Trim}}} \sum_{i=0}^{W_{\text{Trim}}-1} x_i$
  for $t := 0$ to $T - 2W_{\text{Trim}}$ **do**
    schätze nachfolgende Werte ab: $\hat{x}_t := \frac{1}{W_{\text{Trim}}} \sum_{l=1}^{W_{\text{Trim}}} x_{t+O_{\text{Trim}}+l}$
    if $|\hat{x}_t - \hat{d}| > \tau_{\text{Trim}} \cdot A$ **then**
      return t {Trimm-Stelle gefunden}
    end if
  end for

Es wird zunächst am äußeren Rand der Sequenz der Durchschnittswert $\hat{d}$ der dort in einem *Fenster* der Größe W_{Trim} liegenden Samples bestimmt. Der Wert $\hat{d}$ dient als Schätzung des „wahren“ DC-Wertes. Durch die Mittelung werden ggf. auftretende Schwankungen ausgeglichen. Die Sequenz wird nun durchlaufen, wobei für jede Stelle t in analoger Weise eine Schätzung $\hat{x}_t$ des unverrauschten Sequenzsamples an dieser Position ermittelt wird. Tatsächlich wird die Berechnung des Schätzers auf um einen *Offset* O_{Trim} *hinter* der Stelle t liegenden Samples durchgeführt, womit ein Beschädigen des Anfangs des Nutzsignals vermieden werden soll. Falls sich der Sample-Schätzer $\hat{x}_t$ vom DC-Schätzer $\hat{d}$ hinreichend stark unterscheidet, so terminiert das Verfahren mit der Ausgabe der Position t. Für die Definition des Abbruchkriteriums wurde im ersten Schritt des Algorithmus eine Obergrenze A der *maximalen Signalauslenkung* berechnet; überschritten werden muß der Wert $\tau_{\text{Trim}} \cdot A$ mit einem noch näher zu spezifizierenden *Threshold* $\tau_{\text{Trim}} > 0$.

Die Laufzeit des Verfahrens wird offenbar durch die FOR-Schleife dominiert. Zur Bestimmung der beiden Trimmstellen für die Gesamtunterschrift ist nur eine *konstante* Anzahl von Anwendungen des Algorithmus nötig. Die Komplexität beträgt somit $\mathrm{O}(W_{\text{Trim}} \cdot T)$.

Die Leistung des Algorithmus wird maßgeblich von den Werten der Parameter W_{Trim}, O_{Trim} und τ_{Trim} abhängen. Bei der Wahl der Fenstergröße W_{Trim} ist die minimale Länge informationstragender Strukturen bei Unterschriften zu beachten, die gemäß den Ausführungen in 3.2 bei etwa 10 Punkten liegt. Wir wählen:

$$\boxed{W_{\text{Trim}} := 11} \qquad \textit{(Fensterlänge Trimmer)} \tag{4.4}$$

Der „Lookahead-Offset“ O_{Trim} sollte 1–2% der Unterschriftenlänge nicht übersteigen, damit die verbleibenden DC-Anteile keinen störenden Einfluß auf die weiteren Prozeßstufen haben. Wir wählen:

$$\boxed{O_{\text{Trim}} := 20} \qquad \textit{(Trimmer-Offset)} \tag{4.5}$$

Der Threshold τ_{Trim} ist im Prinzip in Abhängigkeit des Signal/Rausch-Verhältnisses zu wählen. Experimente zu seiner Bestimmung haben die folgende Wahl nahegelegt:

$$\boxed{\tau_{\text{Trim}} := 0.08} \qquad \textit{(Trimmer-Threshold)} \tag{4.6}$$

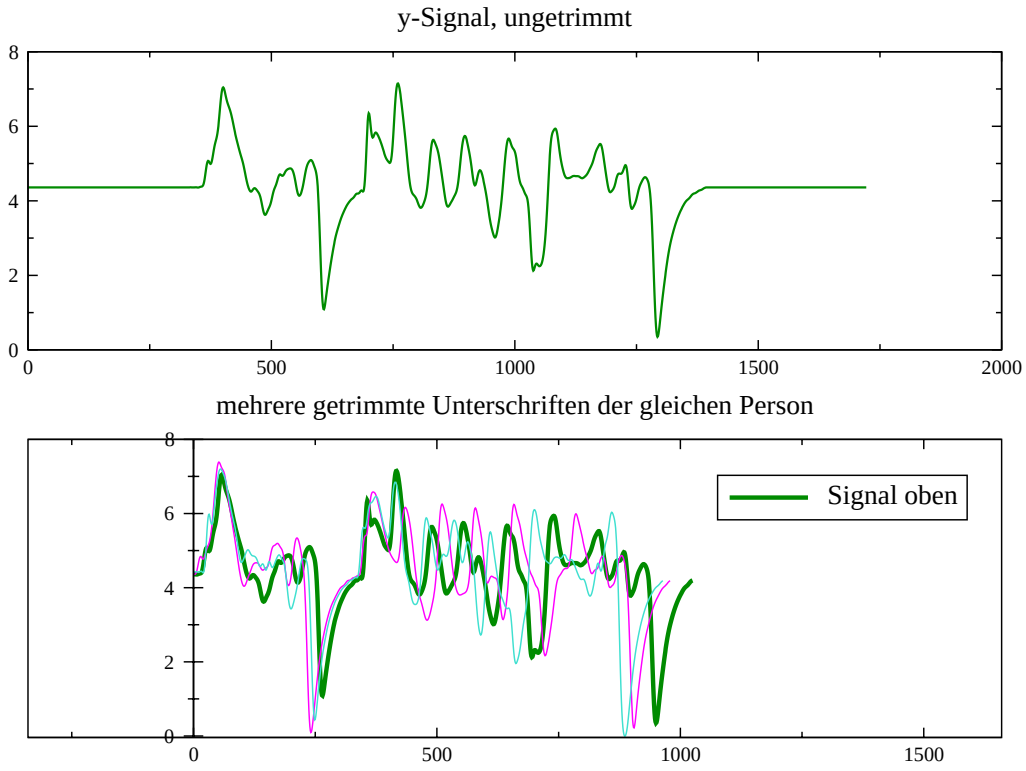

Abbildung 4.3: Unterschrift vor und nach der automatischen Trimmung

Zur Überprüfung der Qualität und Zuverlässigkeit der beschriebenen Heuristik wurden zum einen 100 Unterschriften eines bestimmten Probanten, zum anderen je 2 Unterschriften aller Probanten getrimmt und die Ergebnisse visuell begutachtet. In sämtlichen untersuchten Fällen lagen beide Trimmstellen in der Nähe des Anfangs bzw. Endes der eigentlichen Sequenz, wobei der verbliebene DC-Anteil in den meisten Fällen nur wenige Punkte betrug. Nur in Einzelfällen wurde ein wenige Punkte langes Randstück des Nutzsignals abgeschnitten. Abb. 4.3 zeigt als Beispiel das Ergebnis für mehrere Unterschriften der gleichen Person im Vergleich.

4.1.5 Daten-Zentrierung und Amplituden-Normierung

Entsprechend den Ausführungen in 2.5.2.2 werden die einzelnen Komponenten der Unterschriftensequenz *zentriert* und danach *amplitudennormiert*. Wie im Grundlagenkapitel argumentiert wurde, gehen hierdurch keine für uns wesentlichen Informationen verloren. Einige der Schritte des *Enrollments* in 4.2, der *Verifikation* in 4.4, und auch der in 4.5 vorgestellte *Rotationsausgleich* werden die durch diese beiden Normalisierungsschritte gewonnenen Dateneigenschaften allerdings benötigen.

Abb. 4.4 zeigt ein Beispiel für die Auswirkungen der Zentrierung und Amplitudennormierung. Das dargestellte Signal wurde zuvor *getrimmt* gemäß 4.1.4. Wie man sieht, ändert sich an der grundsätzlichen Charakteristik des Signals nichts, jedoch werden die x- und die y-Komponente in ihrer Amplitude verstärkt, während die im Vergleich zu diesen durch mehrere starke Peaks zu kräftig wirkende z-Komponente abgeschwächt wird. Im Ergebnis stehen alle drei Komponenten hinsichtlich ihrer Amplitudendynamik weitgehend „gleichberechtigt“ nebeneinander. Diese Beobachtung deckt sich mit anderen vom Autor betrachteten Beispielen. Übrigens wird aus dem Bild auch klar, daß man, wie bereits an früherer Stelle bemerkt, i.a. nicht erwarten darf, daß durch die Zentrierung die DC-Anteile der Signalkomponenten auf die 0-Linie verschoben werden.

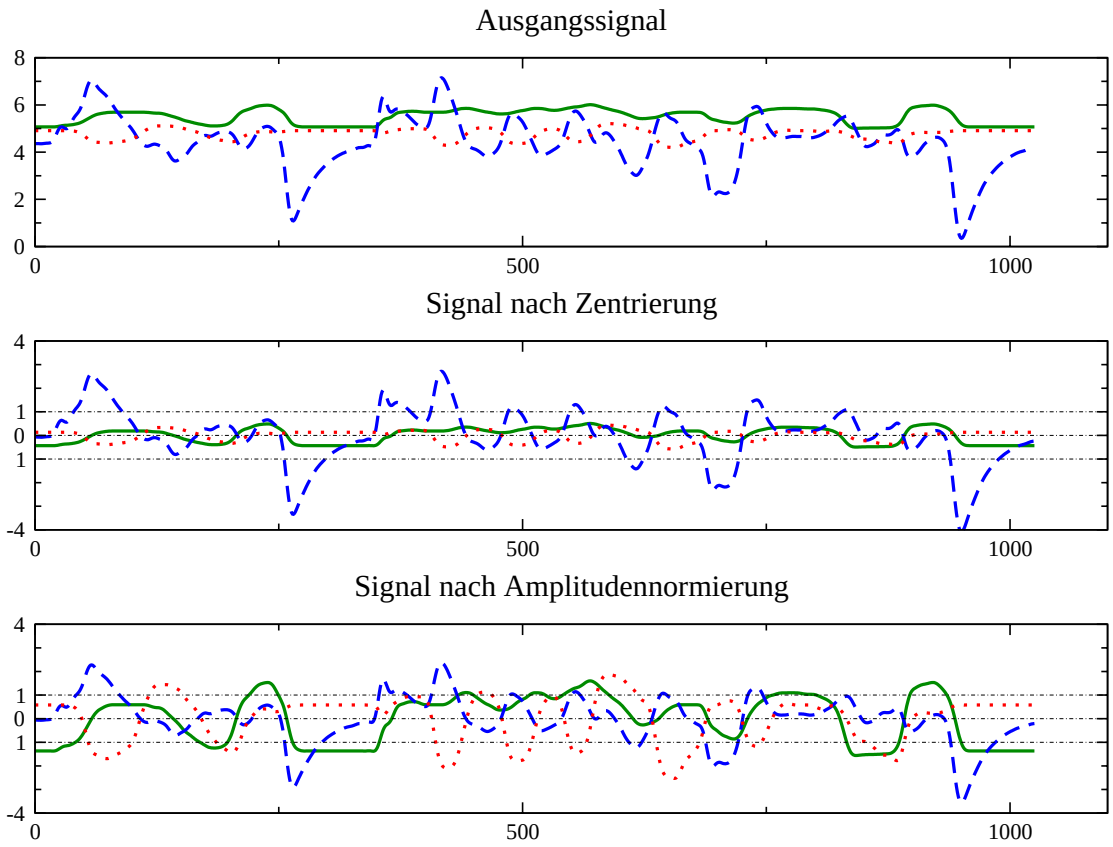

Abbildung 4.4: Zentrierung und Amplitudennormierung einer Unterschrift

4.1.6 Problemdiskussion: Stiftrotation

In 3.1.4 hatten wir die mit der *Stiftrotation* um die Längsachse verbundenen Probleme diskutiert. Die für diese Arbeit verwendeten Daten liegen zwar mit fester Anfangsrotation vor, doch in der Praxis kann man dies nicht voraussetzen, und auch die in Kapitel 3 erwähnte *neue* Version des BiSP-Pens mit ihrer asymmetrischen Formgebung mag das völlig beliebige Rotieren erschweren, größere Schwankungen in der Ausgangsrotation zweier Unterschriftenproben sind dadurch aber dennoch nicht ausgeschlossen. Es sei bemerkt, daß uns lediglich die *Anfangsrotation*, also die Rotation des Stiftes zu Beginn des Schreibprozesses, vor Probleme stellt. Die *dynamische* Rotation des Stiftes während der Schreibbewegung könnte nach Vermutung des Autors hingegen bei Unterschriften ein Personenspezifikum sein, sodaß deren Beibehaltung der Verifikationsleistung vielleicht sogar dienlich ist.[3]

Wir betrachten zwei Unterschriften $u^{(1)}$ und $u^{(2)}$, wobei $u^{(2)}$ aus $u^{(1)}$ durch Phasendrehung um einen konstanten Winkel ω^* hervorgegangen sei, geschrieben als: „$u^{(2)} = \varphi_{\omega^*}(u^{(1)})$“. Idealerweise würden wir nach einem Algorithmus suchen, welcher für $u^{(1)}$ $(u^{(2)})$ einen Winkel $\omega^{(1)}$ $(\omega^{(2)})$ berechnet, sodaß unabhängig von ω^* die Beziehung $\varphi_{\omega^{(1)}}(u^{(1)}) = \varphi_{\omega^{(2)}}(u^{(2)})$ gilt. Es ist dem Autoren nicht gelungen, einen derartigen *„Rotationsangleich“* zu entwickeln, der die gestellte Aufgabe mit großer Zuverlässigkeit in ausreichender Näherung löst, sofern ausschließlich die jeweils zu rotierende Unterschrift u betrachtet werden darf. Die Beseitigung der Ausgangsrotation ist daher in dieser Arbeit nicht Bestandteil der Datenvorverarbeitung; sie wird stattdessen gesondert in 4.5 behandelt werden.

[3] Von Seiten des Autors wird angeregt, daß bei der Entwicklung zukünftiger Versionen des BiSP-Pens auch über die Erfassung der Rotationsdynamik als zusätzliches Merkmal nachgedacht wird.

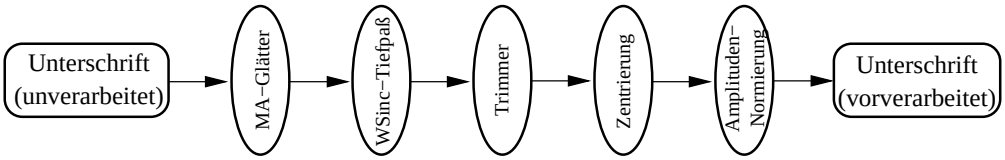

Abbildung 4.5: Ablauf des Vorverarbeitungsprozesses

4.1.7 Problemdiskussion: Stiftneigung

In 3.1.5 waren wir auf das Phänomen der *Stiftneigung* aufmerksam geworden, und wir hatten am Beispiel der Abb. 3.5 auf S. 51 gesehen, wie sie sich auf reale Daten auswirken kann. Da der BiSP-Pen, in der von uns verwendeten Version, die Dynamik der Neigung bislang nicht als zusätzliches Paar zweier Merkmale (x- und y-Tilt) erfaßt, muß man grundsätzlich davon ausgehen, daß die Neigung die anderen Signalkomponenten als Störung überlagert. Der Autor hält es jedoch für realistisch, daß sowohl die *Ausgangsneigung* zu Beginn des Schreiben als auch die *Neigungsdynamik* während des Schreibens bei Unterschriften stets in sehr ähnlicher und darüber hinaus in *personenspezifischer* Weise auftritt. Dadurch sollte das Verifikationspotential nicht allzu sehr beeinträchtigt, evtl. sogar unterstützt werden. Es erscheint daher von diesem Standpunkt aus betrachtet nicht dringend, die Stiftneigung als Störsignal vom restlichen Nutzsignal abzutrennen. Eine zusätzliche Erfassung wäre allerdings – aus den genannten Gründen – wünschenswert.

4.1.8 Zusammenfassung des Datenvorverarbeitungsprozesses

Abb. 4.5 faßt die einzelnen Schritte der Datenvorverarbeitung zusammen. Eingabe ist die „rohe“ mit dem BiSP-Pen erfaßte Unterschriftensequenz. Ausgegeben wird die vorverarbeitete Unterschriftensequenz, wobei sich deren Format (drei Komponentensequenzen) nicht verändert hat. Die Eingabesequenz wird mit dem *Moving-Average-Filter* aus 4.1.1 *geglättet* und mit dem *Windowed-Sinc-Filter* aus 4.1.2 *tiefpaßgefiltert*. Diese beiden Schritte sind aufgrund der *Kommutativität der Konvolution* gemäß (2.7) auf S. 19 vertauschbar. Die anschließende *Trimmer-Stufe* (4.1.4) beseitigt die nicht zur eigentlichen Unterschrift gehörenden Signalränder. Das verbleibende Signal wird komponentenweise *zentriert* und *amplitudennormiert*, wobei für die Normierung die *Standardabweichungen* der Amplitudenwerte zugrundegelegt werden (4.1.5).

Die folgende Liste faßt die Werte sämtlicher Systemparameter zusammen, die in den vorangegangenen Abschnitten für die Datenvorverarbeitung festgelegt wurden:

- Länge des Glättungsfilters ((4.1) 4.1.1): $L_{\mathrm{MA}} = 9$
- Länge des Tiefpaßfilters ((4.3) 4.1.2): $L_{\mathrm{WSinc}} = 160$
- Cutoff-Frequenz des Tiefpaßfilters ((4.2) 4.1.2): $f_C = 0.1$
- Fensterlänge des Trimmers ((4.4) 4.1.4): $W_{\mathrm{Trim}} = 11$
- Trimmer-Offset ((4.5) 4.1.4): $O_{\mathrm{Trim}} = 20$
- Trimmer-Threshold ((4.6) 4.1.4): $\tau_{\mathrm{Trim}} = 0.08$

4.2 Enrollment

Es wird nun das Verfahren zur Modellierung der Unterschrift einer Person präsentiert. Ausgangspunkt ist die Arbeitshypothese, nach der zwei Unterschriftenproben der *gleichen* Person Folgen einander *ähnlicher* Strukturen an *ähnlichen relativen* Positionen in der Unterschriften-Zeitsequenz sind. Es wird sich dabei nicht ausschließen lassen, daß gelegentlich Strukturen auftreten, die in anderen Unterschriftenproben der gleichen Person nicht vorkommen. Falls aber die Unterschrift einer Person wirklich, wie in in 2.2 diskutiert, „am Stück“ im Gehirn repräsentiert ist, dann sollten *Vertauschungen* zweier Strukturen normalerweise selten auftreten. Falls es eine solche Vertauschung gibt, so stellt sie ein *lokales* Problem dar, mit dem wir umzugehen wissen werden: Betrachten wir dazu die idealisierte Situation zweier Strukturensequenzen $(s_k^{(1)})$ und $(s_k^{(2)})$ mit einer Stelle k^*, an der $s_{k^*}^{(1)} = s_{k^*+1}^{(2)}$ und $s_{k^*+1}^{(1)} = s_{k^*}^{(2)}$ gelte. Die $k^* + 1$-te Struktur kann aus $(s_k^{(2)})$ entfernt werden, womit wir das Problem auf die Situation einer *fehlenden Struktur* zurückgeführt haben.

Unser erstes Ziel wird die Bestimmung der postulierten Folge von Strukturen sein. Mit der im Grundlagenkapitel benutzten Sprechweise werden wir nach einer *Segmentierungsstrategie* suchen (vgl. 2.5.2.3.2). Es ist jedoch völlig unklar, nach welchen Kriterien eine solche Segmentierung zu erfolgen hat. Allgemeine Theorien über den Aufbau oder die Erzeugung von Unterschriften sind, wie bereits in Kapitel 1 gesagt, dem Autor während der Beschäftigung mit der vorliegenden Arbeit nicht begegnet. Deshalb werden wir hier nicht fragen, von welcher *Gestalt* Segmente sind, sondern wir werden uns auf die weniger schwierig erscheinende Frage beschränken, wie man geeignete *Segmentierungsstellen* identifiziert. Unser Kriterium zur Bestimmung solcher Stellen wird vage formuliert lauten, daß nach Zeitpunkten t^* in der Zeitsequenz gesucht wird, an denen die zugehörigen Samples u_{t^*} eine *„unvorhersagbare“* Form besitzen. Die Detektion solcher „Points of Interest“ wird mit Hilfe sog. *„Prädiktoren“* geschehen. Letztere werden in dem hier vorgestellten Ansatz aber nicht fest vorgegeben, sondern *adaptiv* anhand des Trainingsdatenmaterials gelernt werden.

Nach der Segmentierung sollten Unterschriften der gleichen Person eine ähnliche Anzahl zueinander passender Segmente besitzen. Die einzelnen Segmente werden einer *Featureextraktion* (vgl. 2.5.3) unterworfen, auf deren Grundlage es ermöglicht werden soll, zusammenpassende von nicht zusammenpassenden Segmenten zu unterscheiden. Auf den so entstandenen *Codevektor-Sequenzen* kann dann ein *Sequence-Alignment* mit dem in 2.5.5.2.1 vorgestellten DTW-Algorithmus durchgeführt werden. Es wird uns in dieser Arbeit vor allem darum gehen, die *„wahre“* Codesequenz einer Person zusammen mit ihrer *„spezifischen Variabilität“* in allen Segmenten zu bestimmen. Wir hatten etwas Entsprechendes bereits für das Beispielmodell aus 2.5.4.2 durchgeführt, indem wir dort die Mittelwerte und Standardabweichungen in allen Features der Trainingsunterschriften berechnet hatten. Allerdings war in diesem Fall eine Unterschrift als *einzelner Vektor* repräsentiert, was die Arbeit stark vereinfacht hat im Vergleich zum hier betrachteten Szenario.

Es wird alsdann um die Identifizierung der jeweils das gleiche Segment repräsentierenden Segmentcodierungen aus *allen* Trainingsunterschriften gehen, um diese Mengen inhaltlich zueinander passender Segmente einer statistischen Analyse unterziehen zu können. Im Prinzip sollte dies dadurch gelingen, daß man für je zwei Unterschriften ein Sequence-Alignment durchführt, und dann die jeweils paarweise assoziierten Segmentcodierungen zusammenfaßt. In der Praxis hat sich allerdings herausgestellt, daß auf diese Weise allein eine akzeptable Aufteilung der Segmente in Mengen zusammenpassender Exemplare nur in seltenen Fällen entsteht. Es wird daher eine der Hauptleistungen dieser Arbeit sein, eine solche Zerlegung, welche die Unterschrift der Person tatsächlich in adäquater Weise zu repräsentieren vermag, nachträglich aus der Menge aller paarweisen Alignments zu „rekonstruieren“.

Die Eingabe für das Enrollment besteht aus einer Menge von gemäß 4.1 vorverarbeiteten Trainingsunterschriften der zu modellierenden Person. In das zu generierende Modell wird keinerlei Information von Fälschungen oder Fremdunterschriften einfließen.

4.2.1 Segmentierung der Trainingsunterschriften

Ziel dieses Abschnittes ist es, eine gegebene Unterschriften-Zeitsequenz in eine Folge von Segmenten zu zerlegen. Wie bereits in 2.5.2.3.2 besprochen, kann durch eine geeignete Segmentierung implizit ein *Längenangleich* verschiedener Unterschriften der gleichen Person erfolgen. Wir werden jedoch keine der im Grundlagenteil angegebenen Methoden verwenden, sondern stattdessen eine *adaptive* Segmentierungsstrategie entwickeln.

4.2.1.1 Lineare Prädiktoren

Die Idee zur Segmentierung wird die Annahme sein, daß sich ein Sample *innerhalb* eines Segmentes gut durch Betrachtung der zeitlich *benachbarten* Samples *vorhersagen* läßt. Eine Zeitstelle t^*, für deren Sample u_{t^*} eine solche Vorhersage scheitert, soll dann als *Trennstelle* zwischen zwei Segmenten in Frage kommen. Dabei müssen zwei identische Samples, die sich an unterschiedlichen Stellen in der Unterschrift befinden, sich *nicht* notwendigerweise gleichermaßen als Kandidaten für Segmentierungsstellen eignen; entscheidend hierfür ist vielmehr ihre Stellung innerhalb des *lokalen Kontextes* in der Zeitsequenz. Wir formalisieren diesen Ansatz mit Hilfe des Begriffs des *„linearen Prädiktors“*:

Definition 4.1 (Linearer Prädiktor). *Sei* $(\boldsymbol{x}_t)$ *eine Zeitreihe mit Samples* $\boldsymbol{x}_t \in \mathbb{F}$. *Eine Funktion* $\pi_{r,\boldsymbol{\alpha}} : \mathbb{F}^+ \times \mathbb{Z} \to \mathbb{F}$, *definiert durch*

$$\pi_{r,\boldsymbol{\alpha}}((\boldsymbol{x}_t), t^*) := \hat{\boldsymbol{x}}_{t^*} := \sum_{-r \le i \le +r} \alpha_i \cdot \boldsymbol{x}_{t^*+i}\,, \quad \alpha_i \in \mathbb{R}\,, \quad \alpha_0 = 0\,, \quad \sum_{-r \le i \le +r} \alpha_i = 1$$

heißt linearer Prädiktor *mit* Radius r *und* Koeffizientenvektor $\boldsymbol{\alpha}$. *Ferner ist der* Prädiktionsfehler $\tilde{x}_{t^*} \in \mathbb{R}_0^+$ *an der Stelle* t^* *definiert durch*

$$\tilde{x}_{t^*} := \|\boldsymbol{x}_{t^*} - \hat{\boldsymbol{x}}_{t^*}\|\,.$$

In Def. 4.1 wird mit „$\alpha_0 = 0$“ ausgedrückt, daß das Sample $\boldsymbol{x}_t$ selbst *nicht* für die Berechnung seiner Vorhersage $\hat{\boldsymbol{x}}_t$ eingesetzt werden darf, andernfalls würde ein Koeffizientenvektor der Form $\alpha_i := \delta_{i0}$ stets perfekte Prädiktionen ermöglichen. Die Normierung der Form „$\sum_{i=-r}^{+r} \alpha_i = 1$“ wird benötigt, um beispielsweise DC-Anteile eines Signals sicher vorherzusagen, und stellt ansonsten keine Einschränkung für unsere Zwecke dar. Es sei erwähnt, daß man sich auch andere Formen von Prädiktoren überlegen kann, z.B. werden im Gebiet der *Datenkomprimierung* verschiedene Ansätze hierzu verfolgt [Say96]. Wir werden uns aber in dieser Arbeit auf *lineare* Prädiktoren beschränken, um deren Leistungspotential für die hier gestellte Aufgabe zu ergründen. Wo immer dies nicht zu Mißverständnissen führt, werden wir daher in der Folgezeit lediglich von „Prädiktoren“ sprechen, wenn wir „lineare Prädiktoren“ meinen.

Ein Prädiktor liefert uns also zu einer Unterschriftensequenz $(u_t)_{0 \le t \le T}$ eine *Vorhersagesequenz* $(\hat{u}_t)_{0 \le t \le T}$ gleicher Länge, und aus diesen beiden Zeitsequenzen läßt sich eine Sequenz $(\tilde{u}_t)_{0 \le t \le T}$ für den *Vorhersagefehler* bestimmen. Gemäß unserer zu Beginn vorgestellten Idee werden wir in dieser Fehlersequenz nun nach Stellen mit relativ großem Wert $\tilde{u}_t$ suchen müssen. Zuvor haben wir uns allerdings noch mit der Wahl geeigneter Prädiktoren zu beschäftigen. Ein triviales Beispiel für einen Prädiktor wäre durch

$$\hat{u}_{t^*} = \pi^{=}_{r,\boldsymbol{\alpha}}((u_t), t^*) := \sum_{-r \le i \le +r} \delta_{-1,i} \cdot u_{t^*+i} = u_{t^*-1} \qquad \text{(„Next = This“)} \tag{4.7}$$

gegeben, also die Strategie, als Vorhersage stets das zeitlich vorangegangene Sample zu wählen. Wir werden aber in der Lage sein, zumeist leistungsfähigere Prädiktoren zu erzeugen als diesen.

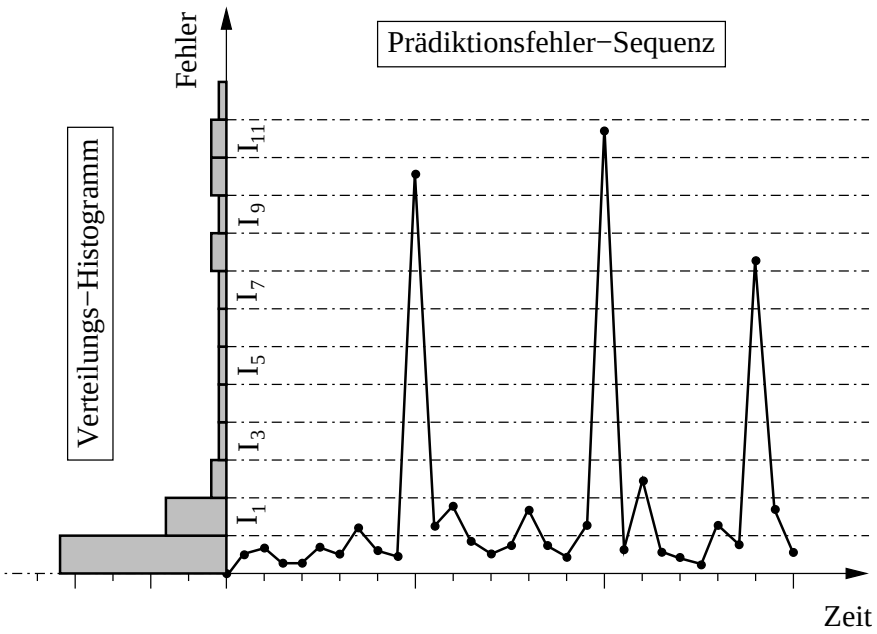

Abbildung 4.6: Konstruierte Darstellung einer „guten“ Prädiktionsfehlersequenz

4.2.1.2 Adaptive Prädiktor-Generierung

Unser Ziel wird es sein, für eine Menge von Trainingsunterschriften $u^{(1)}, \ldots, u^{(m)}$ einen Prädiktor $\pi_{r,\boldsymbol{\alpha}}$ anzugeben, welcher folgende bewußt vage formulierte Eigenschaft besitzt:

> Die Prädiktionsfehler-Sequenzen für sämtliche Trainingsunterschriften sollen längere Abschnitte mit sehr niedrigen Fehlerwerten besitzen, welche durch einzelne Stellen sehr hohen Fehlers unterbrochen werden.

Die genannten Fehlermaxima sollen später als Trennstellen zwischen den zu bestimmenden Segmenten verwendet werden.

4.2.1.2.1 Bewertung eines Prädiktors Abb. 4.6 zeigt ein konstruiertes Beispiel für eine Prädiktionsfehlersequenz, welche nach obiger Forderung als „gut“ eingestuft werden müßte. Für unsere Zwecke wird es sich als zweckmäßig erweisen, wenn wir den Wertebereich $\mathbb{R}_0^+$ der Fehlersequenz zunächst *äquidistant quantisieren* (vgl. 2.3.2.1), wobei die Länge der Quantisierungsintervalle $I_q \subset \mathbb{R}_0^+$ für Quantisierungsstufen $q \in \mathbb{N}_0$ noch festzulegen sein wird. Für unser Beispiel wurde eine solche Quantisierung exemplarisch durchgeführt und das *Verteilungs-Histogramm*[4] der den einzelnen Quantisierungsstufen zugeordneten Fehlerwerten entlang der Ordinate aufgetragen. Die Hauptmasse der Verteilung ist offensichtlich bei den niedrigsten Quantisierungsstufen angesiedelt, während die wenigen hohen Fehlerwerte über den restlichen Bereich des Histogramms verteilt sind.

Wir erinnern an den Begriff der *Entropie*:

Definition 4.2 (Entropie). *Sei p eine Zufallsverteilung über dem diskreten Wahrscheinlichkeitsraum Ω. Die* Entropie $\mathrm{H}(p)$ *für p ist definiert durch*

$$\mathrm{H}(p) := \sum_{x \in \Omega : p(x) > 0} -p(x) \cdot \log p(x)$$

[4] Die Implementierung eines Histogramms für eine auf einem *unbeschränkten* angeordneten Bereich wie $\mathbb{R}_0^+$ verteilte Punktmenge gelingt durch den Aufbau einer *dynamischen Dictionary-Struktur*, beispielsweise realisiert durch einen AVL-Baum [Knu73], mit den Quantisierungsstufen q als Schlüsseln und der Anzahl n_q von in I_q liegenden Punkten als Einträgen. Die Anzahl einer nicht als Schlüssel enthaltenen Quantisierungsstufe q wird als '0' interpretiert. Damit ist der Speicherplatzverbrauch einer solchen Repräsentation proportional zur Anzahl tatsächlich belegter Quantisierungsstufen.

Die Basis des Logarithmus wird in dieser Definition nicht explizit genannt, da sie sich für unsere Zwecke als unerheblich herausstellen wird. Wir werden uns ferner auf die Betrachtung solcher Verteilungen p konzentrieren, welche nur für die Elemente einer *endlichen* Teilmenge $\Omega' \subseteq \Omega$ von 0 verschiedene Wahrscheinlichkeiten besitzen. Für eine solche Verteilung läßt sich zeigen [HQ95], daß stets $\mathrm{H}(p) \geq 0$ gilt, und daß die *maximale* Entropie für die uniforme Verteilung $p_U(x) := 1/|\Omega'|$ angenommen wird. Ihr *Minimum* erreicht die Entropie, wenn es ein $\xi \in \Omega$ gibt mit $p(\xi) = 1$:

$$\mathrm{H}(p) = \sum_{x \in \Omega : p(x) > 0} -p(x) \cdot \log p(x) = -p(\xi) \cdot \log p(\xi) = -1 \cdot 0 = 0$$

Die Fehlersequenzen von Unterschriften sind stets von endlicher Länge, und somit erfüllt auch jede äquidistante Quantisierung Q von $\mathbb{R}_0^+$ die Bedingung, daß das zugehörige Verteilungs-Histogramm nur *endlich* viele nichtleere Quantisierungsintervalle besitzt. Damit hätte die Verteilung einer Fehlersequenz wie diejenige in Abb. 4.6 eine Entropie nahe bei 0. Ein Prädiktor mit schlechter Vorhersagekraft würde sich dagegen eher durch eine Fehlersequenz auszeichnen, deren Werte sich recht gleichmäßig über einen größeren Bereich verteilen, was nach der vorangegangen Besprechung zu einer hohen Entropie führen sollte. Unsere *Bewertungsfunktion* $f_{(u^{(i)})_{1 \leq i \leq m}}(\pi_{r,\boldsymbol{\alpha}})$ für Trainingsunterschriften $u^{(1)}, \ldots, u^{(m)}$ angewandt auf einen Prädiktor $\pi_{r,\boldsymbol{\alpha}}$ werden wir daher wie folgt definieren:

$$f_{(u^{(i)})_{1 \leq i \leq m}}(\pi_{r,\boldsymbol{\alpha}}) := \left\langle \mathrm{H}(p_Q^{(i)}) \right\rangle_i \qquad \textit{(Prädiktor-Bewertungsfunktion)} \tag{4.8}$$

wobei mit $p_Q^{(i)}$ die zur Fehlersequenz $\tilde{u}^{(i)}$ und zur Quantisierung Q gehörende Verteilung bezeichnet sei. Unser Ziel wird es nunmehr sein, nach Prädiktoren π mit einer möglichst *geringen* Bewertung $f(\pi)$ zu suchen. Auf die Wahl der Größe der Quantisierungsintervalle I_q werden wir in 4.2.1.2.3 eingehen.

4.2.1.2.2 Prädiktor-Bestimmung mittels Genetischer Algorithmen Um den bestmöglichen Prädiktor für eine Menge von Trainings-Unterschriften zu bestimmen, können wir aus Gründen des Zeitaufwands nicht einfach für sämtliche möglichen Prädiktoren eines vorgegebenen Radiuses r eine Bewertung mittels der durch (4.8) definierten Zielfunktion durchführen, auch wenn die in Frage kommenden Koeffizienten α_i auf realen Computern aus einem *endlichen* Teilbereich von $\mathbb{R}$ stammen. Tatsächlich benötigen wir nicht wirklich ein *globales* Optimum, ein Prädiktor mit niedriger Bewertung genügt in aller Regel bereits. Aber z.B. für die Anwendung eines *Gradientenverfahrens* [Bra95] zur Approximation des Minimums besteht im Falle unserer auf der Entropie der Verteilung der Vorhersagefehlerwerte basierenden Bewertungsfunktion die Gefahr, in einem *lokalen* Minimum weit weg vom globalen Optimum zu enden.

Einen verhältnismäßig effizienten Ansatz, einen möglichst großen Teil des Definitionsbereichs einer Zielfunktion zu inspizieren und damit die Gefahr schlechter lokaler Optima zumindest einzuschränken, bieten *Genetische Algorithmen* [Bra95]. Bei diesem Ansatz versucht man zunächst, für ein gegebenes Problem eine Menge paarweise möglichst unabhängiger Parameter $\{G_1, \ldots, G_n\}$, *„Gene“* genannt, zu bestimmen, mit denen sich dieses Problem adäquat beschreiben läßt. Jede mögliche Beschreibung des Problems läßt sich nun als Folge $\boldsymbol{g} = g_1 \cdots g_n$ fester Länge beschreiben, wobei sich, um im Bild zu bleiben, die Codierung g_i als *„Ausprägung“* des Gens G_i, und die gesamte Folge $\boldsymbol{g}$ als *„Genom“* der Lösung auffassen läßt. In unserem Fall ist das Problem durch die Suche nach einem für eine Trainingsmenge von Unterschriften $u^{(i)}$ geeigneten Prädiktor $\pi_{r,\boldsymbol{\alpha}}$ bestimmt. Geben wir den Radius r fürs erste fest vor, so sind die Genome gegeben durch die Folgen der *Linearkoeffizienten* α_i, d.h. für die Gene G_i gilt: $G_i = \mathbb{R}$. Benötigt wird ferner eine *„Fitneßfunktion“* $f : G_1 \times \cdots \times G_n \to \mathbb{R}_0^+$, welche mit ihrem Ausgabewert die Qualität einer Lösung $\boldsymbol{g}$ für das gegebene Problem spezifiziert, und die es zu optimieren gilt. Wir haben für unser Problem eine solche Funktion bereits in (4.8) definiert.

4.2.1.2.2.1 Ablauf eines Genetischen Algorithmus Es wird als nächstes die allgemeine Arbeitsweise eines Genetischen Algorithmus beschrieben. In einem vorbereitenden Schritt wird eine *„Ausgangspopulation"* $\Gamma_0 := \{\boldsymbol{g}^{(0,1)}, \ldots, \boldsymbol{g}^{(0,N)}\}$ der Größe N generiert, was zufällig oder aber gemäß eines vom Anwendungsfall abhängigen Kriteriums geschehen kann. Aus der k-ten Population Γ_k entsteht nun die $k+1$-te *„Generation"* Γ_{k+1} durch die folgenden Schritte:

1. *„Selektion":* Für alle Genome $\boldsymbol{g}^{(k,j)}$, $1 \leq j \leq N$, wird die Fitneß $f(\boldsymbol{g}^{(k,j)})$ bestimmt, woraufhin ein Teil der Population mit schlechter Fitneß aus der Population ausscheidet (*„Surviving of the Fittest"*).

2. *„Variation":* Die Population wird wieder auf ihre Ausgangsgröße zurückgeführt, indem aus den verbliebenen Genomen $\boldsymbol{g}^{(k,j)}$ zusätzliche neue Genome durch Anwendung bestimmter *„genetischer Operatoren"* gebildet werden, von denen hier zwei näher beschrieben werden sollen:

   - *„Kreuzung":* Zu zwei Genomen $\boldsymbol{g}^{(k,j)}$ und $\boldsymbol{g}^{(k,l)}$, wir nennen sie die *„Eltern"*, wird eine Position $\lambda \in \{0, \ldots, n\}$ bestimmt und neue Genome $\tilde{\boldsymbol{g}}^{(k,j)}$ und $\tilde{\boldsymbol{g}}^{(k,l)}$, die *„Kinder"*, gebildet mit der Form

   $$\tilde{\boldsymbol{g}}^{(k,j)} = g_1^{(k,j)} \cdots g_\lambda^{(k,j)} \cdot g_{\lambda+1}^{(k,l)} \cdots g_n^{(k,l)}$$
   $$\tilde{\boldsymbol{g}}^{(k,l)} = g_1^{(k,l)} \cdots g_\lambda^{(k,l)} \cdot g_{\lambda+1}^{(k,j)} \cdots g_n^{(k,j)}$$

   Diese Art der Variation soll das Verbleiben in der Nähe schlechter lokaler Optima vermeiden helfen, denn die Kindergenome erhalten Teile des Genoms zweier sich bislang als „fit" herausgestellter Eltern-Lösungen, die aber möglicherweise aus ganz unterschiedlichen Bereichen des Lösungsraumes stammen. Durch die Mischung dieser Gene werden die Kinder ihrerseits möglicherweise in weit von den Eltern entfernte Bereiche gestreut.

   - *„Mutation":* Die einzelnen Gene g_j eines Genoms $\boldsymbol{g}$ werden um einen bestimmten Wert abgeändert. Die Größe dieses Betrags kann beispielsweise zufällig bestimmt sein. Durch diese Art der Variation wird die *lokale* Umgebung von $\boldsymbol{g}$ im Lösungsraum inspiziert.

Nach Abbruch des Verfahrens wird das dann fitteste Genom als Ergebnis ausgegeben. Für die Terminierung des Algorithmus kommen unterschiedliche Abbruchkriterien in Frage. So kann z.B. eine feste Anzahl an Generationen vorgegeben werden („a priori-Kriterium"), was den Vorteil hat, daß die Laufzeit recht genau abgeschätzt werden kann. Für das Erreichen eines guten Optimums kann es hingegen besser sein, eine Wertschranke $S > 0$ vorzugeben, und das Verfahren stoppen zu lassen, sobald ein Genom $\boldsymbol{g}$ mit $f(\boldsymbol{g}) < S$ in der Population auftritt. Ein solches „a posteriori-Kriterium" setzt aber bereits eine bestimmte Kenntnis der Struktur der Zielfunktion voraus, und kann im schlimmsten Fall zur Nichtterminierung führen.

4.2.1.2.2.2 Die eigene Implementierung Für den in dieser Arbeit eingesetzten Genetischen Algorithmus zur Bestimmung von Prädiktoren werden folgende Spezifika festgelegt:

- Es wird ein fester *Prädiktorradius* r für sämtliche Prädiktoren gewählt. In Experimenten hat sich folgende Wahl als akzeptabel erwiesen:

$$\boxed{r_{\mathrm{Pred}} := 5} \qquad \textit{(Prädiktor-Radius)} \tag{4.9}$$

- Die Ausgangspopulation Γ_0 wird zufällig bestimmt, indem für jedes Genom $\boldsymbol{\alpha}^{(0,j)}$ jeder Linearkoeffizient α_i zunächst einen zufälligen Wert zwischen -1 und $+1$ erhält, und danach das Genom normiert wird gemäß Def. 4.1.

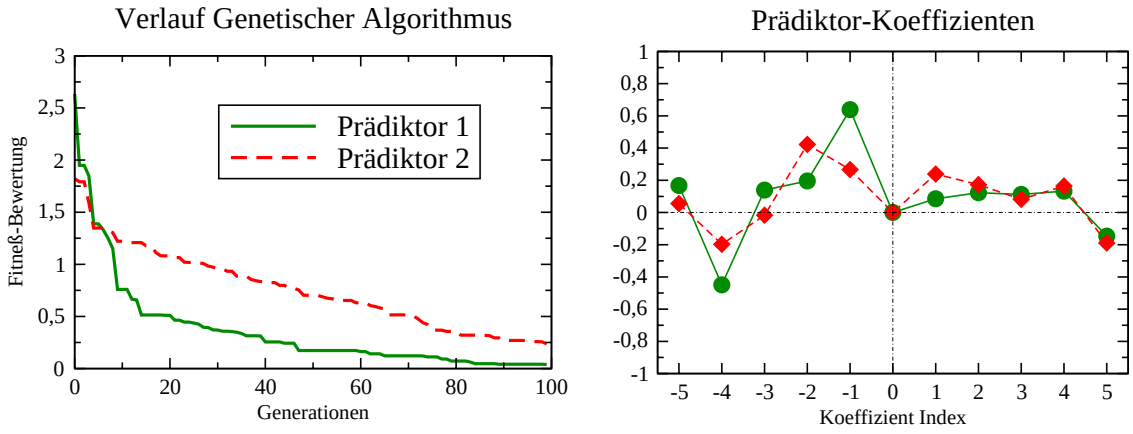

Abbildung 4.7: Prädiktoren: Erzeugung mittels GA (links); Koeffizienten (rechts)

- Der Anteil an *fitten Prädiktoren* in der Population, welche in die nächste Generation übernommen werden, beträgt 25%.
- Aus jedem überlebenden Genom $\boldsymbol{\alpha}$ wird ein neues Genom $\boldsymbol{\alpha}'$ durch *Mutation* der Gene α_i, $-r \leq i \leq r$, erzeugt. Es wird eine *maximale Mutationsrate* μ festgelegt, d.h. zu jedem Linearkoeffizienten α_i entsteht der Wert $\alpha_i' := (1+\gamma) \cdot \alpha_i$ mit einem zufällig gewählten Faktor $\gamma \in [-\mu, +\mu]$.

$$\boxed{\mu_{\text{Mut}} := 0.05} \qquad \textit{(maximale Mutationsrate)} \tag{4.10}$$

- Aus den überlebenden Genomen werden zufällige Paare $(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)})$ gebildet. Durch *Kreuzung* werden Kinderpaare $(\boldsymbol{\alpha}'^{(1)}, \boldsymbol{\alpha}'^{(2)})$ generiert: Für sämtliche Genpositionen i, $-r \leq i \leq r$, findet mit einer gewissen festen Wahrscheinlichkeit p ein Genaustausch statt. Bei einer ungeraden Anzahl von überlebenden Genomen wird das nicht in ein Paar aufgenommene Genom kopiert.

$$\boxed{p_{\text{Cross}} := 1/2} \qquad \textit{(Kreuzungs-Wahrscheinlichkeit)} \tag{4.11}$$

- Die verbleibenden 25% an Genomen für die neue Generation werden zufällig gewählt in der gleichen Weise wie die Ausgangspopulation Γ_0.

Auf die Wahl der Populationsgröße und des Abbruchkriteriums wird im nächsten Abschnitt eingegangen werden.

4.2.1.2.3 Zur Qualität des Prädiktor-Ansatzes Abb. 4.7, links, zeigt den Verlauf zweier Anwendungen des Genetischen Algorithmus aus 4.2.1.2.2.2 auf die jeweils gleiche Trainingsmenge. Die Abszisse repräsentiert die Anzahl an Generationen, auf der Ordinate ist der Fitneßwert hinsichtlich der entropiebasierten Bewertungsfunktion (4.8) aus 4.2.1.2.1 aufgetragen. Es ist der Fitneßwert des jeweils *bestbewerteten* Prädiktors einer Generation dargestellt. Qualitativ erkennt man, daß in beiden Fällen der Prädiktorgenerierung die Bewertung durch die Fitneßfunktion mit zunehmender Generationsanzahl zu geringeren (besseren) Werten führt.

Für die absoluten Fitneßwerte maßgeblich ist die Größe der Quantisierungsstufen zur Berechnung der Fitneßfunktion $f_{(u^{(i)})}(\pi)$, welche wie folgt festgelegt wird:

$$\boxed{Q_{Fit} := 0.01} \qquad \textit{(Quantisierungsstufe Fitneßfunktion)} \tag{4.12}$$

Größere Quantisierungsstufen führen i.d.R. zu niedrigeren Bewertungen, jedoch darf dies nicht unbedingt als Vorteil angesehen werden. Es ergeben sich zwei Probleme: *Erstens* wird es durch größere Intervalle für den Genetischen Algorithmus schwieriger, Detailverbesserungen an einem Prädiktor durchzuführen, denn leichte Änderungen in der Fehlersequenz resultieren dann nicht unbedingt auch in einer Verbesserung des Fitneßwertes. *Zweitens* ist es in Experimenten desöfteren vorgekommen, daß der Fitneßwert 0 aufgetreten ist, was darauf hindeutet, daß dann sämtliche Werte der Prädiktions-Fehlersequenz bereits in der niedrigsten Quantisierungsstufe angesiedelt waren. In diesem Fall kann der Genetische Algorithmus keine Verbesserung mehr erzielen, obwohl es vermutlich noch Verbesserungspotential gäbe. Die Wahl deutlich *kleinerer* Quantisierungsstufen als in (4.12) hat in der Praxis hingegen keine wesentlichen Vorteile gebracht.

Als *Populationsgröße* N wird

$$\boxed{N_{\mathrm{GA}} := 100} \qquad \textit{(GA-Populationsgröße)} \tag{4.13}$$

verwendet. Eine größere Anzahl an Genomen hat in den durchgeführten Tests zu keinen signifikanten Verbesserungen geführt.

Die Anzahl an *Generationen* beträgt in Abb. 4.7 jeweils

$$\boxed{G_{\mathrm{GA}} := 100} \qquad \textit{(GA-Generationsanzahl)} \tag{4.14}$$

Eine größere Anzahl an Iterationen hat in den Experimenten meist nur zu geringfügigen Verbesserungen geführt, die zudem in keinem Verhältnis zum damit verbundenen Berechnungsaufwand standen.

Die Art des Verlaufs der Fitneßverbesserung, wie sie in Abb. 4.7 dargestellt ist, hat sich in weiteren Experimenten in der wesentlichen Form bestätigt. Die Verringerung der Fitneßbewertung beginnt in den Beispielen um den Wert 2 herum. Dabei ist zu beachten, daß wie erwähnt stets der *bestbewertete* Prädiktor einer Generation dargestellt ist. Die *schlechtesten* Bewertungen für zufällig generierte Prädiktoren lagen in den Experimenten fast immer oberhalb des Wertes 5.

Abb. 4.7, rechts, stellt die Koeffizienten der beiden durch den Genetischen Algorithmus entstandenen Ergebnisprädiktoren dar. Das Zentrum der Prädiktoren (im Prinzip die Position, an der das jeweils vorherzusagende Sample liegen würde) wurde durch eine vertikale Linie markiert. Offensichtlich unterscheiden sie beide Prädiktoren voneinander, sie scheinen aber zumindest eine grobe Ähnlichkeit zu besitzen. Wie ein bestmöglicher Prädiktor für die Trainingsmenge des Beispiels auszusehen hätte, läßt sich nach Ansicht des Autors nicht klar erkennen.

In Tab. 4.1 werden den beiden erlernten Prädiktoren π^1 und π^2 die Bewertungen zweier einfacher handkonstruierter Prädiktoren gegenübergestellt. Der *„Rechteck-Prädiktor“* $\pi^{\square}$ besitzt die Koeffizienten

$$\alpha_i^{\square} := \begin{cases} 1/2 & : i \in \{-1, +1\} \\ 0 & : \text{sonst} \end{cases} \qquad \textit{(Rechteck-Prädiktor)}$$

Die ersten beiden Spalten der Tabelle enthalten die Fitneßwerte für zwei der *Trainingsunterschriften*, welche für das genetische Erlernen von π^1 und π^2 verwendet wurden. Man sieht, daß der Rechteck-Prädiktor signifikant schlechtere Bewertungen erhält als die beiden erlernten Prädiktoren. Dies hat sich in anderen Experimenten durchgehend bestätigt. Der zweite betrachtete Prädiktor $\pi^{=}$ ist der in (4.7) auf S. 62 definierte *„Next=This“-Prädiktor*. Die Bewertungen für $\pi^{=}$ liegen durchgehend um den Wert 3 herum, sind also sehr schlecht.

In der dritten und vierten Spalte sind Unterschriften der Person repräsentiert, von der auch die Trainingsunterschriften stammen. Die beiden *Testunterschriften* wurden jedoch *nicht* für die Anwendung des Genetischen Algorithmus verwendet. Die Bewertungen sind ähnlich denjenigen für die Trainingsunterschriften. Dieses Phänomen wurde regelmäßig beobachtet.

	Train 1	*Train 2*	*Test 1*	*Test 2*	*Fremd 1*	*Fremd 2*	*Fremd 3*
Prädiktor π^1	0.026	0.020	0.038	0.008	0.119	0.006	0.034
Prädiktor π^2	0.185	0.240	0.152	0.202	0.379	0.120	0.242
Rechteck $\pi^{\square}$	0.471	0.604	0.527	0.613	0.628	0.446	0.594
Next=This $\pi^{=}$	2.991	3.083	2.936	3.050	2.808	3.031	2.956

Tabelle 4.1: Fitneß-Bewertungen für verschiedene Prädiktoren und Unterschriften

Der ursprüngliche Versuch des Autors, einem möglichen *Overfitting* des Verfahrens mit der in 2.5.4.4 vorgestellten *Validierungsmethode* zu begegnen, ließ sich nicht durchführen, weil der Effekt, daß von einer bestimmten Generation ab der in (2.16), S. 33, definierte Validierungsfehler wieder ansteigt, in den Experimenten praktisch nicht auftrat. Aufgrund der weiter oben gemachten Beobachtung, wonach jenseits von G_{GA}-vielen Generationen keine wesentliche Verbesserung mehr auftritt, wurde daher als *Abbruchkriterium* für den Genetischen Algorithmus die Anzahl G_{GA} an Generationen gemäß (4.14) festgelegt.

Man könnte nach dem eben Diskutierten zu dem Schluß kommen, daß der Genetische Algorithmus zur Prädiktorgenerierung, wie er hier entwickelt wurde, *nicht* zu Overfitting neigt. Dies erscheint dem Autor jedoch wenig plausibel. Betrachten wir die letzten drei Spalten in Tabelle 4.1, in denen die Bewertungen für *Fremdunterschriften* dreier verschiedener Personen angegeben sind. Erstaunlicherweise erhalten die erlernten Prädiktoren auch für diese Unterschriften mit den Trainingsdaten vergleichbare Bewertungen, für Fremdunterschrift Nr. 2 erhalten π^1 und π^2 sogar die jeweils besten für sie gemessenen Fitneßwerte. Auch dieses Phänomen hat sich als recht typisch herausgestellt. Es ergibt sich somit, daß ein durch die angegebene Lernmethode generierter Prädiktor häufig für *alle* Unterschriften eine gute Fitneßbewertung erhält, ohne daß die Form des Prädiktors in offenkundiger Weise „generisch" zu nennen wäre. Der Autor hat dieses Problem bislang nicht verstanden.

Abb. 4.8 stellt zwei *Prädiktions-Fehlersequenzen* gemäß Def. 4.1 einander gegenüber. Es wurden die beiden erlernten Prädiktoren π^1 und π^2 auf jeweils die gleiche Trainingsunterschrift angewandt. Die Anforderung nach *wenigen kräftigen Peaks* erscheint einigermaßen erfüllt zu sein. Desweiteren ist die Struktur beider Fehlersequenzen zumindest als *ähnlich* zu bezeichnen, zumindest die Positionen der großen Peaks stimmen ungefähr überein. Bei genauerer Betrachtung fällt allerdings auf, daß die untere Fehlersequenz in etwa doppelt so große Werte annimmt wie die obere Sequenz, denn die Ordinatenskala unten stellt den Bereich $[0, 0.025]$ dar, im Vergleich zu dem Bereich $[0, 0.125]$ im oberen Datengraphen. In den Experimenten hat der Autor diese Art von *„ungefährer Skalierungsinvarianz"* der Form der Prädiktions-Fehlersequenz gelegentlich in noch viel stärkerem Maße bis hin zu zwei Zehnerpotenzen beobachtet. Auch für dieses Phänomen hat der Autor bislang keine Erklärung finden können. Diese Eigenschaft wird aber dazu beitragen, daß die im nächsten Abschnitt vorgestellte Bestimmungsmethode für Segmentierungsstellen ausreichend stabile Ergebnisse liefern wird.

4.2.1.3 Ermittlung von Segmentierungsstellen

Unser nächstes Ziel ist das Auffinden geeigneter Segmentierungsstellen anhand der Prädiktionsfehlersequenz.

4.2.1.3.1 Bestimmung aller lokaler Maxima In einem ersten Schritt bestimmen wir sämtliche *lokalen Maxima* der Fehlersequenz $(\tilde{u}_t)_{0\le t\le T}$, welche uns als *Kandidaten* für die gesuchten Segmentierungsstellen dienen werden. Eine Stelle t lokalen Maximums ist gegeben genau dann, wenn $\tilde{u}_t \ge \tilde{u}_{t-1}$ und $\tilde{u}_t \ge \tilde{u}_{t+1}$ gilt; am Signalrand $t = 0$ bzw. $t = T$ muß entsprechend nur eine dieser beiden Beziehungen gelten. Gesondert behandeln wollen wir den Fall des Auftretens von *„Höhenplateaus"*, d.h. Intervalle $I = [t_l, t_r] \subseteq [0, T]$ mit der

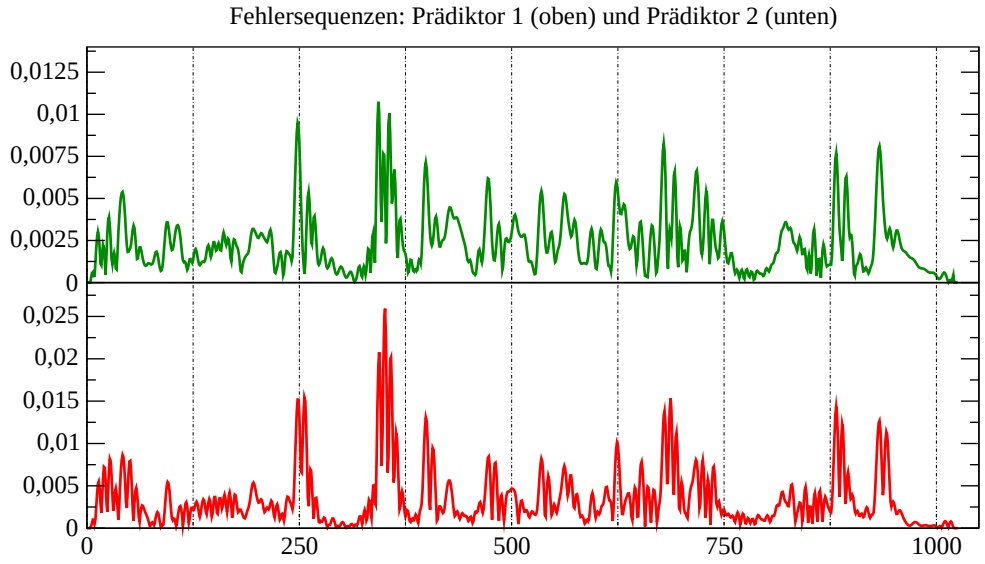

Abbildung 4.8: Prädiktions-Fehlersequenzen zweier Prädiktoren für die gleiche Unterschrift

Eigenschaft $\tilde{u}_t = \tilde{u}_{t'}$ für alle $t, t' \in I$, sowie $\tilde{u}_{t_l} > \tilde{u}_{t_l-1}$, falls $t_l > 0$, und $\tilde{u}_{t_r} > \tilde{u}_{t_r+1}$, falls $t_r < T$. Für ein solches Intervall lokaler Maxima wählen wir die Mittelstelle $\bar{t} := \lceil (t_l + t_r)/2 \rceil$ als repräsentierendes lokales Maximum aus. Für sehr breite Intervalle dieser Art könnte die Wahl des Mittelindexes vielleicht als nicht sehr plausibel für unseren angestrebten Zweck erscheinen. Es hat sich in der Praxis aber gezeigt, daß solche Höhenplateaus nicht sehr häufig auftreten, und daß sie zumeist in dem im folgenden Abschnitt vorgestellten *Pruning*-Schritt aussortiert werden.

Obige Vorgaben lassen sich nun in naheliegender Weise dergestalt umsetzen, daß jede Stelle t der Fehlersequenz genau einmal inspiziert werden muß („Scanline-Prinzip“). Der Rechenaufwand beträgt somit $\mathrm{O}(T)$.

Abb. 4.9 zeigt im mittleren Graphen die Fehlersequenz eines zuvor generierten Prädiktors für die im oberen Graphen dargestellte Unterschriftensequenz. Im unteren Graphen sind mit gepunkteten senkrechten Linien die Stellen aller lokaler Maxima kenntlich gemacht. Neben den Stellen großer, und damit für unsere Zwecke prinzipiell in Frage kommender Maxima wurden auch sehr viele kleine Maxima entdeckt, die z.T. aus nach der Datenvorverarbeitung gemäß 4.1 verbliebenem Signalrauschen resultieren. Aber auch viele der Maxima, die wohl dem Nutzsignal entspringen, sind für uns untauglich, nämlich dann, wenn sie in der Nähe eines sehr viel größeren Maximums liegen.

4.2.1.3.2 Pruning der lokalen Maxima Die folgende Unterauswahl an lokalen Maxima legt die Menge der Segmentierungsstellen fest. Sei M die Menge aller lokaler Maxima und S die Ergebnismenge der Segmentstellen. Zu Beginn des Verfahrens gilt $S = \emptyset$. Wir betrachten nun für jede Stelle $t^* \in M$ die lokale Umgebung $U_r(t^*) := [t^* - r, t^* + r]$ in der Fehlersequenz mit Radius r (auf die Wahl von r werden wir im nachfolgenden Abschnitt eingehen). Stelle t^* wird genau dann in S aufgenommen, wenn es kein $t \in U_r(t^*)$ gibt, dessen Prädiktionsfehler $\tilde{u}_t$ echt größer ist als $\tilde{u}_{t^*}$. Eine direkte Umsetzung dieses Verfahrens besitzt eine Laufzeit von $\mathrm{O}(r \cdot |M|) = \mathrm{O}(r \cdot T)$.

4.2.1.3.3 Wahl der Segmentgröße Die zu erwartende Segmentgröße hängt wesentlich von der Wahl des Radius r aus 4.2.1.3.2 ab. Wir sind in dieser Wahl nicht völlig frei: Sehr kurze Segmente besitzen möglicherweise eine zu unspezifische Struktur und passen dann mit sehr vielen Abschnitten einer zweiten Unterschrift überein. Sehr lange Segmente sind hingegen evtl. überspezifisch, sodaß selbst mit an sich passenden Segmenten einer zweiten

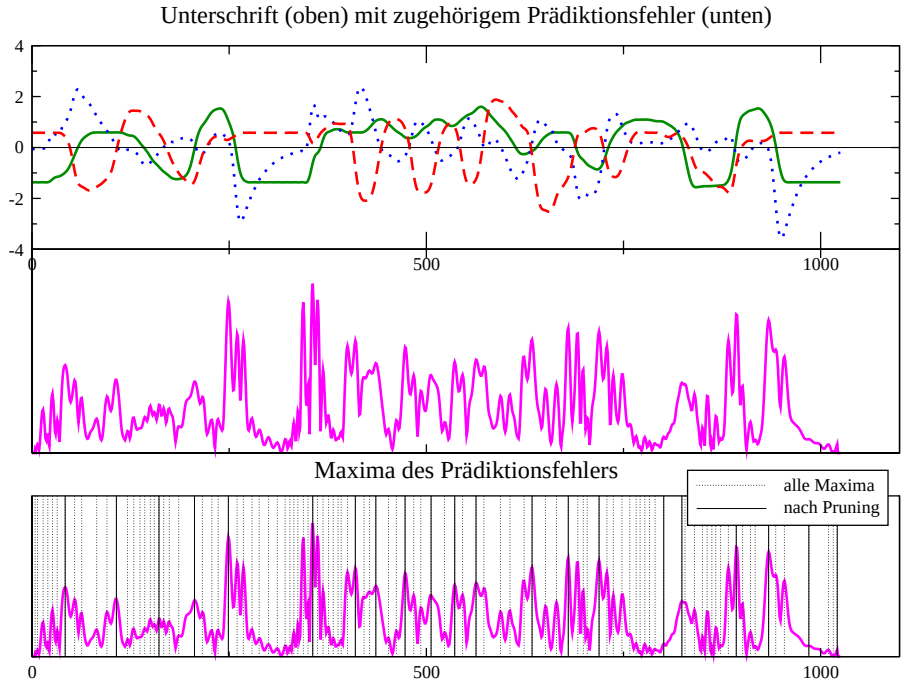

Abbildung 4.9: Lokale Maxima der Prädiktionsfehlersequenz

Unterschrift kein befriedigender Vergleich gelingt. Eigentlich läßt sich eine geeignete Wahl des Pruningradius r erst dadurch bestimmen, daß man mit dem *fertigen* System für unterschiedliche Radien Tests durchführt. Einige dem Autor vernünftig erscheinende Kriterien zur Wahl von r lassen sich aber dennoch an dieser Stelle angeben. Als erstes ist die *minimale Strukturbreite* des Nutzsignals zu nennen. Bei der Betrachtung der Frequenzgangeigenschaften der verwendeten Daten in Kapitel 3.2 war deutlich geworden, daß die höchsten vorkommenden Frequenzen unter dem 0.1-fachen der Samplingfrequenz liegen, und somit die kleinsten informationstragenden Strukturen aus mindestens 10 Zeitsamples bestehen müssen. Eine geringere Segmentbreite scheint mithin nicht sinnvoll zu sein.

Als weiteres Kriterium kann die *augenscheinliche* Qualität des Alignments zweier zuvor segmentierter Unterschriften der gleichen Person dienen, wie es in 4.2.2 praktiziert werden wird. Der Autor hat eine Reihe von diesbezüglichen Experimenten durchgeführt und festgestellt, daß für Radien r unterhalb der Länge 15 und oberhalb von 40 die Alignmentergebnisse merklich an Qualität verlieren; als seines Erachtens brauchbarster Wert hat sich

$$\boxed{r^*_{\mathrm{Seg}} := 20} \qquad \textit{(Basisradius Segmentierung)} \tag{4.15}$$

herausgestellt.

Die Größe r^*_{Seg} wird in diesem System allerdings nicht direkt als Radius zur Segmentierung eingesetzt. Stattdessen wird zunächst der Mittelwert $\bar{T} := \langle |u^{(i)}| \rangle$ der *Längen* sämtlicher Trainingsunterschriften gebildet, und der tatsächlich eingesetzte Radius r bestimmt sich dann für eine Unterschrift u der Länge T über die Beziehung

$$r := \left[(T/\bar{T}) \cdot r^*_{\mathrm{Seg}} \right] \qquad \textit{(Pruning-Radius)} \tag{4.16}$$

Die Idee ist, daß für zwei Unterschriften der gleichen Person, welche sich in ihrer physikalischen Länge deutlich voneinander unterscheiden, auch die enthaltenen Segmente, welche nach unserer Grundannahme weitgehend ähnlich sein sollten, ungefähr im entsprechenden Längenverhältnis zueinander auftreten. Der zugehörige Ausgleichsfaktor ist in (4.16) durch '$T/\bar{T}$' gegeben. In Abb. 4.9 sind die nach dem Pruning verbleibenden lokalen Maxima durch durchgezogene Linien markiert.

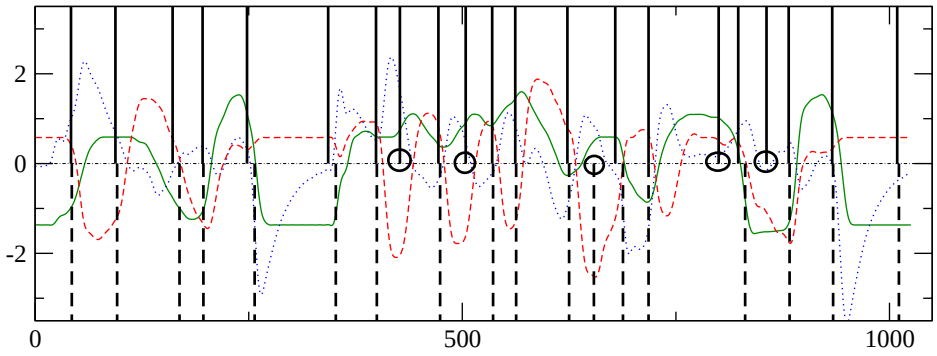

Abbildung 4.10: Segmentierung der gleichen Unterschrift mit zwei Prädiktoren

4.2.1.3.4 Zur Stabilität der Segmentierungsstellen Abb. 4.10 zeigt zwei Folgen von Segmentierungsstellen für die gleiche Unterschrift, dargestellt durch durchgezogene (oben) und gestrichelte (unten) vertikale Linien. Die Segmentierungen basieren auf den beiden in Abb. 4.8 gezeigten Prädiktions-Fehlersequenzen. Man erkennt zweierlei: *Erstens* gibt es mehrere Segmentierungsstellen in einer der beiden Segmentierungen, zu welchen keine Segmentierungsstelle der anderen Segmentierung in räumlicher Nähe liegt (kreisförmig markiert). *Zweitens*: In allen anderen Fällen passen die gefundenen Stellen beider Segmentierungen recht genau zueinander. Dabei ist eine perfekte Übereinstimmung gar nicht angestrebt, wir werden nämlich in 4.2.2 nicht die vollständigen Segmentsequenzen, sondern lediglich bestimmte statistische Merkmale derselben für einen Vergleich zweier Unterschriften heranziehen. Dies wird eine gewisse Toleranz gegenüber der genauen Position von Segmentierungsstellen erlauben, und die in Abb. 4.10 auftretenden Abweichungen erscheinen diesbzgl. ausreichend klein. Die Stabilität des hier gezeigten Beispiels hat sich in anderen Fällen bestätigt. Für das erstgenannte Problem der *fehlenden* Segmentierungsstellen, was auf fehlende Segmente in der entstehenden Segmentsequenz hinausläuft, werden wir an späterer Stelle eine Lösung anzugeben in der Lage sein.

4.2.2 Alignment von Segment-Sequenzen

Mit den in 4.2.1 eingeführten Mitteln wurde die Voraussetzung dafür geschaffen, jede der für das Enrollment eingesetzten Trainingsunterschriften in eine Folge von Segmenten zu zerlegen. Abb. 4.9 ließ zumindest erahnen, daß sich die Segmentierung damit in natürlich wirkender Weise durchführen läßt. Als nächstes wird es darum gehen, die *zueinander passenden* Segmente *zweier* Trainingsunterschriften zu identifizieren, denn unsere Grundannahme bestand darin, daß zwei Unterschriften der gleichen Person aus *ähnlichen* Sequenzen von Segmenten bestehen.

Abb. 4.11 stellt zwei verschiedene Unterschriften der gleichen Person einander gegenüber. Die vertikalen Linien stehen für Segmentierungsstellen, welche in beiden Fällen auf der Grundlage des gleichen Prädiktors berechnet wurden. Die Anzahl der Segmente unterscheidet sich, sodaß eine vollständige paarweise Zuordnung von vornherein ausgeschlossen ist. Ein dem Autor sinnvoll erscheinendes Alignment ist durch die Pfeile in der Bildmitte dargestellt. Man erkennt hier, daß bis zum 12. Segment im Großen und Ganzen Übereinstimmung herrscht. Das 13. Segment der unteren Sequenz findet jedoch keinen Partner, was bei genauerer Betrachtung auch nachvollziehbar ist: Die untere Sequenz besitzt in ihrem Mittelteil *vier* Wellen der y-Komponente, die obere Sequenz hingegen nur *drei*. Ob man eher Segment 16 oder 17 der oberen Sequenz dem 17. Segment der unteren Sequenz zuordnen möchte ist diskutierbar.

Die Abbildung gibt einen guten Hinweis darauf, wie ähnlich im Groben, wie unterschiedlich aber im Detail zwei Unterschriften der gleichen Person sein können. Das hier definierte automatisierte Alignment wird daher nicht die vollständigen Segmente der Segmentsequenzen

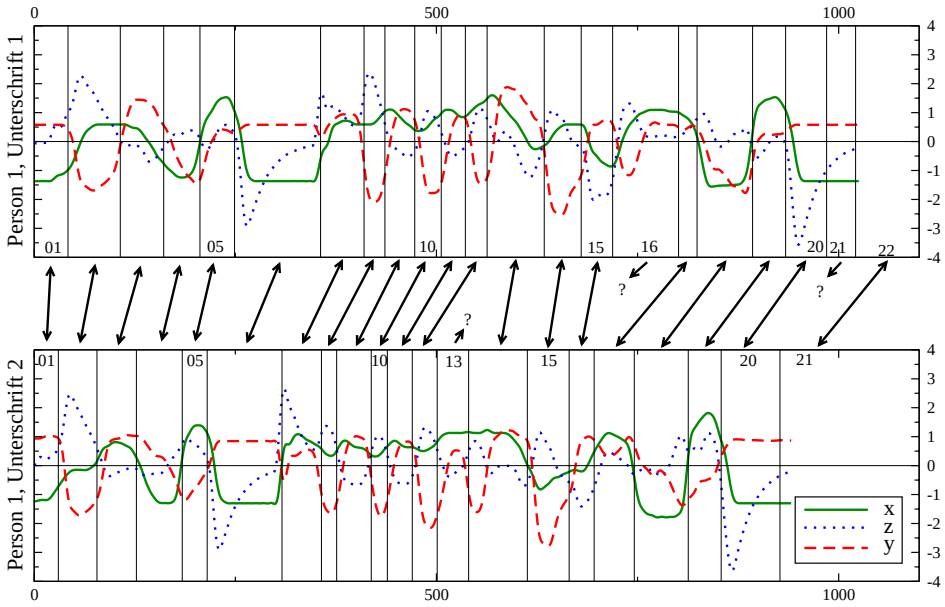

Abbildung 4.11: Segment-Zuordnung für zwei Unterschriften der gleichen Person

miteinander vergleichen, denn diese können sich wie gesehen erheblich voneinander unterscheiden. Entscheidend wird daher die Auswahl geeigneter *Features* im Sinne von 2.5.3 sein, anhand derer sich das „wesentlich Gemeinsame“ zweier ähnlicher Segmente erkennen läßt.

4.2.2.1 Definition der Alignment-Features

Es geht darum, die ein Segment darstellende Teil-Zeitreihe $(u_t)_{l\le t\le r}$, mit den Segmentierungsstellen l und r, durch einen Featurevektor dergestalt zu codieren, daß „inhaltlich zueinandergehörige“ Segmente aus zwei Unterschriften der gleichen Person eine möglichst hohe Ähnlichkeit haben, während dies für nicht zueinanderpassende Segmente nicht zutreffen soll. Aufgrund des Fehlens einer Theorie über Aufbau und Eigenschaften von dynamisch erfaßten Unterschriften wählen wir hier ein sehr unspezifisches *statistisches* Featureset $\boldsymbol{f}$.

Für die j-te Komponente $(u_{j,t})_{l\le t\le r}$ eines Segmentes definieren wir die beiden Features $f_{j,1}$ und $f_{j,2}$ durch

$$\begin{aligned} f_{j,1} &:= \langle u_{j,t}\rangle_{l\le t\le r} \\ f_{j,2} &:= 2\cdot\sqrt{\langle (u_{j,t}-\langle u_{j,t}\rangle_{l\le t\le r})^2\rangle_{l\le t\le r}} - 1 \end{aligned} \qquad \textit{(statistische Features)} \qquad (4.17)$$

Die Features $f_{j,1}$ stellen also den *Mittelwert* der Samples der j-ten Segmentkomponente dar, die Features $f_{j,2}$ im wesentlichen deren *Standardabweichung*. Die lineare Transformation der Form ‘$2x-1$’ bei $f_{j,2}$ hat den Zweck, einen für sämtliche Features $f_{j,\lambda}$ *ungefähr* vergleichbaren Wertebereich zu schaffen. Für den Mittelwert liegt dieser Wertebereich aufgrund der Zentrierung und Amplitudennormierung gemäß 4.1.5 in den meisten Fällen zwischen -1 und $+1$ („Ausreißer“ treten gelegentlich auf). Die herkömmliche Standardabweichung ist jedoch immer nichtnegativ, und sie nimmt für Unterschriftensegmente nach Beobachtungen des Autors zumeist Werte zwischen 0 und 0.5 an.

Die so definierten Features besitzen folgende wichtigen Eigenschaften:

- *Wechselseitige Unabhängigkeit der Features:* Im Idealfall sollte jeder mögliche Mittelwert μ_j einer Segmentkomponente $(u_{j,t})_l^r$ mit jeder beliebigen Streuung σ_j einhergehen können. Beispielsweise kann ein Mittelwert $\mu_j = 0$ sowohl durch ein Segment mit konstanten Werten (DC-Anteil) als auch durch ein Segment mit hoher Dynamik entstehen. Andererseits kann der Signalverlauf eines Segmentes entlang der Ordinate verschoben

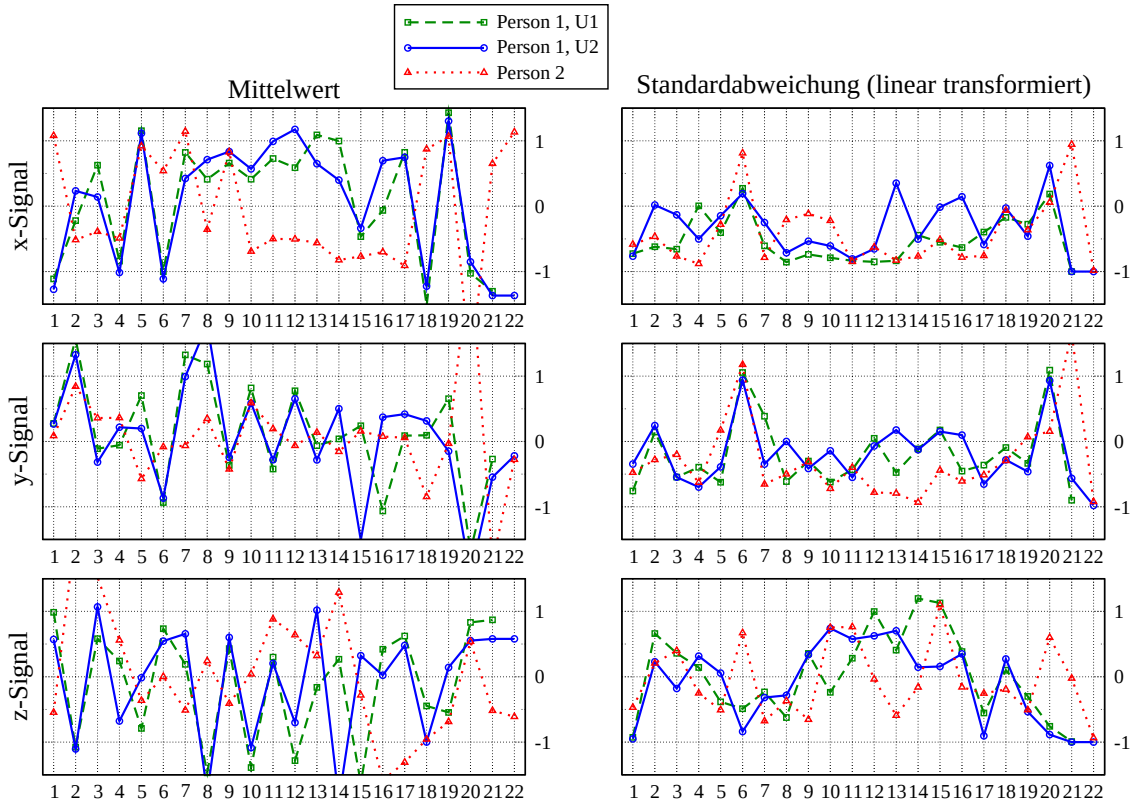

Abbildung 4.12: Segmentcodierungs-Sequenzen, aufgeteilt nach Features

sein und damit bei gleichbleibender Standardabweichung unterschiedliche Mittelwerte annehmen. Auch die verschiedenen Segment-*Komponenten* $(u_{j,t})_l^r$, $1 \leq j \leq 3$, sollten in ihren statistischen Momenten weitgehend unabhängig voneinander sein, denn die Drucksensoren des BiSP-Pens stehen *senkrecht* zueinander. Genaugenommen müßte dies aber noch eingehender untersucht werden.

- *Gleichmäßiger Beitrag des gesamten Segmentes:* Die Samples der Segmentfolge $(u_{j,t})_l^r$ gehen allesamt mit gleichem Gewicht in die Berechnung eines Featurewertes ein. Dies bringt eine gewisse Robustheit gegenüber leichten lokalen Verzerrungen und noch vorhandenen Störeinflüssen im Segment mit sich.

Anschaulich beschreiben die definierten Features grob das Gebiet mit Mittelpunkt und Ausdehnung in dem von den Wertebereichen des x-, y- und z-Sensors aufgespannten Raum, welches von den Samples einer Segment-Zeitreihe $(u_t)_l^r$ ausgefüllt wird. Zwei Segmente sind intuitiv somit dann ähnlich, wenn sie ähnlich geformte „Sample-Cluster“ an ähnlichen Positionen in diesem Raum bilden. Die zeitliche Information innerhalb des Segmentes wird bei diesem Vergleich allerdings nicht mehr beachtet.

Abb. 4.12 stellt die Segmentfolgen zweier Unterschriften der gleichen Person (blaue durchgezogene und grüne gestrichelte Linien) einander gegenüber. Zum Vergleich ist außerdem eine beliebige Fremdunterschrift (rot gepunktet) dargestellt. Alle drei Unterschriften wurden mit dem gleichen Prädiktor segmentiert. Auf der Abszisse sind die Segmente der Unterschriften abgezählt, die Ordinate gibt den Wertebereich von ungefähr $[-1, +1]$ wieder. Genaugenommen dürfte es sich nur um *Punktfolgen* handeln, die Verbindungslinien zwischen den Punkten dienen lediglich der Veranschaulichung. Die linken Schaubilder repräsentieren die *Mittelwert-Features* $f_{j,1}$, rechts sind entsprechend die *Streuungs-Features* $f_{j,2}$ zu betrachten. In allen sechs Schaubildern kann man sehen, daß der Bereich $[-1, +1]$ von den Features recht gut ausgenutzt wird. Vergleicht man je zwei Featurefolgen der gleichen Unterschrift (verschiede-

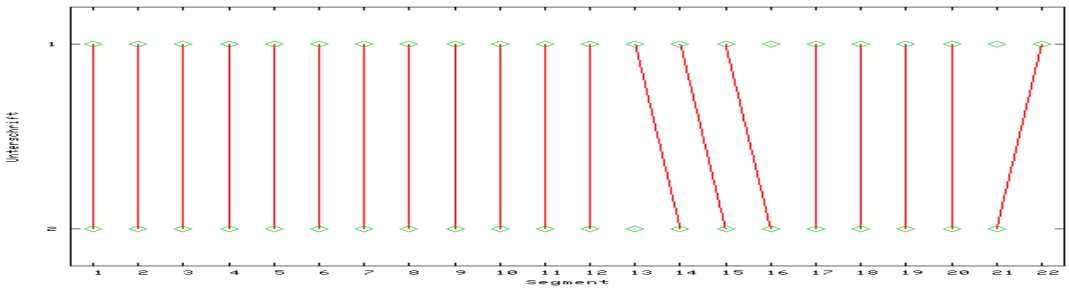

Abbildung 4.13: Alignment der Segmentsequenzen aus Abb. 4.11

ne Schaubilder), so kann man feststellen, daß sich deren Verläufe zumeist deutlich voneinander unterscheiden, was als Indiz für die zuvor postulierte Unabhängigkeit der Features angesehen werden kann. Die Featurefolgen der beiden Unterschriften der gleichen Person besitzen *tendenziell* einen ähnlichen Verlauf, wobei diese Aussage für die Mittelwert-Features in stärkerem Maße zu gelten scheint als für die Streuungs-Features. Die Unterschiede sind jedoch teilweise recht groß, sodaß wir uns an dieser Stelle des Textes noch kein Urteil über die allgemeine Güte der definierten Features erlauben können.

4.2.2.2 Sequence-Alignment mittels Dynamic-Time-Warping (DTW)

Auf der Grundlage der segmentweisen Feature-Extraktion wird nun zum Sequence-Alignment der *DTW-Algorithmus* aus 2.5.5.2 verwendet. Als *Einzelfehler* $\epsilon(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$ zwischen zwei Segmentcodierungen $\boldsymbol{x}^{(1)}$ und $\boldsymbol{x}^{(2)}$ (vgl. (2.19), S. 38) wird die Metrik $\mathrm{d}_{\|\cdot\|_2}$ der in 2.5.5.1 angegebenen *Euklidischen Norm* (2.18) eingesetzt.

Den *Gap-Fehler* $\epsilon_\perp(\boldsymbol{x})$ für Gap-Paare $(\boldsymbol{x}, \perp)$ bzw. $(\perp, \boldsymbol{x})$ mit einem Featurevektor $\boldsymbol{x}$ werden wir aus den Gapfehlern $\epsilon_\perp(x_j)$ für jedes einzelne Feature f_j zusammensetzen:

$$\epsilon_\perp(\,(x_1, \ldots, x_n)^\top\,) := \|(\epsilon_\perp(x_1), \ldots, \epsilon_\perp(x_n))^\top\|_2$$

Zur Wahl der *Komponenten-Gap-Fehler* $\epsilon_\perp(x_j)$ stellen wir folgende Überlegungen an: *Erstens:* Da wir die faktischen Wertebereiche der einzelnen Features ungefähr angeglichen hatten (vgl. die Diskussion zur Featurewahl in 4.2.2.1), erscheint es gerechtfertigt, allen Komponenten-Gap-Fehlern den *gleichen* Wert zuzuordnen. *Zweitens:* Der Komponenten-Gap-Fehler sollte als *„faire Bestrafung“* angesehen werden. Dies bedeutet, daß er zum einen *nicht zu groß* gewählt werden sollte, da sich ansonsten möglicherweise ein Segmentcode $\boldsymbol{x}^{(1)}$, welcher in der zweiten Unterschrift eigentlich keinen Partner besitzen sollte, dennoch ein unpassendes Partnersegment $\boldsymbol{x}^{(2)}$ sucht, um der höheren Bestrafung durch einen Gap-Fehler zu entgehen. Zum anderen sollte der Gap-Fehler aber auch *nicht zu klein* gewählt sein, um keinen Anreiz dafür zu bieten, sich zu einem Gap-Paar anstatt mit einem eigentlich passenden Segment $\boldsymbol{x}^{(2)}$ zu verpaaren. Bei einer Reihe von Experimenten hat sich herausgestellt, daß, bezogen auf den ungefähren Wertebereich $[-1, +1]$ der verwendeten Features, der Komponenten-Gap-Fehler

$$\boxed{\varepsilon_{\mathrm{Align},\perp} := 2/3} \qquad \textit{(Komponenten-Gap-Fehler)} \tag{4.18}$$

zu brauchbaren Resultaten führt.

Abb. 4.13 stellt das durch den hier vorgestellten Ansatz erzeugte Alignment für die beiden Segmentsequenzen aus Abb. 4.11, S. 72, schematisch als Graph dar. Jeder Knoten steht für ein Segment, wobei die obere Reihe die Segmente der oberen Sequenz aus Abb. 4.11 repräsentiert, die untere Reihe entsprechend die untere Unterschrift. Jede Kante steht für eine Zuordnung zwischen zwei Segmenten, wie sie in Abb. 4.11 durch Pfeile dargestellt waren. Beim Vergleich beider Abbildungen wird man feststellen, daß die Alignments übereinstimmen. Tatsächlich wurde das *„dem Autor sinnvoll erscheinende Alignment“* aus Abb. 4.11 nicht von diesem selbst manuell, sondern automatisch mit dem hier vorgestellten Verfahren erzeugt.

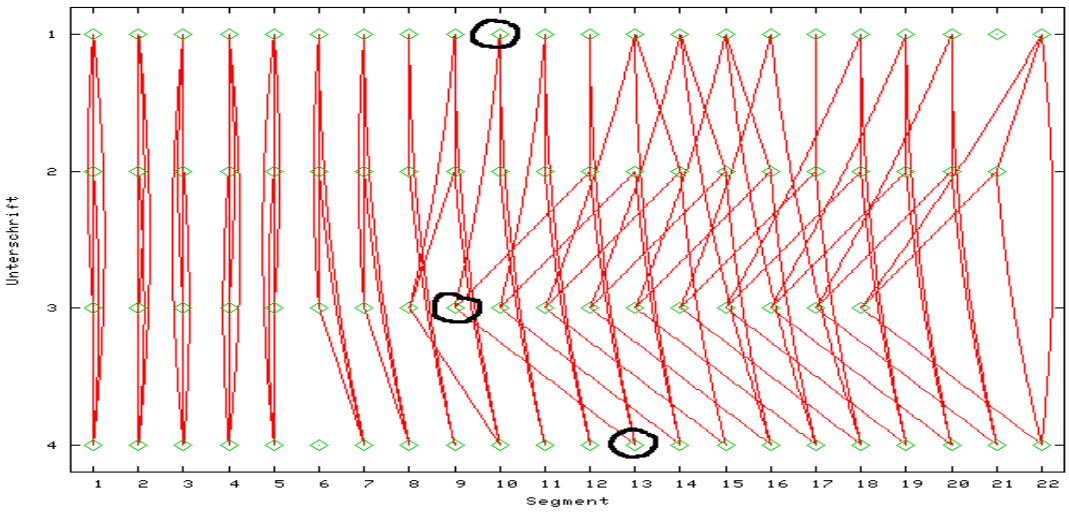

Abbildung 4.14: Paarweises Alignment für vier Unterschriften der gleichen Person

4.2.3 Generierung einer Modellsequenz

In diesem Abschnitt wird es darum gehen, aus der Menge der Trainingsunterschriften ein adäquates Modell für die Segmentfolge von Unterschriften einer Person zu bilden, welche zum einen gewissermaßen die *„wahre" Segmentfolge* repräsentiert, darüberhinaus aber auch die *personenspezifische Variabilität* in jedem Abschnitt der Unterschrift nachstellt. Wir werden hier, ähnlich wie im Grundlagenabschnitt 2.5.4 zur Modellierung, Unterschriften u als Zeitsequenzen $(\boldsymbol{x}_l)$ von *Featurevektoren* darstellen, wobei hier allerdings die Vektoren $\boldsymbol{x}_l$ als Codierungen von *Segmenten* und nicht von einzelnen Unterschriften-Samples zu deuten sind. Per Konvention werden wir von einem „Segment" $\boldsymbol{x}_l$ sprechen, wenn eigentlich die *Codierung* des entsprechenden Segmentes gemeint ist.

4.2.3.1 Aufbau des Alignment-Graphen

In 4.2.2 wurde ein Verfahren für das Alignment *zweier* Segmentsequenzen vorgestellt. Wir berechnen damit nun für je zwei zuvor segmentierte Trainingsunterschriften $u^{(i)}$ und $u^{(k)}$ das Alignment $A_{i,k}$, wodurch ein *„Alignment-Graph"* ähnlich dem in Abb. 4.14 entsteht (die Interpretation des Schaubildes folgt der von Abb. 4.13). Für ein Alignment $A_{i,k}$ zwischen Segmentsequenzen $(\boldsymbol{x}_l^{(i)})_{0\le l\le L_i}$ und $(\boldsymbol{x}_l^{(k)})_{0\le l\le L_k}$ hatte der DTW-Algorithmus aus 2.5.5.2.2 die Laufzeit $\mathrm{O}(L_i \cdot L_k)$. Da $\binom{m}{2} = \frac{1}{2}m(m-1) = \Theta(m^2)$-viele solcher Alignments durchzuführen sind, erhalten wir mit $L := \max_{1\le i\le m}\{L_i\}$ einen Gesamtrechenaufwand von $\mathrm{O}(m^2 \cdot L^2)$.

Nach unseren Beobachtungen in 4.2.2.2 dürfen wir hoffen, daß jedes einzelne dieser Alignments $A_{i,k}$ im Großen und Ganzen die inhaltlich zueinandergehörenden Segmente der beiden Unterschriften wiedergibt. Abb. 4.14 bestärkt uns in dieser Hoffnung teilweise: Am linken und rechten Ende des Graphen werden jeweils die vier Segmente mit gleicher relativer Position in ihrer jeweiligen Sequenz zu einer *Zusammenhangskomponente* (zwischen je zwei Knoten existiert ein Weg) zusammengefaßt. Als ideal wird man es bezeichnen, wenn *Transitivität* in solchen Zusammenhangskomponenten gilt, d.h. wenn mit drei Segmentsequenzen $(\boldsymbol{x}_l^{(i)})_{l\le L_i}$, $(\boldsymbol{x}_l^{(j)})_{l\le L_j}$ und $(\boldsymbol{x}_l^{(k)})_{l\le L_k}$ sowie Segmentzuordnungen $(\boldsymbol{x}_\lambda^{(i)}, \boldsymbol{x}_\mu^{(j)})$ und $(\boldsymbol{x}_\mu^{(j)}, \boldsymbol{x}_\nu^{(k)})$ auch die Zuordnung $(\boldsymbol{x}_\lambda^{(i)}, \boldsymbol{x}_\nu^{(k)})$ existiert. Sofern diese Voraussetzung für einen Alignment-Graphen stets erfüllt ist, macht es Sinn, die Zusammenhangskomponenten als Ausgangspunkt für die Segmente der Modellsequenz zu verwenden. In diesem Fall werden nämlich durchgehend zueinander passende Segmente der Trainingsunterschriften durch eine solche Zusammenhangskomponente repräsentiert. In unserem Beispiel wird die Eigenschaft der Transitivität jedoch

gelegentlich verletzt, wie die drei markierten, offenbar zur gleichen Zusammenhangskomponente gehörenden Segmente zeigen. Das Problem ist, daß dadurch in diesem Beispiel eine Zusammenhangskomponente entsteht, welche *mehrere Segmente der gleichen Unterschrift* enthält (die Segmente Nr. 10 und 12 von Unterschrift Nr. 2), was unserem Vorhaben zuwider läuft.

Für Alignment-Graphen mit mehr als vier Trainingsunterschriften hat sich gezeigt, daß die Transitivität häufig nicht gilt, und in einem Fall mit neun Unterschriften ist dem Autor eine Zusammenhangskomponente begegnet, welche 190 der insgesamt 192 im Graph vorhandenen Segmente enthielt. Wir werden also den Alignmentgraph zuerst in einer Weise zerlegen müssen, daß die entstehenden *Partitionen* unserem Ideal transitiver Zusammenhangskomponenten zumindest hinreichend nahe kommen.

4.2.3.2 Partitionierung des Graphen

Wir gehen aus vom Alignmentgraph $G = (S, A)$ mit der Menge S der *Segmente* $\boldsymbol{x}_l^{(i)}$ der Trainingsunterschriften $u^{(i)}$ als Knoten, und der Menge A sämtlicher *Alignmentzuordnungen* $(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(k)})$ zwischen den Segmenten zweier Unterschriften $u^{(i)}$ und $u^{(k)}$ als Kanten. Es erscheint plausibel, daß jede Trainingsunterschrift *höchstens eines* ihrer Segmente zu einem Segment der zu konstruierenden Modellsequenz beiträgt. Wir definieren daher eine *Partition* als eine Teilmenge $P \subseteq S$, für die für je zwei Segmente $\boldsymbol{x}$ und $\boldsymbol{x}'$ aus Unterschriften $u^{(i)}$ und $u^{(k)}$ mit $\boldsymbol{x} \neq \boldsymbol{x}'$ auch $i \neq k$ gilt. Unter dem *Grad* $\mathrm{grad}(P)$ einer Partition P wollen wir die Anzahl an Kanten $(\boldsymbol{x}, \boldsymbol{x}') \in A$ mit $\boldsymbol{x} \in P$ und $\boldsymbol{x}' \in P$ verstehen. Da transitive Zusammenhangskomponenten sich durch einen *maximalen* Verknüpfungsgrad auszeichnen (eine transitive Zusammenhangskomponente bestehend aus k Knoten besitzt $\binom{k}{2} = k(k-1)/2$ Kanten), werden wir innerhalb von Zusammenhangskomponenten des Alignmentgraphen nach Partitionen mit *maximalem Grad* suchen.

Algorithmus 4.2 Partitionierung eines Alignmentgraphs $G = (S, A)$

```
bestimme alle Zusammenhangskomponenten G_i := (S_i, A_i), 1 ≤ i ≤ N, in G
for all G_i do
  bestimme eine Partition P_i ⊆ S_i mit maximalem Grad
  bilde den neuen Graph G'_i := (S_i \ P_i, A_i^{P_i}), mit A_i^{P_i} := {(x, x') ∈ A_i | x ∉ P_i ∧ x' ∉ P_i}
  if G'_i ≠ ∅ then
    rufe den Algorithmus rekursiv für G'_i auf
  end if
end for
gib Partitionen P_1, ..., P_N aus
```

Eine Grobfassung unserer Vorgehensweise wird in Algorithmus 4.2 angeben. Die Bestimmung aller *Zusammenhangskomponenten* kann durch sukzessive Anwendung von *Breitensuche (BFS)* erfolgen, wobei besuchte Knoten des Graphen zu markieren sind, und als Ausgangspunkt jeder Anwendung der BFS stets ein bislang unmarkierter Knoten verwendet wird. Der Gesamtzeitaufwand hierfür beträgt $O(|S| + |A|)$ [OW93]. Das Auffinden einer *Partition mit maximalem Grad* geschieht durch aufeinanderfolgendes Generieren aller möglichen Partitionen der jeweiligen Zusammenhangskomponente G_i. Für jede erzeugte Partition P wird $\mathrm{grad}(P)$ bestimmt und mit dem bisherigen Maximum verglichen. Tatsächlich kann man sich auf solche Partitionen beschränken, in denen sämtliche Unterschriften, welche in G_i repräsentiert sind, durch Segmente vertreten sind. Zu jeder Partition P, die diese Bedingung *nicht* erfüllt, gibt es nämlich eine Partition P' mit $P \subseteq P'$, welche Segmente aus allen repräsentierten Unterschriften enthält. Damit gilt dann aber $\mathrm{grad}(P) \leq \mathrm{grad}(P')$, womit die Betrachtung von P überflüssig ist. Die Maximumsbestimmung wird als *Backtracking-Verfahren* [Eng93] realisiert.

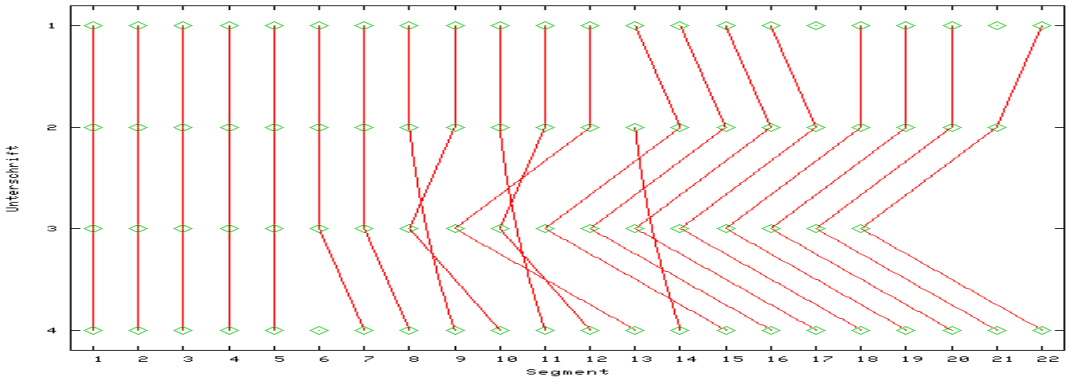

Abbildung 4.15: Partitionierung des Alignmentgraphen aus Abb. 4.14

Sei nun G_i die aktuelle Zusammenhangskomponente, und seien $\{\boldsymbol{x}_{l_1}^{(k)}, \ldots, \boldsymbol{x}_{l_r}^{(k)}\}$, $r \geq 0$, die in G_i enthaltenen Segmente der Unterschrift $u^{(k)}$. Wir geben eine Zahl $\beta \in \mathbb{N}$ vor, und beschränken die Suche nach der Partition P^* maximalen Grads, indem wir lediglich die Segmente $\boldsymbol{x}_{l_1}^{(k)}, \ldots, \boldsymbol{x}_{l_\rho}^{(k)}$, mit $\rho := \min\{\beta, r\}$, als mögliche Kandidaten für P^* betrachten.

Wir haben die Suche auf eine konstante Anzahl β von Segmenten pro Unterschrift (*„Analysebreite“*) beschränkt, um das Verfahren nicht zu ineffizient werden zu lassen. Selbst mit dieser Beschränkung β ist im Worstcase bei m Trainingsunterschriften noch eine Laufzeit von $\Omega(\beta^m)$ für die einmalige Anwendung des Algorithmus möglich. Wir werden daher bemüht sein, einen möglichst kleinen Wert für β zu bestimmen. In den vom Autor durchgeführten Experimenten haben sich ab dem Wert $\beta = 5$ keine Änderungen am Ergebnis der Partitionierung mehr ergeben. Für die Werte 4 und 3 ergaben sich in erster Linie Unterschiede hinsichtlich *kleiner* Partitionen, die aber mit dem in 4.2.3.3 vorzustellenden *Pruningverfahren* zumeist ohnehin entfernt werden dürften. Dennoch wird, um das Leistungspotential des Systems untersuchen zu können, als Analysebreite β in dieser Arbeit der Wert

$$\boxed{\beta_{\text{Part}} := 5} \qquad \textit{(Analysebreite)} \tag{4.19}$$

verwendet. In Zukunft muß nach einer effizienten Heuristik zur Bestimmung von Partitionen mit ausreichend hohem Grad gesucht werden.

Abb. 4.15 stellt das Ergebnis der Partitionierung des in Abb. 4.14 vorgestellten Alignmentgraphen vor. In dieser neuen Form der Darstellung, *„Partitionsgraph“* genannt, befinden sich zwei Segmente in einer gemeinsamen Partition genau dann, wenn es einen Weg zwischen den die beiden Segmente repräsentierenden Knoten gibt. Die meisten entstandenen Partitionen enthalten Segmente aus allen vier Trainingsunterschriften, und auch die Zuordnungen untereinander entsprechen per Auge betrachtet recht genau dem, was man als Mensch bei Ansehung des zugehörigen Alignmentgraphen erwarten würde.

Als *quantitatives* Beurteilungswerkzeug läßt sich die *Verteilung der Grade* der Partitionen heranziehen. Für die in Abb. 4.16 gezeigte Partitionierung eines Alignmentgraphen mit *neun* Trainingsunterschriften beträgt der höchstmögliche Grad einer Partition $\binom{9}{2}$. Im Beispiel gibt es 35 Partitionen. Davon besitzen 11 einen Grad $\geq \binom{8}{2}$, 17 einen Grad $\geq \binom{7}{2}$, und für 19 ist der Grad $\geq \binom{6}{2}$. Diese 19 Partitionen besitzen jeweils sieben von neun oder mehr Segmente, scheinen also als Kandidaten zur Bestimmung eines Segments der angestrebten Modellsequenz grundsätzlich in Frage zu kommen. Übrigens liegt der Partitionierung aus Abb. 4.16 derjenige Graph zugrunde, von dem weiter oben berichtet wurde, daß nahezu sämtliche Knoten in der gleichen Zusammenhangskomponente liegen. Der Algorithmus könnte somit den an ihn gestellten Anforderungen genügen.

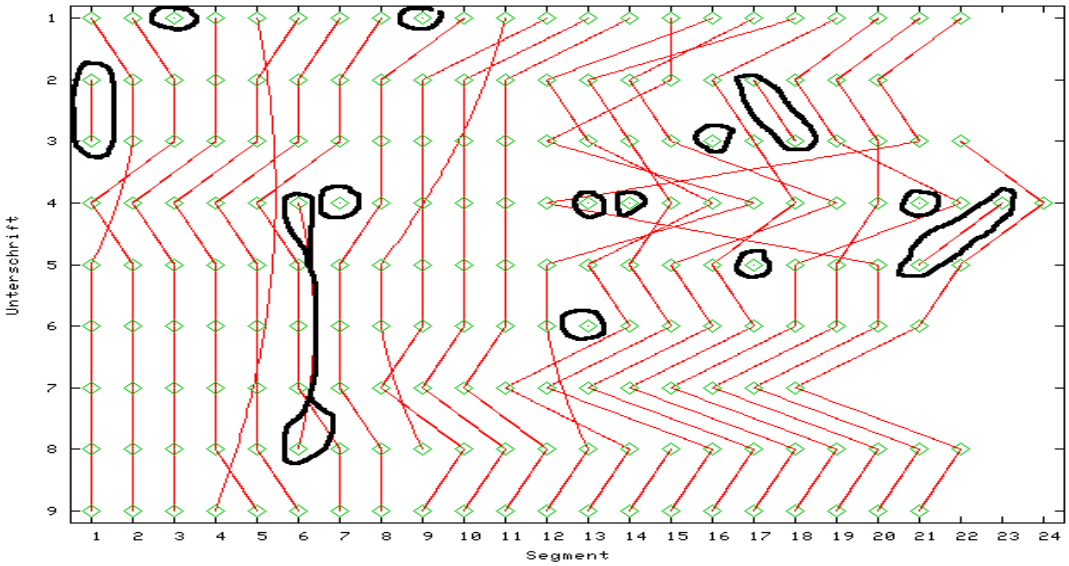

Abbildung 4.16: Partitionierung für 9 Unterschriften mit zu löschenden Partitionen

4.2.3.3 Pruning der Partitionsmenge

Abb. 4.16 zeigt eine Partitionierung ähnlich derjenigen in Abb. 4.15 für neun Trainingsunterschriften. Unser Segmentierungsalgorithmus aus 4.2.1 war daraufhin angelegt gewesen, daß auch bei zwei sehr unterschiedlich langen Unterschriften der gleichen Person eine ähnliche Anzahl an Segmenten entsteht. In der gezeigten Abbildung erkennt man, daß dieses Ziel nicht vollständig erreicht wird, denn die längste Unterschrift besteht aus 24, die kürzeste aus 18 Segmenten, wobei die meisten Unterschriften allerdings 21 oder 22 Segmente besitzen. Unsere Modellsequenz soll die „wahre“ Segmentsequenz einer Person repräsentieren, und dies hat dann konsequenterweise auch für die „wahre“ Segment*anzahl* zu gelten. Die Anzahl der in 4.2.3.2 erzeugten Partitionen beläuft sich in Abb. 4.16 auf 35, aber diese deutlich zu hohe Zahl kommt in der Hauptsache durch das Auftreten sehr *kleiner* Partitionen (im Schaubild markiert) zustande. Für eine adäquate statistische Repräsentation eines Segmentes benötigt man möglichst umfangreiche Partitionen, und tatsächlich gibt es in unserem Beispiel 19 Partitionen mit mindestens sieben von neun möglichen Segmenten, was der *mittleren Segmentanzahl* aller Trainingsunterschriften nahe kommt.

Motiviert durch diese Betrachtungen wählen wir nun die zur Modellierung verwendeten Partitionen wie folgt:

1. Sortiere sämtliche N Partitionen nach ihrer Größe.

2. Wähle die μ größten Partitionen aus, wobei μ den *Median* der Längen aller Trainingsunterschriften bezeichne.

3. Falls unter den *gewählten* Partitionen solche sind, deren Größe einen bestimmten prozentualen Anteil $\tau_\perp$ der Anzahl m an Trainingsunterschriften *unterschreitet*, so entferne diese aus der Zielmenge.

4. Falls unter den *nicht gewählten* Partitionen solche sind, deren Größe einen bestimmten prozentualen Anteil $\tau_\top$ der Anzahl m an Trainingsunterschriften *überschreitet*, so füge diese in die Zielmenge ein.

Die Laufzeit dieses Verfahrens wird durch den Sortierschritt 1 dominiert und beträgt daher $\mathrm{O}(N \cdot \log N)$ für die Anzahl N aller Ausgangspartitionen.

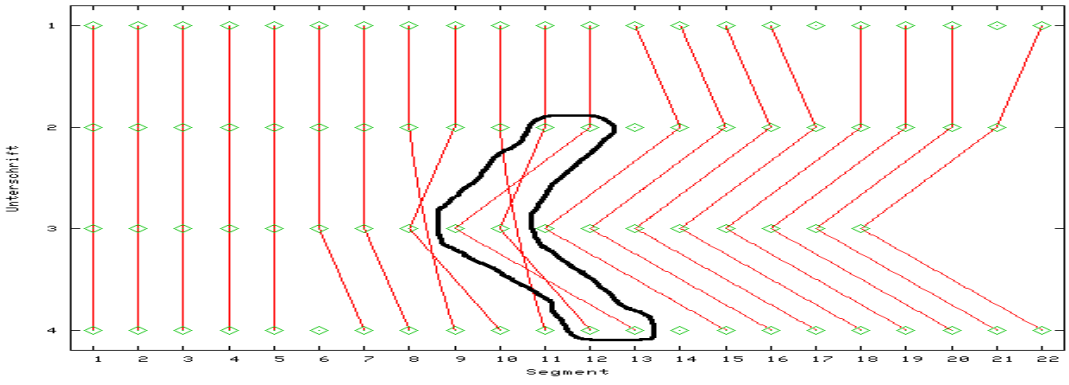

Abbildung 4.17: Partitionierung mit zeitlich sich überkreuzenden Partitionen

Wir verwenden den *Median* anstelle des arithmetischen Mittelwertes, da dieser unempfindlich ist gegenüber einzelnen Segmentsequenzen extremer Länge, und weil der Median tatsächlich als Länge mindestens einer der Trainingsunterschriften auftritt. Bei den beiden Sonderregelungen am Ende des Algorithmus handelt es sich um Maßnahmen, die verhindern sollen, daß sehr gute Partitionen kein Segment repräsentieren dürfen, bzw. sehr schlechte Partitionen dieses tun dürfen. Der Autor verwendet hierzu folgende Werte:

$$\boxed{\tau_{\text{Prune},\perp} := 1/2} \qquad \textit{(unterer Pruning-Threshold)} \tag{4.20a}$$

$$\boxed{\tau_{\text{Prune},\top} := 3/4} \qquad \textit{(oberer Pruning-Threshold)} \tag{4.20b}$$

In dieser Form haben sich in allen näher untersuchten Fällen akzeptable Anzahlen an Partitionen ausreichender Größe ergeben.

4.2.3.4 Sequenzialisierung der Partitionen

Wir besitzen nach dem Bisherigen eine *Menge* von Partitionen geeigneter Kardinalität, die sich zur Repräsentation von Segmenten der Unterschriften einer Person verwenden lassen. Wir wissen allerdings noch nicht, *welche* Segmente dadurch repräsentiert werden, d.h. wir müssen noch die *Reihenfolge* der Partitionen in der Modell*sequenz* festlegen. Abb. 4.17 gibt den bereits in Abb. 4.15 für vier Unterschriften dargestellten Alignmentgraph nach Anwendung des Pruningschrittes aus 4.2.3.3 wieder. In den meisten Fällen des Beispiels macht es keine Schwierigkeit zu sagen, welche von zwei Partitionen P_1 und P_2 zeitlich vor der anderen liegen soll, denn fast immer gilt: Die Segmente von P_1 liegen allesamt zeitlich entweder vor oder hinter den zur *gleichen Unterschrift* gehörenden Segmenten von P_2. Im Fall der beiden markierten Partitionen kommt es allerdings zu zeitlichen Überschneidungen, d.h. die eben genannte Eigenschaft gilt nicht mehr durchgehend. Allerdings liegt eine der beiden Partitionen für immerhin 3 von 4 Segmenten vor der anderen, sodaß man sich auch hier noch auf eine Reihenfolge verständigen können wird.

Wir wollen eine '$<$'-Relation auf der Menge der Partitionen definieren, für die wir hoffen dürfen, daß sie in der Praxis hinreichend häufig die „wahre" Reihenfolge der Partitionen widerspiegelt. Wir führen dazu zunächst einige Konventionen ein. Sei $\boldsymbol{x}_l^{(i)}$ das l-te Segment der Unterschrift $u^{(i)}$. Wir schreiben '$u^{(i)} \in P_j$' („P_j repräsentiert $u^{(i)}$"), falls es ein l gibt mit $\boldsymbol{x}_l^{(i)} \in P_j$. Falls nun $u^{(i)} \in P_j$, dann bezeichne 'P_{ji}' den Index l des dann eindeutig bestimmten Segmentes $\boldsymbol{x}_l^{(i)} \in P_j$. Es werden folgende Gewichte definiert:

- '$g_c(P_j, P_k)$': Für jedes Paar (P_j, P_k) von Partitionen zählen wir mit '$g_c(P_j, P_k)$' die *Anzahl* an Unterschriften mit *späteren* Segmenten in P_k als in P_j:

$$g_c(P_j, P_k) := \left| \left\{ i \mid u^{(i)} \in P_j \cap P_k \wedge P_{ji} < P_{ki} \right\} \right|$$

- '$g_d(P_j, P_k)$': Das Gewicht '$g_d(P_j, P_k)$' ist ähnlich definiert wie '$g_c(P_j, P_k)$', nur daß hier zusätzlich die *zeitlichen Abstände* zwischen zwei Segmenten berücksichtigt werden:

$$g_d(P_j, P_k) := \sum_{i:u^{(i)} \in P_j \cap P_k \wedge P_{ji} < P_{ki}} P_{ki} - P_{ji} + 1$$

- '$g_c(P_k)$': Für jede Partition P_k ist '$g_c(P_k)$' die *Gesamtanzahl* aller in Partitionen auftretender zeitlich voranliegender Segmente:

$$g_c(P_k) := \sum\nolimits_j g_c(P_j, P_k)$$

- '$g_d(P_k)$': Das Gewicht '$g_d(P_k)$' ist das Analogon zu '$g_c(P_k)$' für '$g_d(P_j, P_k)$':

$$g_d(P_k) := \sum\nolimits_j g_d(P_j, P_k)$$

Die beiden *Paargewichte* '$g_c(P_j, P_k)$' und '$g_d(P_j, P_k)$' sollen für je zwei Partitionen den Grad des „Davorliegens" von P_j zu P_k ausdrücken: Wie in obigem Beispiel gesehen wollen wir mit '$g_c(P_j, P_k)$' sagen, daß P_j vor P_k liegt, wenn es mehr Unterschriften gibt, deren Segmente in P_j zeitlich vor denen in P_k liegen als umgekehrt. Mit '$g_d(P_j, P_k)$' betrachten wir auch noch den zeitlichen Abstand der jeweiligen Segmente: Ein sehr weiter Abstand zwischen zwei Segmenten spricht für ein sehr deutliches Davorliegen. Jedoch hat sich gezeigt, daß dieses Kriterium mit Vorsicht anzuwenden ist: Es kommt vor, daß eine Partition in fast allen Unterschriften *knapp vor* einer zweiten Partition, in wenigen Unterschriften aber *deutlich dahinter* liegt. In diesem Fall werden wir eher dazu neigen, der reinen Anzahl in Form des Gewichtes '$g_c(P_j, P_k)$' den Vorrang zu geben. Die akkumulierten, aus den Paargewichten abgeleiteten Partitions-Gewichte '$g_c(P_k)$' und '$g_d(P_k)$' lassen sich als Maß für den Grad des allgemeinen „Vorneliegens" in der Menge der Partitionen auffassen: Ein niedriger Wert bedeutet, daß es nur wenige andere Partitionen mit früheren Segmenten als in dieser Partition gibt. Die Benutzung dieser Gewichte eignet sich dann, wenn für zwei Partitionen im direkten Vergleich nicht klar entschieden werden kann, welche von beiden zeitlich vor der anderen liegt.

Nach diesen Ausführungen definieren wir die angekündigte '<'-Relation. Die folgenden Bedingungen sind dabei wie folgt anzuwenden: Für je zwei Partitionen P_j und P_k wird eine Bedingung dann und nur dann geprüft, wenn durch die vorangegangene Bedingung nicht bereits eine Anordnung zwischen beiden Partitionen festgelegt wurde.

1. $P_j < P_k$, falls $g_c(P_k, P_j) < g_c(P_j, P_k)$
2. $P_j < P_k$, falls $g_d(P_k, P_j) < g_d(P_j, P_k)$
3. $P_j < P_k$, falls $g_c(P_j) < g_c(P_k)$
4. $P_j < P_k$, falls $g_d(P_j) < g_d(P_k)$

Offenbar wird durch '<' eine *(irreflexive) partielle Ordnung* auf der Menge der Partitionen definiert, denn deren Eigenschaften Transitivität und Asymmetrie übertragen sich direkt aus denen der '<'-Relation auf Gewichten (Zahlen). Eine *partielle Ordnung* läßt sich nun über eine *topologische Sortierung* zu einer *totalen* Ordnung ergänzen.

Definition 4.3 (Topologische Sortierung). *Sei durch '$\sqsubset$' eine partielle Ordnung auf einer endlichen Menge $M := \{x_1, \ldots, x_n\}$ gegeben. Eine Permutation $\pi \in S_n$ heißt* topologische Sortierung *für $(M, \sqsubset)$, wenn mit $x_j \sqsubset x_k$ stets $\pi(j) < \pi(k)$ gilt.*

Zur algorithmischen Bestimmung einer topologischen Sortierung existieren Verfahren mit Laufzeit $O(|M| + |E|)$, wobei $E := \{(x, y) \in M^2 \mid x \sqsubset y\}$ [OW93].

Ein Maß für die Qualität der Ergebnisse des Algorithmus anzugeben erscheint schwierig. Der Autor hat daher eine Reihe von Sequenzialisierungen durchgeführt und sie mit den zugehörigen Partitionsgraphen, für die Abb. 4.17 ein Beispiel darstellt, visuell verglichen. Er hat kein Gegenbeispiel für den Eindruck gefunden, daß die Ergebnisse mit denen, wie sie ein Mensch erzeugen würde, i.w. vergleichbar sind.

4.2.3.5 Berechnung der Modellsequenz

Durch die Partitionierung des Alignmentgraphen in 4.2.3.2 bestehen – im Idealfall – die einzelnen Partitionen aus den jeweils inhaltlich zusammenpassenden Segmenten der Trainingsunterschriften. Wir hatten in 4.2.3.3 u.a. dafür gesorgt, daß, eine ausreichende Anzahl an Trainingsunterschriften vorausgesetzt, die Anzahl an Segmenten in den Partitionen groß ist. Damit sind wir in der Lage, anhand der in einer Partition enthaltenen Featurevektoren statistische Aussagen über das entsprechende Segment der Unterschrift einer Person zu machen.

Wir orientieren uns inhaltlich an der Argumentation und Vorgehensweise des Modellierungsbeispiels aus 2.5.4.2. Es sei $(P_1, \ldots, P_N)$ die Folge der Partitionen $P_k = \{\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_{\nu_k}^k\}$, wie sie durch die *Partitionierung* (4.2.3.2), das *Pruning* (4.2.3.3) und die *Sequenzialisierung* (4.2.3.4) aus dem in 4.2.3.1 erzeugten Alignmentgraph gewonnen wurde. Wir bilden die *Modell-Mittelsequenz* $(\boldsymbol{\mu}_k)_{1\le k\le N}$ durch

$$\boldsymbol{\mu}_k := \langle \boldsymbol{x}_l^k \rangle_{1\le l\le \nu_k} \qquad \textit{(k-tes Folgenglied der Modell-Mittelsequenz)} \tag{4.21a}$$

Die Folge $(\boldsymbol{\sigma}_k)_{1\le k\le N}$ der *segmentspezifischen Variabilitäten* mit $\boldsymbol{\sigma}_k := (\sigma_{1,k}, \ldots, \sigma_{n,k})^\top$ definieren wir über

$$\sigma_{j,k} := \sqrt{\langle (x_{j,l}^k - \mu_{j,k})^2 \rangle_{1\le l\le \nu_k}} \qquad \textit{(segmentspezifische Variabilität)} \tag{4.21b}$$

Damit ist unsere Modellierung der Unterschrift einer Person abgeschlossen.

Anschaulich kann man sich ein Modell als einen zeitlichen „Schlauch“ vorstellen, mit der Mittelwertsequenz $(\boldsymbol{\mu}_k)$ als Mittellinie, und mit variabel verlaufender Formgebung, definiert durch die Folge $(\boldsymbol{\sigma}_k)$. Die Eigenunterschriften der modellierten Person sollten über weite Strecken im *Innern* dieses Schlauches verlaufen. Abb. 4.18 gibt ein Beispiel einer Modellsequenz für 9 Trainingsunterschriften an, dargestellt für alle sechs statistischen Features $f_{j,\lambda}$ aus (4.17), S. 72. Die Abszisse gibt jeweils die Anzahl an Segmenten an, die Ordinate die vom jeweiligen Mittelwert-Feature angenommenen Werte. Die Fehlerbalken geben die jeweilige *Standardabweichung* an.

Regelmäßig wurde für Partitionen P_k und Features f_j beobachtet, daß für einige wenige Segmente $\boldsymbol{x}^k \in P_k$ der Abstand $|x_j^k - \mu_{j,k}|$ unverhältnismäßig groß ist („statistische Ausreißer“). Aus diesem Grund werden die Segmente mit den größten Abweichungen in einem Feature f_j zum Mittelwert $\mu_{j,k}$ in positiver und negativer Ordinatenrichtung aus den Berechnungen gemäß (4.21a) und (4.21b) ausgenommen. Welche der Segmente einer Partition dabei ignoriert werden, wird für jedes Feature getrennt entschieden. Der genaue Anteil pro Richtung (positive und negative Richtung) an ignorierten Segmenten beträgt:

$$\boxed{\alpha_{\text{Ignore}} := 0.1} \qquad \textit{(Modellierungs-Ignoranz)} \tag{4.22}$$

An der *Mittelsequenz* ändert diese Operation wenig: Wie man sieht, verdeckt die durchgezogene, blaue Linie (Mittelsequenz *mit* Ignorierung von Ausreißern) nahezu vollständig die gestrichelte grüne Linie (*ohne* Ignorierung). Die Auswirkungen dieser leichten Modifikation des oben angegebenen Verfahrens sind aber deutlich für die *Standardabweichungen*: Der „Schlauch“ wird insgesamt dünner, vor allem an solchen Stellen, wo er zuvor sehr breit war.

Die für uns wichtigste Erkenntnis aus Abb. 4.18 dürfte sein, daß die Modellsequenz eine deutlich ausgeprägte Struktur besitzt, und nicht etwa zu einer horizontalen Linie um den Nullwert degeneriert ist. Abb. 4.18 läßt sich grob mit Abb. 4.12 auf S. 73 vergleichen, da es sich in beiden Fällen um die gleiche Person handelt. Die wesentliche Struktur der dort gezeigten Einzelunterschriften-Codierungen stimmt recht gut mit der hier dargestellten überein; ein weiteres Indiz dafür, daß unsere Modellierung, insbesondere der Partitionierungsschritt aus 4.2.3.2, wirklich die passenden Segmente der Trainingsunterschriften einander zugeordnet hat. Der Autor weiß ferner zu berichten, daß mehrere Modellierungen der gleichen Person

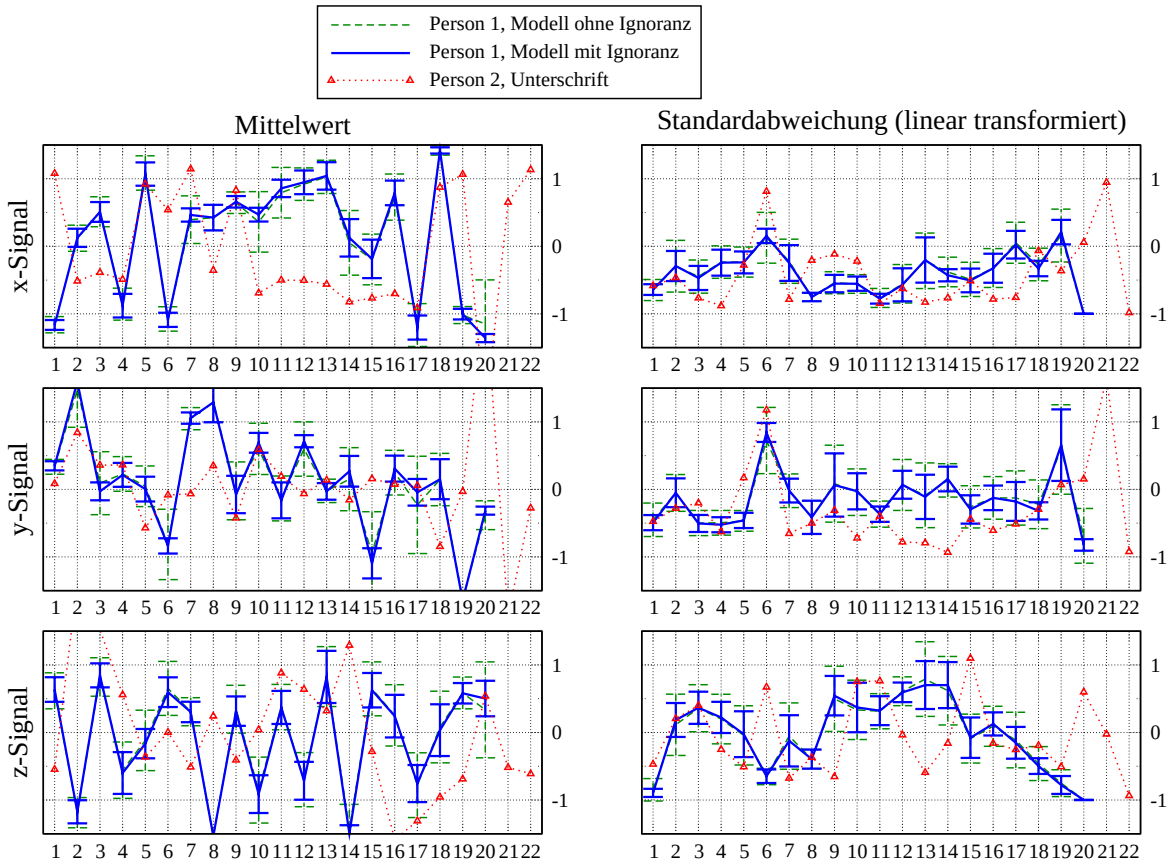

Abbildung 4.18: Modellsequenz, aufgeteilt nach Features

mit *unterschiedlichen* Trainingsunterschriften zu sehr ähnlichen Ergebnissen führen.[5] Für eine *Fremdunterschrift*, wie das rotgepunktete aus Abb. 4.12 übernommene Beispiel, sollte es somit schwierig werden, eine hohe Ähnlichkeit zur Mittelunterschrift zu erlangen.

4.2.4 Zusammenfassung des Enrollmentprozesses

Abb. 4.19 faßt die einzelnen Schritte des Enrollments zusammen. Eingabe ist eine Menge von unverarbeiteten Unterschriftenproben der gleichen Person, ausgegeben wird ein Modell für die Unterschrift dieser Person (s.u.). Sämtliche Trainingsunterschriften werden *vorverarbeitet* gemäß 4.1. Auf den so erhaltenen Sequenzen basierend wird mittels eines Genetischen Algorithmus ein *linearer Prädiktor* generiert (4.2.1.2). Der so entstandene Prädiktor wird daraufhin zur *Segmentierung* sämtlicher Trainingsunterschriften eingesetzt, wobei als Segmentierungspunkte Zeitstellen mit besonders *großen Prädiktionsfehlern* gewählt werden (4.2.1.3). Für die erhaltenen Segmente der Trainingsunterschriften wird eine Featureextraktion für *statistische Features* durchgeführt (4.2.2.1). Für die so codierten Trainingsunterschriften wird paarweise ein *Sequence-Alignment* gemäß 4.2.2.2 vollzogen, was im Ergebnis zu dem *Alignment-Graphen* aus 4.2.3.1 führt. Dieser Graph wird in *Partitionen* zerlegt (4.2.3.2), welche maximal *einen* Segment-Featurevektor jeder Trainingsunterschrift enthalten dürfen, und sich ansonsten dadurch auszeichnen, daß die repräsentierten Segmente einen möglichst hohen Verknüpfungsgrad hinsichtlich des vorangegangenen Alignments besitzen. In einem anschließenden *Pruning-Schritt* (4.2.3.3) wird die Partitionsmenge derart reduziert, daß zum einen die Anzahl an Partitionen in etwa der durchschnittlichen Anzahl an Segmenten in den Trainingsunterschriften entspricht, zum anderen durch eine Partition eine für statistische

[5] Dies gilt allerdings nur, wenn in allen Fällen stets der *gleiche Prädiktor* gemäß 4.2.1.2 verwendet wird. Wir werden auf diese Problematik in Kapitel 6 noch einmal zu sprechen kommen.

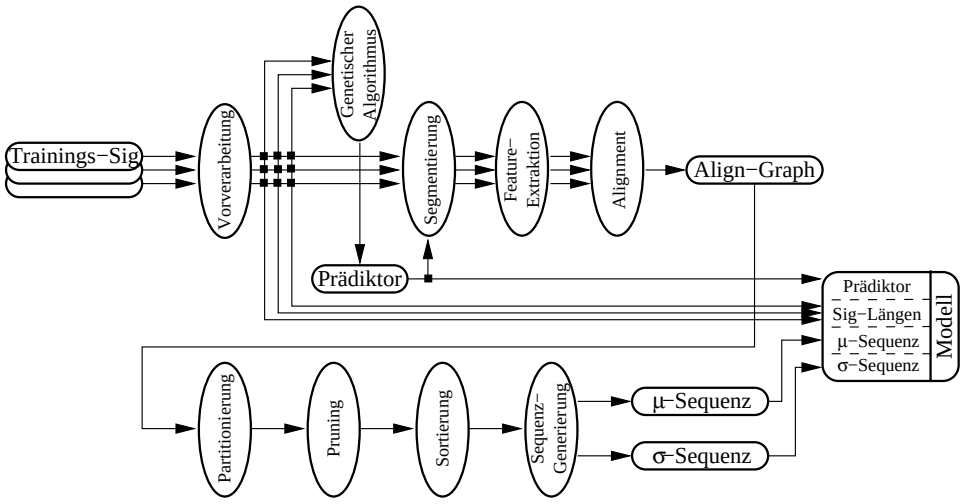

Abbildung 4.19: Ablauf des Enrollmentprozesses

Zwecke hinreichend große Anzahl an Unterschriften repräsentiert wird. Für die verbleibenden Partitionen wird nun eine *Anordnung* mittels topologischer Sortierung bestimmt, welche die Reihenfolge der sie in den Trainingsunterschriften repräsentierenden Segmente möglichst adäquat wiedergeben soll (4.2.3.4). Abschließend werden in 4.2.3.5 die Mittelwerte und Standardabweichungen der in den Partitionen enthaltenen Segmentcodierungen berechnet, was im Ergebnis zu einer *Modellsequenz* für die Unterschriften der zu modellierenden Person führt.

Die folgende Liste faßt sämtliche Informationen zusammen, welche in einer Modellstruktur abgespeichert werden:

- Menge der *physikalischen Längen* der vorverarbeiteten Trainingsunterschriften. Die Menge liegt sortiert vor gemäß der Reihenfolge, in der Trainings- und Update-Unterschriften (vgl. 4.3) in das Modell eingefügt wurden.
- Für die Trainingsunterschriften generierter *Prädiktor* (4.2.1.2).
- Menge der *Segmentcodierungs-Sequenzen* (4.2.2.1) aller Trainingsunterschriften. Die Menge liegt sortiert vor gemäß der Einfügungsreihenfolge.
- Sequenz der *Partitionen des Alignmentgraphen* (4.2.3.2). Die Partitionen können kompakt repräsentiert werden als Mengen von *Referenzen* auf den jeweiligen Featurevektor in der Menge der Segmentcodierungs-Sequenzen.

Es sei betont, daß in der hier angegebenen Modellstruktur nicht die ursprünglichen Unterschriftensequenzen enthalten sind. Die stattdessen vorhandenen Daten, maßgeblich die Folgen von Segmentcodierungen gemäß 4.2.2.1, haben zum einen ein erheblich geringeres Speicherplatzvolumen, zum anderen dürfte es mit diesen Daten nur schwer möglich sein, auf die originalen Drucksequenzen der modellierten Person zurückzuschließen, was dem Datenschutz dienlich ist.

Die für das Enrollment verwendeten Systemparameter wurden wie folgt festgelegt:

- Prädiktor-Radius ((4.9) 4.2.1.2.2.2): $r_{\mathrm{Pred}} = 5$
- Quantisierungsstufe Prädiktor-Fitneßfunktion ((4.12) 4.2.1.2.3): $Q_{Fit} = 0.01$
- Populationsgröße Genetischer Algorithmus ((4.13) 4.2.1.2.3): $N_{\mathrm{GA}} = 100$
- GA-Generationsanzahl ((4.14) 4.2.1.2.3): $G_{\mathrm{GA}} = 100$
- maximale GA-Mutationsrate ((4.10) 4.2.1.2.2.2): $\mu_{\mathrm{Mut}} = 0.05$

- GA-Kreuzungswahrscheinlichkeit ((4.11) 4.2.1.2.2.2): $p_{\text{Cross}} = 0.5$
- Pruning-Basisradius für Segmentierung ((4.15) 4.2.1.3.3): $r^*_{\text{Seg}} = 20$
- Komponenten-Gap-Fehler Alignment ((4.18) 4.2.2.2): $\varepsilon_{\text{Align},\perp} = 0.67$
- Analysebreite der Partitionierung ((4.19) 4.2.3.2): $\beta_{\text{Part}} = 5$
- unterer Pruning-Threshold ((4.20a) 4.2.3.3): $\tau_{\text{Prune},\perp} = 0.5$
- oberer Pruning-Threshold ((4.20b) 4.2.3.3): $\tau_{\text{Prune},\top} = 0.75$
- Modellierungs-Ignoranz ((4.22) 4.2.3.5): $\alpha_{\text{Ignore}} = 0.1$

4.3 Modell-Update

Bei der Modellierung der Unterschrift einer Person stoßen wir auf zwei praktische Probleme:

- Die Anzahl an Unterschriften, die man einer Person beim Enrollment abverlangen kann, wird in der Regel gering ausfallen. Für eine *statistische* Modellierung, wie sie in dieser Arbeit erfolgt, ist jedoch eine größere Menge von Trainingsunterschriften erforderlich.
- Die Unterschrift einer Person kann sich im Laufe der Jahre ändern, sodaß es beim Vergleich neuerer Eigenunterschriften gegen ein vor langer Zeit generiertes Modell evtl. zu einer hohen Falschablehnungsrate kommt.

Dem Problem der *geringen Anzahl* an Trainingsdaten kann dadurch begegnet werden, daß im Enrollmentprozeß zunächst ein *Ausgangsmodell* aus wenigen Unterschriften generiert wird, welches dann im Laufe der Zeit durch Einpflegen weiterer Einzelunterschriften sukzessive verfeinert wird. Diese zusätzlichen Unterschriften lassen sich z.B. durch explizites Erbitten von der Person in regelmäßigen Abständen gewinnen, oder automatisch durch Verwendung von Unterschriften mit besonders hoher Verifikationsbewertung. Das Problem der *Aktualität* eines Modells läßt sich ebenfalls durch Einbinden neuerer Unterschriften, hier allerdings bei gleichzeitiger Entfernung der ältesten vorhandenen Trainingsdaten (FIFO-Prinzip) bewerkstelligen. Beide Ansätze können in Kombination verwendet werden, indem bis zu einer Mindestanzahl von Trainingsdaten zunächst die erste, danach dann nur noch die zweite Methode praktiziert wird. Wir werden im Folgenden untersuchen, in wieweit sich für das in 4.2 erzeugte Modell, wie es in 4.2.4 auf S. 83 beschrieben ist, ein Modell-Update der erwähnten Art durchführen läßt.

4.3.1 Update der Unterschriften-Längen

Das Update der Menge der gespeicherten Unterschriftenlängen geschieht durch Hinzunahme der Länge der neuen Unterschriftensequenz nach deren Vorverarbeitung gemäß 4.1; im Falle einer *Aktualisierung* wird zusätzlich die älteste Längenangabe des Modells entfernt.

4.3.2 Update des Prädiktors

Für ein Update des Prädiktors bedarf es der Neugenerierung durch den in 4.2.1.2.2 vorgestellten Genetischen Algorithmus. Notwendig hierzu sind jedoch die (vorverarbeiteten) originalen Trainingsunterschriften, die in unserem Modell nicht enthalten sind. Somit ist ein Update des Prädiktors mit dem hier verwendeten Modell-Format nicht durchführbar. Der zu erwartende Nachteil dürfte allerdings gering sein, denn wir hatten bei der Betrachtung der Qualität der Prädiktorgenerierung in 4.2.1.2.3 beobachtet, daß ein für eine Unterschrift guter Prädiktor oft auch für alle anderen, und damit insbesondere auch für modifizierte *eigene* Unterschriften gute Ergebnisse liefert.

4.3.3 Update der Segment-Codierungen

Für eine neue (vorverarbeitete) Unterschrift wird mit dem im Modell gespeicherten Prädiktor die Sequenz der Prädiktionsfehler berechnet gemäß 4.2.1.1, aus der dann die Segmentierungsstellen mit der Methode aus 4.2.1.3 gewonnen werden. Die für die Berechnung des *Pruning-Radius* notwendige *mittlere Länge der Trainingsunterschriften* läßt sich aus den mit dem Modell gespeicherten Unterschriftenlängen der alten Trainingsunterschriften ermitteln. Nach einer *segmentweisen Featureextraktion* gemäß 4.2.2.1 existiert eine neue Featurevektor-Sequenz. Diese wird den alten Segmentcodierungs-Sequenzen hinzugefügt; im Falle einer *Aktualisierung* wird zusätzlich die älteste Featurevektor-Sequenz des Modells entfernt.

4.3.4 Update der Partitions-Sequenz

Für die in 4.3.3 neu generierte Menge an Codierungs-Sequenzen wird mit den in 4.2.3 vorgestellten Schritten (unter Auslassung der abschließenden *Berechnung der Modellsequenz* in 4.2.3.5) eine neue Partitions-Sequenz erzeugt. Die alte Partitions-Sequenz des Modells wird durch diese vollständig ersetzt.

4.3.5 Beurteilung des Modell-Updates

Daß sich der für die Segmentierung einer Unterschrift nötige adaptiv generierte Prädiktor nicht ohne zusätzliche Speicherung der Originalunterschriften aktualisieren läßt, ist als deutlicher Nachteil zu werten, wie wohl sich dies in der Praxis vielleicht nicht als schweres Problem erweisen wird. Abgesehen davon gelingt das Update des Modells in sehr natürlich wirkender Weise: Es werden dazu weder neue Konzepte eingeführt, noch sind dem Modell komplexe Hilfsinformationen hinzuzufügen. Die einzige Zusatzinformation, welche ausschließlich für das Update benötigt wird, ist die zeitliche *Reihenfolge* der Einfügungen der Informationen in das Modell. Dies hatten wir bereits in 4.2.4 vorgesehen, und die technische Umsetzung dessen läßt sich in trivialer Weise durch Verwendung von Feldern oder verketteten Listen realisieren. Der *Zeitaufwand* für ein Update dürfte allerdings, trotz der fehlenden Prädiktorgenerierung, in der Anwendung nicht viel geringer ausfallen als der Aufwand für die Modellgenerierung selbst, da der komplette Alignmentgraph neu aufgebaut und partitioniert werden muß (vgl. die entsprechenden Laufzeitabschätzungen in 4.2.3.1 und 4.2.3.2). Allerdings wird man ein Modell-Update relativ selten benötigen, und man kann es dann z.B. über Nacht („offline“) durchführen.

4.4 Verifikation

In den folgenden Abschnitten werden die einzelnen Stufen des vom Autor entwickelten Verifikationsprozesses dargestellt. Die Verifikation vergleicht eine gemäß 4.1 vorverarbeitete Einzelunterschrift, die sog. *Query*, mit einem gemäß 4.2 generierten Modell. Als zusätzliches Argument wird ein *Separations-Threshold* τ_{Sep} im Sinne von 2.5.4.1 übergeben. Das Ergebnis wird eine Zahl zwischen -1 und $+1$ sein, wobei negative Zahlen die Ablehnung, positive Zahlen die Akzeptanz der Query-Unterschrift anzeigen, und der Absolutbetrag des Ergebnisses ein relatives Maß für die Vertrauenswürdigkeit der Diagnose ist.

4.4.1 Segmentierung

Die Queryunterschrift wird in der gleichen Weise in Segmente zerlegt, wie dies beim Enrollment in 4.2.1 für die Trainingsunterschriften geschehen ist. Grundlage hierfür ist der *Prädiktor des Vergleichsmodells* (vgl. 4.2.1.2), mit dem eine *Prädiktionsfehler-Sequenz* gemäß Def. 4.1, S. 62, für die Query berechnet wird. Entsprechend 4.2.1.3 werden nun die als Segmentierungsstellen in Frage kommenden *lokalen Maxima der Fehlersequenz* bestimmt. Entscheidend ist hierbei die Wahl des *Pruning-Radius* r, welcher gemäß (4.16) auf S. 70 zu wählen ist: Aus den

mit dem Modell gespeicherten physikalischen *Längen aller Trainingsunterschriften* (vgl. 4.2.4) läßt sich der hierfür notwendige *Längenmittelwert* $\bar{T}$ berechnen, während der durch (4.15) auf S. 70 festgelegte *Basisradius* r^*_{Seg} eine systemglobale Konstante darstellt.

4.4.2 Alignment

Für die im vorangegangenen Abschnitt erzeugte Segmentsequenz einer Query u^* wird ein *Alignment* gemäß 4.2.2 gegen die Sequenz $(\boldsymbol{\mu}_k)_k$ der *mittleren Segment-Featurevektoren* eines Modells $\mathcal{M}$, wie sie in 4.2.3.5 erzeugt wurde, durchgeführt (die *Standardabweichungen* der Features bleiben in diesem Schritt unberücksichtigt). Hierzu wird für die Segmente der Query zuvor eine *Featureextraktion* gemäß 4.2.2.1 vollzogen. Die Modell-Mittelsequenz $(\boldsymbol{\mu}_k)_k$ war unser Repräsentant für die „wahre“ Unterschrift einer Person, sodaß, falls die Modellierung gelungen ist, wir für Querys der *eigenen* Person im Großen und Ganzen die inhaltlich zueinanderpassenden Segmente zwischen Modell $\mathcal{M}$ und Query u^* auffinden sollten.

Welches Ergebnis das Alignment für Queryunterschriften einer *fremden* Person bringt, ist *inhaltlich* gesehen völlig unklar, da es in einer solchen Unterschrift keine zur Modellunterschrift „passenden“ Segmente geben wird: Wir vergleichen hier gewissermaßen Äpfel mit Birnen. *Formal* betrachtet wird der verwendete DTW-Algorithmus aus 2.5.5.2.2 ein minimales Alignment bestimmen, sodaß *irgendwelche* Segmente mit geringer Featuredistanz einander zugeordnet werden; in einigen Fällen wird es auch Gap-Fehler (vgl. 2.5.5.2.1) geben. Es ist aber zu erwarten, daß der Gesamtfehler des Alignments verhältnismäßig hoch sein wird im Vergleich zum Alignmentfehler für zwei Unterschriften der *gleichen* Person. Betrachten wir dazu noch einmal Abb. 4.12 auf S. 73. Neben zwei Unterschriften der gleichen Person ist dort auch die Segmentcodesequenz einer zufällig ausgewählten Fremdunterschrift (gepunktete rote Linie) dargestellt. Die Segmentfeatures dieser Unterschrift kommen in einigen Fällen nahe an die entsprechenden Features der Eigenunterschriften heran, in anderen Fällen haben sie einen weiten Abstand. Wir erinnern uns daran, wie die Features in 4.2.2.1 gewählt waren: Es handelte sich i.w. um die *Mittelwerte* und *Standardabweichungen* der Segment-Signalkomponenten. Den Autor würde es nicht wundern zu erfahren, daß sich die *Folge der Featurewert-Abstände* für zwei beliebig gewählte Fremdunterschriften ähnlich einer *Zufallsfolge* verhalten, denn es scheint keinen Grund für die Existenz von Abhängigkeiten zwischen Abschnitten von Unterschriften *verschiedener* Personen zu geben. Ein statistischer Test dieser Vermutung steht derzeit noch aus.

4.4.3 Struktur-Ähnlichkeit

In diesem Abschnitt wird eine *Strukturähnlichkeit* zwischen Query und Modellsequenz definiert. Ausgangspunkt sind die *Paare von Segment-Featurevektoren*, wie sie durch das Alignment in 4.4.2 gebildet worden waren. Wir betrachten zuerst *„reguläre“* Segmentpaare mit zwei Featurevektoren $\boldsymbol{x}^{(1)}$ und $\boldsymbol{x}^{(2)}$, *Gap-Paare* (vgl. 2.5.5.2.1) werden wir weiter unten im Text behandeln.

Für die Ähnlichkeitsbestimmung eines Segmentpaares $\left((u^{(1)})_{l_1}^{r_1}, (u^{(2)})_{l_2}^{r_2} \right)$ betrachten wir erneut die Segment-Featurevektoren $\boldsymbol{x}^{(1)}$ und $\boldsymbol{x}^{(2)}$ zu den in (4.17) auf S. 72 definierten *statistischen Features* $f_{j,1}$ und $f_{j,2}$, für $j \in \{1,2,3\}$. Wir werden diese eigentlich für das Alignment konstruierten Merkmale auch für Ähnlichkeitsaussagen heranziehen, da nach der Diskussion im vorangegangen Abschnitt zu vermuten ist, daß der Alignmentfehler für Unterschriften der gleichen Person i.d.R. geringer ausfallen wird als für Fremdunterschriften. Anders als beim Alignment wird für den Ähnlichkeitsvergleich allerdings neben der *Modell-Mittelsequenz* $(\boldsymbol{\mu}_k)$ auch die ebenfalls in 4.2.3.5 konstruierte *Sequenz der Feature-Standardabweichungen* $(\boldsymbol{\sigma}_k)$ eines Modells verwendet werden. Die Idee ist, daß es für eine *Eigenunterschrift* üblicherweise keine Rolle spielen wird, welche absolute Größe die Featurestreuungen $\sigma_{j,k}$ zum j-ten Feature f_j besitzen: Geht man von einer plausibel erscheinenden *gaußischen* Streuung um ein Mittelfeature $\mu_{j,k}$ aus, so ist z.B. für das Intervall $I := [\mu_{j,k} - 2\sigma_{j,k}\,,\, \mu_{j,k} + 2\sigma_{j,k}]$ zu er-

warten, daß für ca. 95% aller Unterschriften der gleichen Person das j-te Feature des k-ten Segmentes (bezogen auf das Alignment) einen Wert $x_{j,k} \in I$ besitzen wird. Für *Fremdunterschriften* stellt eine geringe Standardabweichung $\sigma_{j,k}$ jedoch ein Problem dar, denn wie bereits in 4.4.2 vermutet, dürfte der Abstand zwischen den Segment-Features von Unterschriften unterschiedlicher Personen weitgehend ein *Zufallswert* im Wertebereich des Features (bei uns also grob im Bereich $[-1,+1]$) sein, sodaß die Wahrscheinlichkeit im „Streuungsbereich" des Mittelwertes $\mu_{j,k}$ zu liegen mit abnehmender Feature-Standardabweichung $\sigma_{j,k}$ geringer wird.

Die angestrebte Ähnlichkeitsmessung für ein Segmentpaar $(\boldsymbol{x}, \boldsymbol{\mu})$ mit dem Query-Featurevektor $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$ und dem Modell-Mittelvektor $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^\top$, unter Berücksichtigung der Modell-Feature-Standardabweichungen $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)^\top$, führen wir obigen Überlegungen gemäß wie folgt durch:

1. Wir definieren den *relativen Abstand* $\mathrm{d}_j(\boldsymbol{x}, \boldsymbol{\mu})$ zum j-ten Feature durch

$$\mathrm{d}_j(\boldsymbol{x}, \boldsymbol{\mu}) := \frac{|x_j - \mu_j|}{\sigma_j}$$

2. Wir wählen aus der Menge aller Segmentfeatures einen Anteil $\alpha < 1$ der *kleinsten* Streuungen, d.h. wir verwenden eine Permutation $\pi \in S_n$ mit

$$\sigma_{\pi(1)} \leq \cdots \leq \sigma_{\pi(\alpha n)} \leq \cdots \leq \sigma_{\pi(n)}$$

und nutzen für die Definition des *mittleren relativen Abstandes* $\mathrm{d}(\boldsymbol{x}, \boldsymbol{\mu})$ nur die Features mit den Indexen $\pi(1), \cdots, \pi(\alpha n)$:

$$\mathrm{d}(\boldsymbol{x}, \boldsymbol{\mu}) := \big\langle \mathrm{d}_{\pi(j)}(\boldsymbol{x}, \boldsymbol{\mu}) \big\rangle_{1 \leq j \leq \alpha n}$$

Damit werden wir es nach obiger Diskussion Fremdunterschriften erschweren gute Ähnlichkeitsbewertungen zu erhalten. Experimente legen folgende Wahl für α nahe:

$$\boxed{\alpha_{\mathrm{Feat}} := 0.5} \qquad \textit{(Feature-Ignoranz-Faktor)} \tag{4.23}$$

3. Die *Ähnlichkeit* $\mathrm{sim}(\boldsymbol{x}, \boldsymbol{\mu})$ zwischen den beiden Featurevektoren wird definiert als

$$\mathrm{sim}(\boldsymbol{x}, \boldsymbol{\mu}) := e^{-\left(\frac{\mathrm{d}(\boldsymbol{x}, \boldsymbol{\mu})}{\rho}\right)^2}$$

Mit der Verwendung der *Gaußfunktion* soll die oben vermutete *gaußverteilte Streuung* bei Eigenunterschriften berücksichtigt werden.[6] Es gilt offenbar $\mathrm{sim}(\boldsymbol{x}, \boldsymbol{\mu}) \in [0, 1]$. Der *„Feature-Toleranz-Radius"* ρ bestimmt dabei grob formuliert die Größe des „Streuungsgebietes" im Featureraum $\mathbb{F}$, innerhalb dem Segmente einen hohen Ähnlichkeitswert erhalten. Experimente sprechen für die Setzung

$$\boxed{\rho_{\mathrm{Struct}} := 2.0} \qquad \textit{(Feature-Toleranz-Radius)} \tag{4.24}$$

Das mittels der Gaußfunktion definierte Ähnlichkeitsmaß $\mathrm{sim}(\cdot, \cdot)$ wird Query-Segmenten $\boldsymbol{x}$ mit größerem Abstand zum Mittelsegment $\boldsymbol{\mu}$ einen Wert nahe 0 zuweisen. Es erscheint dem Autor daher nicht gerechtfertigt, *Gap-Paaren* $(\boldsymbol{x}, \perp)$ bzw. $(\perp, \boldsymbol{x})$, welche für Segmente *ohne* passendes Partnersegment stehen, einen höheren Ähnlichkeitswert $s_\perp$ zuzuordnen:

$$\boxed{s_{\mathrm{Struct},\perp} := 0} \qquad \textit{(Gap-Paar-Ähnlichkeit)} \tag{4.25}$$

[6] Vergleiche auch die Wahl der Ähnlichkeitsfunktion für das Beispielmodell in 2.5.4.2.2.

Die *Struktur-Ähnlichkeit* $\mathrm{sim}^{\mathrm{Struct}}(\mathcal{M}, u^*)$ zwischen einem Modell $\mathcal{M}$ und einer Query u^* mit dem Alignment $((\boldsymbol{x}_k, \boldsymbol{\mu}_k))_k$ zwischen den Segment-Featurevektoren $\boldsymbol{x}_k$ von u^* und den Mittel-Vektoren $\boldsymbol{\mu}_k$ von $\mathcal{M}$ wird nun definiert als *mittlere Segmentpaar-Ähnlichkeit*:

$$\mathrm{sim}^{\mathrm{Struct}}(\mathcal{M}, u^*) := \langle \mathrm{sim}(\boldsymbol{x}_k, \boldsymbol{\mu}_k) \rangle_k \qquad \textit{(Struktur-Ähnlichkeit)} \tag{4.26}$$

Damit liegt auch die Struktur-Ähnlichkeit im Bereich $[0, 1]$.

Versuche wurden durchgeführt, bei denen Segmentpaare mit besonders niedriger bzw. hoher Ähnlichkeitsbewertung von der Berechnung der Strukturähnlichkeit gemäß (4.26) ausgenommen wurden, um damit, ähnlich wie zuvor für Segmentfeatures, eine Benachteiligung für Fremdunterschriften zu erreichen. Die Idee hierbei war, daß der Anteil an Segmentpaaren mit niedriger Bewertung für Fremdunterschriften i.a. höher ausfallen könnte als für Eigenunterschriften. In den Experimenten hat dieser Ansatz aber zu keinen signifikanten Verbesserungen geführt, weswegen hier nur der Vollständigkeit halber ein zusätzlicher Systemparameter, der *„Segmentpaar-Ignoranz-Faktor“*, eingeführt werden soll:

$$\boxed{\alpha_{\mathrm{Seg}} := 0} \qquad \textit{(Segmentpaar-Ignoranz-Faktor)} \tag{4.27}$$

4.4.4 Längen-Ähnlichkeit

Der Segmentierungsalgorithmus aus 4.4.1 sorgt durch die in 4.2.1.3.3 definierte Methode zur Festlegung der Segmentgröße dafür, daß zwei Unterschriften der *gleichen* Person in etwa die gleiche Anzahl an Segmenten erhalten, unabhängig davon wie stark sich die Unterschriften hinsichtlich ihrer physikalischen Längen unterscheiden. Die angegebene Realisierung dieser an sich wünschenswerten Eigenschaft bringt es allerdings mit sich, daß *sämtliche* Unterschriften, auch Fremdunterschriften und Fälschungen, auf eine ähnliche Segmentanzahl hin zerlegt werden. Hätte die Segmentsequenz einer Fremdunterschrift eine von einer Originalunterschrift deutlich verschiedene Länge, so würde sich vermutlich bereits dadurch eine geringe Strukturähnlichkeit in 4.4.3 ergeben, daß zuvor beim Alignment in 4.4.2 viele *Gap-Fehler* entstanden sind. Es ist zwar auch dann, wenn eine Unterschrift in für sie „unnatürlich“ lange oder kurze Segmente zerlegt wird, kaum zu erwarten, daß sich eine hohe strukturelle Übereinstimmung mit der Modellsequenz einer anderen Person ergeben wird, aber zumindest was die allgemeine *Trennschärfe* zwischen Fremd- und Eigenunterschriften betrifft erscheinen Einbußen möglich.

Wir werden nun ein Verfahren angeben, welches die *physikalische* Länge einer Unterschrift, d.h. die Anzahl an Samples in der (vorverarbeiteten) Ausgangssequenz, dazu verwendet, möglichst viele Fremdunterschriften zu eliminieren, ohne daß im Idealfall dabei Eigenunterschriften zu Schaden kommen. Die Idee dazu ist, aus den physikalischen Längen $|u^{(i)}|$ der Trainingsunterschriften $\{u^{(1)}, \ldots, u^{(m)}\}$, wie sie nach 4.2.4 im Modell $\mathcal{M}$ gespeichert sind, die „übliche“ Länge einer Unterschrift der modellierten Person, zusammen mit ihrer Variabilität abzuschätzen. Dabei wollen wir Unterschriften mit der *halben* üblichen Länge in der gleichen Weise bewerten wie Unterschriften mit der *doppelten* üblichen Länge, denn dies erscheint bei solchen Längenvergleichen plausibel, bei denen sehr unterschiedliche Distanzen auftreten können wie in unserem Fall. Allgemein möchten wir zu zwei Zahlen $a > 0$ und $b > 0$ deren „Mitte“ μ so bestimmen, daß die Verhältnisbeziehung $a : \mu = \mu : b$ gilt. Als die übliche Länge wählen wir daher das *geometrische Mittel* [Stö93] der Längen der Trainingsunterschriften:

$$\mu := \sqrt[m]{\prod_{i=1}^{m} |u^{(i)}|} \qquad \textit{(Geometrisches Mittel der Unterschriftenlängen)}$$

Da wir es von nun an mit Längen*verhältnissen* zu tun haben, definieren wir den *Längenunterschied* zwischen zwei Unterschriften u und u' als *prozentuale* Abweichung:

$$\mathrm{d}(u, u') := \frac{max(|u|, |u'|)}{min(|u|, |u'|)} \; - \; 1 \qquad \textit{(Längenunterschied)}$$

Damit gilt stets $\mathrm{d}(u,u') \geq 0$, und es ist $\mathrm{d}(u,u') = 0$ genau dann wenn $|u| = |u'|$ ist. $\mathrm{d}(\cdot,\cdot)$ ist ferner symmetrisch, womit zumindest ein *Teil* der in (2.17) auf S. 36 angegebenen *Metrik-Eigenschaften* erfüllt sind (die Dreiecksungleichung werden wir für unserem Anwendungsfall nicht benötigen).

Die *Variabilität* der Längen aller Trainingsunterschriften ist definiert als *mittlerer Längenunterschied zu* μ durch

$$\sigma := \sqrt[m]{\prod_{i=1}^{m} \frac{max(|u^{(i)}|,\mu)}{min(|u^{(i)}|,\mu)}} - 1 \qquad \textit{(Variabilität)}$$

Wir können nun den Längenunterschied $\mathrm{d}(u^*,\mu)$ einer Einzelunterschrift u^* zur üblichen Länge μ der Unterschriften der modellierten Person mit der Variabilität σ vergleichen, und erhalten dadurch den *relativen Längenunterschied* von u^* im Vergleich zu μ:

$$\tilde{\mathrm{d}}_\mu(u^*) := \mathrm{d}(u^*,\mu)/\sigma \qquad \textit{(relativer Längenunterschied)}$$

Eigenunterschriften werden i.a. einen relativen Längenunterschied nicht viel größer als 1 besitzen, bei Fremdunterschriften können hingegen sehr große Werte entstehen.[7]

Unsere eigentliche, zwischen 0 und +1 liegende *Längenähnlichkeit* für eine Query u^* und ein Modell $\mathcal{M}$ mit der üblichen Länge μ erreichen wir über eine *gaußische* Bewertung:

$$\mathrm{sim}^{\mathrm{Len}}(\mathcal{M},u^*) := e^{-\left(\frac{\tilde{\mathrm{d}}_\mu(u^*)}{\rho}\right)^2} \qquad \textit{(Längenähnlichkeit)}$$

Mit dem *„Längen-Toleranz-Radius“* ρ läßt sich einstellen, wie stark die Länge einer Unterschrift u^* von der üblichen Länge μ abweichen darf, um noch hoch bewertet zu werden. Für unsere Zwecke erscheint ein Wert von $\rho = 1$ als zu gering, denn daraus würde bereits bei einer durchschnittlichen Abweichung von $\tilde{\mathrm{d}}_\mu(u^*) = 1$ eine Bewertung von lediglich $e^{-1} \approx 0.368$ resultieren. Wir müssen in Hinblick auf die Gesamtbewertung (s. 4.4.5) aber sicherstellen, daß Eigenunterschriften *fast immer* eine Bewertung nahe 1 erhalten, und wir werden es deshalb hinnehmen müssen, wenn gelegentlich auch Fremdunterschriften hohe Bewertungen erhalten. Experimente sprechen für einen Toleranz-Radius der Größe

$$\boxed{\rho_{\mathrm{Len}} := 10.0} \qquad \textit{(Längen-Toleranz-Radius)} \qquad (4.28)$$

Zum Testen der angegebenen Methode wurde je ein Modell pro Probant gegen sämtliche zur Verfügung stehenden Eigenunterschriften, und gegen je 10 zufällig ausgewählte Unterschriften aller anderen Probanten (ca. 700 Stichproben pro Probant) verglichen. Abb. 4.20 zeigt die Verteilungen der ermittelten Werte als Histogramme (logarithmische Ordinate). Man erkennt, daß Eigenunterschriften fast immer Bewertungen sehr nahe bei 1 erhalten: Der Mittelwert beträgt 0.972, 94,7% aller Eigenunterschriften erhalten eine Bewertung von mehr als 0.9, und immerhin noch 87,3% erhalten mehr als 0.95. Nur für 0.6% aller Unterschriften ergibt sich ein „gefährlicher“ Wert kleiner als 0.5, wirklich ganz niedrige Werte sind sehr selten. Bei den Fremdunterschriften erhält zwar ein großer Teil an Unterschriften eine sehr niedrige Bewertung (31,2% erhalten eine Bewertung von weniger als 0.1), andererseits gibt es auch etwa genauso viele (28,7%) Unterschriften, die höher als 0.9 bewertet werden. Als Fazit ergibt sich, daß das Verfahren in der angegebenen Form seinen Zweck i.w. erfüllt: Eigenunterschriften werden fast nie beeinträchtigt, während immerhin mindestens 1/3 aller Fremdunterschriften nach Durchlaufen dieser Stufe praktisch keine Chance mehr haben werden, eine für die Akzeptanz notwendige Ähnlichkeit zu erhalten.

Es sei abschließend darauf hingewiesen, daß die Qualität des hier präsentierten Verfahrens stark von der Qualität des *Trimmers* aus 4.1.4 abhängt: Findet dieser für eine Unterschrift nicht die richtigen Randstellen, so ergeben sich völlig falsche Ähnlichkeitswerte, was speziell für Originalunterschriften von großem Nachteil ist.

[7] Den pathologischen Fall, das sämtliche Trainingsunterschriften exakt die gleiche Länge besitzen, womit $\tilde{\mathrm{d}}_\mu(\cdot)$ undefiniert wäre, wollen wir in dieser anwendungsorientierten Erörterung ignorieren.

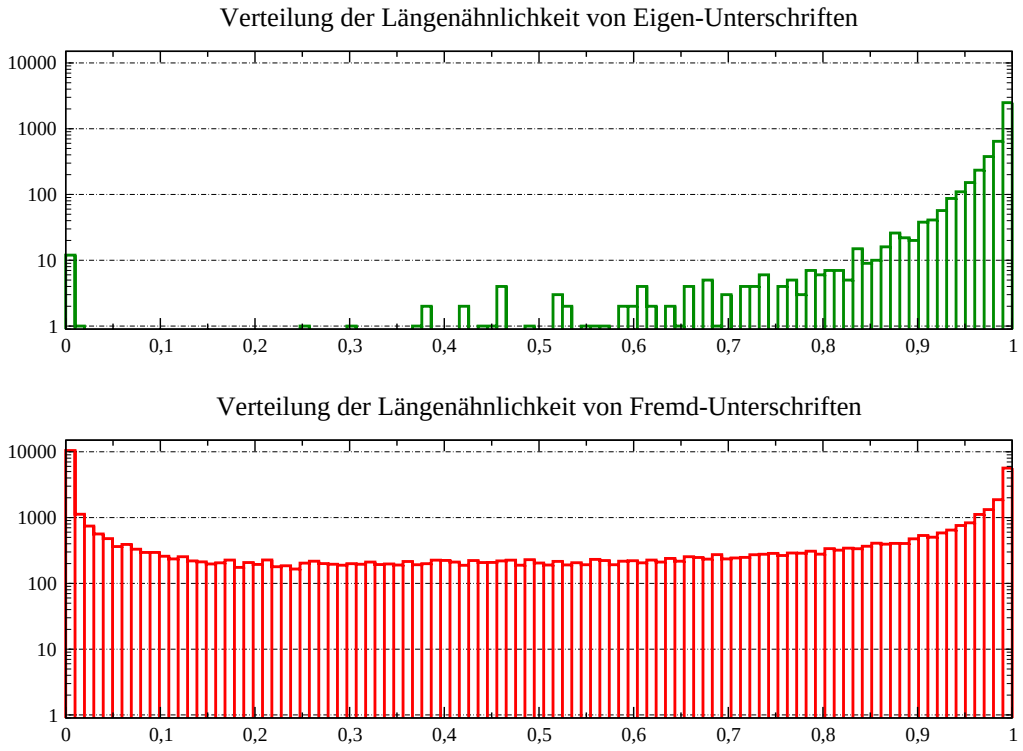

Abbildung 4.20: Längenähnlichkeiten für Fremd- und Eigenunterschriften

4.4.5 Definition der Gesamtbewertung

Die Gesamtbewertung für eine Unterschrift relativ zu einem Modell setzt sich zusammen aus der *Strukturähnlichkeit* gemäß 4.4.3 und der *Längenähnlichkeit* nach 4.4.4. Die Längenähnlichkeit war in einer sehr konservativen Weise definiert worden, indem sie Eigenunterschriften in den meisten Fällen einen Wert von ungefähr 1 zuweist. Damit können wir die Längenähnlichkeit gewissermaßen als „Sieb“ verwenden: Liefert sie für eine Unterschrift einen geringen Wert, so handelt es sich mit großer Zuverlässigkeit um eine Fremdunterschrift, andernfalls muß mit Hilfe der Strukturähnlichkeit eine endgültige Entscheidung getroffen werden. Wir definieren zu diesem Zweck die Gesamtbewertung $\mathrm{sim}(\mathcal{M}, u^*)$ für eine Query u^* und ein Modell $\mathcal{M}$ durch *Multiplikation* der beiden zuvor betrachteten Ähnlichkeitsmaße:

$$\mathrm{sim}(\mathcal{M}, u^*) := \mathrm{sim}^{\mathrm{Len}}(\mathcal{M}, u^*) \cdot \mathrm{sim}^{\mathrm{Struct}}(\mathcal{M}, u^*) \qquad \text{(Gesamtbewertung)}$$

Der von außen vorgegebene Separationsthreshold τ_{Sep} ist ein Wert zwischen 0 und 1. Durch die Operation

$$\mathrm{sim}(\mathcal{M}, u^*) - \tau_{\mathrm{Sep}}$$

erhalten wir wie gewünscht eine Bewertung für eine Unterschrift zwischen -1 und +1, welche negativ für abgelehnte Unterschriften ist, und für die der Absolutbetrag ein relatives Maß für die Vertrauenswürdigkeit der Bewertung darstellt. Als sinnvolle *Defaultwerte* τ^*_{Sep} für den Separationsthreshold kommen vor allem zwei Zahlen in Frage:

- $\tau^*_{\mathrm{Sep}} := 0$: In diesem Fall wird der Verifizierer die „rohe“ Bewertung der Unterschrift im Bereich $[0, 1]$ als Ergebnis zurückreichen. Dies ist vor allem interessant, um die Leistungsfähigkeit des Verifikationsverfahrens beurteilen zu können. Alle Tests aus Kapitel 5 wurden mit diesem Threshold durchgeführt.
- $\tau^*_{\mathrm{Sep}} :=$ EER: Die Equal-Error-Rate (vgl. 2.6.1) des Systems wird in Kapitel 5 ermittelt werden.

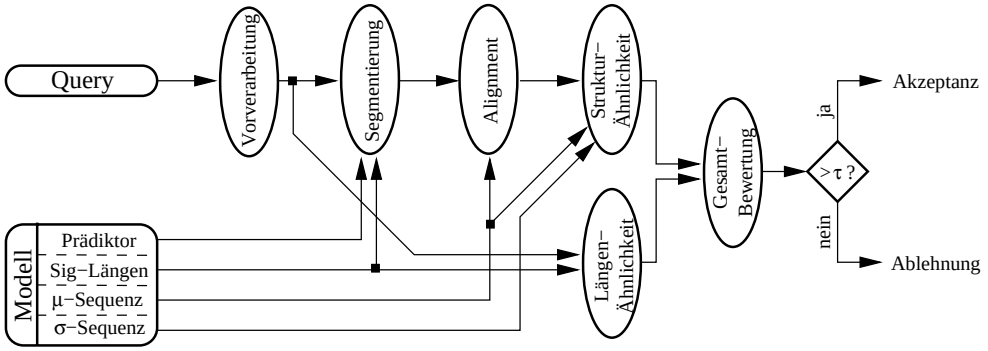

Abbildung 4.21: Ablauf des Verifikationsprozesses

4.4.6 Zusammenfassung des Verifikationsprozesses

Abb. 4.21 faßt die einzelnen Schritte der Verifikation zusammen. Verglichen wird eine unverarbeitete *Query-Unterschrift* mit einem *Modell* gemäß 4.2.4 bei Angabe eines *Separierungsthresholds* $\tau \in [0, 1]$. Als Ergebnis der Verifikation ergibt sich eine Zahl zwischen -1 (Ablehnung) und +1 (Akzeptanz). Die Query wird *vorverarbeitet* gemäß 4.1. Sie wird anschließend mit dem im Modell enthaltenen Prädiktor *segmentiert* (4.4.1), wobei für die Länge und damit implizit für die Anzahl der Segmente die im Modell enthaltenen *Längen der Trainingsunterschriften* maßgeblich sind. Danach wird ein *Alignment* zwischen Query und *Modell-Mittelsequenz* durchgeführt, wobei letztere auf der Grundlage der im Modell gespeicherten Featurevektor-Sequenzen und der Partitionssequenz berechnet wird (4.4.2). Die eigentliche Ähnlichkeitsbewertung setzt sich aus zwei Berechnungen zusammen: einem *Längenvergleich* zwischen den physikalischen Längen der Query und der Trainingsunterschriften (4.4.4), und einem *Strukturvergleich* auf der Grundlage des Featurevektor-Alignments (4.4.3). Hierbei gehen zusätzlich die *Standardabweichungen* der Featurevektoren der Trainingsunterschriften mit in die Bewertung ein; diese lassen sich ebenfalls wie die Mittelunterschrift aus den im Modell gespeicherten Informationen berechnen. Die Bewertungen für Struktur- und Längenähnlichkeit werden in einer *Gesamtbewertung* zusammengefaßt und das Ergebnis mit dem Separierungsthreshold τ_{Sep} verglichen (4.4.5).

Die für die Verifikation verwendeten Systemparameter wurden wie folgt festgelegt:

- Gap-Paar-Ähnlichkeit ((4.25) 4.4.3): $s_{\mathrm{Struct},\perp} = 0.0$
- Feature-Ignoranz-Faktor ((4.23) 4.4.3): $\alpha_{\mathrm{Feat}} = 0.5$
- Segmentpaar-Ignoranz-Faktor ((4.27) 4.4.3): $\alpha_{\mathrm{Seg}} = 0.0$
- Feature-Toleranz-Radius ((4.24) 4.4.3): $\rho_{\mathrm{Feat}} = 2.0$
- Längen-Toleranz-Radius ((4.28) 4.4.4): $\rho_{\mathrm{Len}} = 10.0$

4.5 Beseitigung der Stiftrotation

Eine Rotationsnormierung für Unterschriften basierend allein auf der jeweiligen Unterschriftensequenz ist dem Autor nicht gelungen (vgl. die Diskussion in 4.1.6). In 3.1.4 wurde allerdings eine Methode angegeben, mit der ein *manueller* Rotationsangleich zwischen zwei Unterschriften der gleichen Person unter theoretisch idealen Bedingungen stets möglich ist. Wir hatten zudem an einem realistischen Beispiel in Abb. 3.4 auf S. 50 gesehen, daß sich diese Methode in der Praxis mit recht gutem Erfolg anzuwenden scheinen läßt. Sofern dies generell zutrifft, sollte eine entsprechende Rotationskorrektur für gedrehte Unterschriften zu

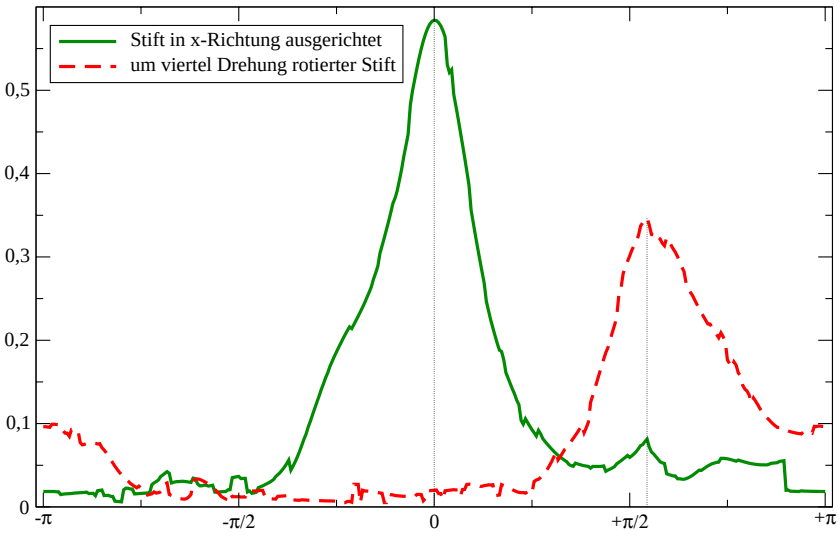

Abbildung 4.22: Bewertungen bei unterschiedlicher Rotation

einer deutlichen Verbesserung der Bewertung durch das in dieser Arbeit präsentierte Verifikationssystem führen.

Abb. 4.22 zeigt die Bewertungen (Ordinate) für die beiden in Abb. 3.4 miteinander verglichenen Unterschriften bei verschiedenen Drehwinkeln zwischen $-\pi$ und $+\pi$ (Abszisse): Die gestrichelte rote Linie gehört zur oberen, die durchgezogene grüne Linie zur unteren dort dargestellten Unterschrift. Als Separations-Threshold (vgl. 4.4.5) wurde $\tau_{\mathrm{Sep}} := 0$ verwendet, um Bewertungen zwischen 0 und 1 zu erhalten. Die untere in Abb. 3.4 gezeigte Unterschrift war durch den in x-Richtung ausgerichteten BiSP-Pen erfaßt worden, und gleiches gilt für die dem verwendeten Vergleichsmodell zugrundeliegenden Trainingsunterschriften. Gemäß 3.1.4 war die obere Unterschrift dagegen mit einem um etwa eine viertel Drehung rotierten Stift aufgezeichnet worden.

Die Bewertungskurve für die *ausgerichtete* Unterschrift besitzt ein unter den genannten Bedingungen zu erwartendes Maximum bei einem Winkel von 0°. Überraschender dürfte die Beobachtung sein, daß dieses Maximum sehr deutlich ist: Im Vergleich zu den am schlechtesten bewerteten Drehwinkeln beträgt der Unterschied mehr als das 20-fache. Darüber hinaus fällt die Kurve links und rechts des Maximums schnell ab; für verschiedene Beispiele hat sich herausgestellt, daß der Toleranzbereich für hohe Bewertungen nicht viel größer als 20° ist, was auch bei bewußt ausgerichtet gehaltenem Stift durch leichte Handbewegungen einfach überschritten werden kann.

Die Bewertung der *gedrehten* Unterschrift liegt ohne Rotationskorrektur, d.h. bei einem Winkel von 0°, nahe bei 0. Das Maximum beträgt hingegen ca. 0.35 und wird bei einem Drehwinkel von etwas mehr als $+\pi/2$ angenommen, in Übereinstimmung mit Abb. 3.4, und auch in Übereinstimmung mit der tatsächlichen Drehung des Stiftes. Diese Beobachtung deckt sich mit weiteren vom Autor betrachteten Beispielen. Damit liegt es nahe, den Rotationsangleich anhand der Bewertung durch das Verfahren selbst durchzuführen, indem nach einem *Drehwinkel mit maximaler Bewertung* gesucht wird. Dieses Verfahren ist *konservativ hinsichtlich der Ähnlichkeitsbewertung*, denn es kann nicht zu einer schlechteren Bewertung als im unrotierten Fall führen. Wie sicher der richtige Rotationswinkel aufgefunden wird, darüber läßt sich anhand der wenigen und nicht in repräsentativer Weise erfaßten rotierten Unterschriftenproben, die dem Autor zur Verfügung stehen, keine legitime statistische Aussage treffen.

In den folgenden Abschnitten werden die für die Hinzunahme der Rotationsbeseitigung zum System notwendigen Anpassungen am Enrollment, am Modell-Update und an der Verifikation beschrieben.

4.5.1 Rotationsbeseitigung bei der Verifikation

Eine adäquate Rotationsbeseitigung bei der Verifikation setzt ein Vergleichsmodell voraus, welches seinerseits aus Trainingsunterschriften mit paarweise gleicher Ausgangsrotation erzeugt wurde. Wir nehmen in diesem Abschnitt an, daß unsere Modelle diese Bedingung erfüllen. *„Unrotiert"* heißt eine Queryunterschrift dann, wenn sie mit der gleichen Grundrotation wie die Trainingsunterschriften erfaßt wurde.

Nach der Erfassung der Unterschrift wird die Queryunterschrift gemäß 4.1 *vorverarbeitet*. Aus dem Bereich $[-\pi, +\pi)$ werden äquidistant 360 Winkel $w_0, \ldots, w_{359}$ gewählt, und es wird wie folgt für jeden Winkel w_i vorgegangen:

1. Die vorverarbeitete Queryunterschrift wird um w_i *rotiert* mit der Methode aus 3.1.4.
2. Die um w_i rotierte Unterschrift wird *segmentiert* gemäß 4.4.1.
3. Für die erzeugten Segmente wird eine *Featureextraktion* und ein anschließendes *Alignment* gegen die Modell-Mittelsequenz durchgeführt gemäß 4.4.2.
4. Die *Strukturähnlichkeit* gegen die Modellsequenz wird ermittelt gemäß 4.4.3.

Der Winkel w_i^*, der die höchste Strukturähnlichkeit liefert, wird als Rotationswinkel festgelegt. Es wird nun die Längenähnlichkeit für die um w_i^* rotierte Unterschrift gemäß 4.4.4 bestimmt, und abschließend das Verifikationsergebnis gemäß 4.4.5 berechnet.

$$\boxed{\omega_{\mathrm{Rot}} := 2\pi/360} \qquad \textit{(Rotationsphasen-Granularität)} \tag{4.29}$$

4.5.2 Rotationsbeseitigung beim Modell-Update

Wir setzen auch für das Modell-Update voraus, daß das betrachtete Modell aus Trainingsunterschriften mit paarweise gleicher Ausrichtung generiert wurde. Für die neu einzufügende Unterschrift wird nun der Rotationswinkel w^* in der gleichen Weise bestimmt wie für die Verifikation in 4.5.1. Die demgemäß rotationskorrigierte Neuunterschrift kann in das Modell eingefügt werden, welches damit die Ausgangsvoraussetzung wieder erfüllt.

4.5.3 Rotationsbeseitigung beim Enrollment

Aus sämtlichen Trainingsunterschriften wird in einem ersten Schritt ein *„Protomodell"* über den in 4.2 angegebenen Enrollment-Prozeß generiert. Anschließend wird für jede Trainingsunterschrift eine Rotationskorrektur anhand dieses Protomodells in der gleichen Weise wie für die Verifikation in 4.5.1 durchgeführt. Aus diesen *rotierten* Trainingsunterschriften wird nun das endgültige Modell durch ein erneutes Enrollment gemäß 4.2 generiert.

4.5.4 Beurteilung der Rotationsbeseitigung

Für Enrollment, Update und Verifikation ergibt sich nach Betrachtung von Abb. 4.22 gleichermaßen, daß eine Zerlegung des Intervalls $[-\pi, +\pi)$ in 360 äquidistante Abschnitte hinreichend fein ist, um das gesuchte Maximum mit ausreichender Genauigkeit zu bestimmen. Der Verifikationsprozeß wird damit allerdings ebenfalls beinahe um den Faktor 360 verlangsamt. Eine etwas gröbere Auflösung brächte bei einer entsprechenden Verringerung der Laufzeit ggf. immer noch gute Ergebnisse, der Vergröberung sind allerdings enge Grenzen gesetzt, wie wir bereits in der Einleitung dieses Unterkapitels gesehen haben. Darüberhinaus ist die Anwendung anderer Methoden zur Bestimmung globaler Maxima denkbar, welche aber ebenfalls lediglich eine *Beschleunigung* des Verfahrens versprechen. Auf die Dauer des Modell-Updates und des Enrollments hat die hier verwendete „naive" Suchmethode in der Praxis hingegen so gut wie keine negativen Auswirkungen, da deren Laufzeiten durch andere Faktoren, wie die *Analysebreite* β_{Part} des Partitionierungsalgorithmus aus 4.2.3.2, dominiert werden.

Die Vorgehensweise beim Enrollment, zunächst ein *Protomodell* aus nicht rotationskorrigierten Unterschriften zu bilden und danach alle Trainingsunterschriften daran auszurichten, wirkt unausgegoren: Dieser Ansatz wird wohl nur dann brauchbare Ergebnisse liefern, wenn die Trainingsunterschriften bereits in *ähnlicher* Weise ausgerichtet waren. Der Autor hält diese an sich starke Forderung aber für praxistauglich: Man wird von den Probanten zumindest beim Enrollment (nicht unbedingt bei der Verifikation und auch nicht beim Modell-Update) verlangen dürfen, im eigenen Interesse den Smartpen bei allen abgegebenen Unterschriftenproben *ungefähr* in der gleichen Weise zu halten. Dazu genügt es bereits, die Haltung des Stiftes zwischen zwei hintereinander stattfindenden Erfassungen einfach beizubehalten. Unterstützt werden kann dies durch eine *asymmetrische* Bauform des Schreibgerätes, wie sie z.B. der *neue* BiSP-Pen besitzt (vgl. 3.1.4). Die dann noch auftretenden *kleinen* Winkeldifferenzen sollten bei einer ausreichenden Anzahl an Trainingsunterschriften die Generierung eines recht zuverlässigen Protomodells zulassen. Sofern dies zutrifft, sollte ein solches Protomodell seinerseits in der Lage sein, die verbleibenden Winkeldifferenzen endgültig zu eliminieren. Eine statistisch relevante Bestätigung dieses Gedankenexperimentes steht aufgrund des Mangels an adäquatem Datenmaterial allerdings noch aus.

Kapitel 5

Resultate

In diesem Kapitel werden die Ergebnisse diverser Experimente zum Laufzeitverhalten, Speicherplatzbedarf und zur Leistungsevaluation des in Kapitel 4 vorgestellten Verfahrens präsentiert. Es wurden für alle Betrachtungen die gleichen Systemeinstellungen, wie sie in 4.1.8, 4.2.4, 4.4.6 und durch (4.29) in 4.5 zusammengefaßt sind, verwendet, wobei der Fokus der Einstellungen auf möglichst hoher Verifikationsleistung lag.

5.1 Leistungsevaluation

Für jeden der 70 gemäß Kapitel 3 zur Verfügung stehenden Probanten wurden 5 Modelle generiert, wobei jedes Enrollment mit 10 Unterschriften des jeweiligen Probanten durchgeführt wurde. Die Auswahl der Trainingsunterschriften geschah stets zufällig. Die verwendeten Unterschriften wurden zusammen mit dem generierten Modell für eine spätere Ermittlung des Trainingsfehlers abgespeichert. Es sei bemerkt, daß selbst in den Fällen, in denen ein Probant lediglich 10 Unterschriften zur Verfügung gestellt hatte, durch den für die Prädiktorgenerierung eingesetzten Genetischen Algorithmus aus 4.2.1.2.2, und den damit vorhandenen probabilistischen Anteil des Modellierungsprozesses, dennoch stets unterschiedliche Modelle entstanden sind. Es wurde nun für jedes der so erzeugten 350 Modelle und für jede der insgesamt 4464 Unterschriften (vgl. Kapitel 3) eine Verifikation durchgeführt. Basierend auf diesen ca. 1.5 Mill. Einzelergebnissen[1] wurden verschiedene Statistiken erstellt, die im folgenden vorgestellt werden sollen. Die Bezeichnungen und durchgeführten Berechnungen folgen denjenigen aus 2.6.1.

5.1.1 Intra-Probant-Variabilität

Es soll zunächst ein Eindruck davon gewonnen werden, wie stark die Verifikationsergebnisse für einen Probanten von der Wahl der Trainingsunterschriften und den probabilistischen Anteilen des Modellierungsprozesses abhängig sind. Abb. 5.1 stellt das Ergebnis der Verifikation zweier verschiedener Modelle des gleichen Probanten gegen sämtliche 4464 Einzelunterschriften dar.

Das Histogramm links oben stellt für jede mögliche Verifikationsbewertung zwischen 0 und 1 in logarithmischer Darstellung die Anzahl der Probanten dar, welche die jeweilige Bewertung erhalten haben. Sämtliche Fremdunterschriften, markiert durch rot gestrichelte Balken, haben eine Bewertung von unter 0.25 erhalten. Die grün durchgezogenen Balken repräsentieren die zum Test eingesetzten Eigenunterschriften. Ihre Bewertung ist in den meisten Fällen größer ausgefallen als die der Fremdunterschriften. Die zum Training eingesetzten

[1] Der Rechenaufwand für diese Evaluation war verhältnismäßig hoch (vgl. hierzu auch die Ergebnisse in 5.2). Tatsächlich waren mit dieser Aufgabe 57 moderne Computer des Fachbereichs Informatik der Universität Frankfurt rund eine Woche lang beschäftigt. Der Autor möchte sich an dieser Stelle bei der Rechnerbetriebsgruppe Informatik (RBI) für deren Kooperation bedanken.

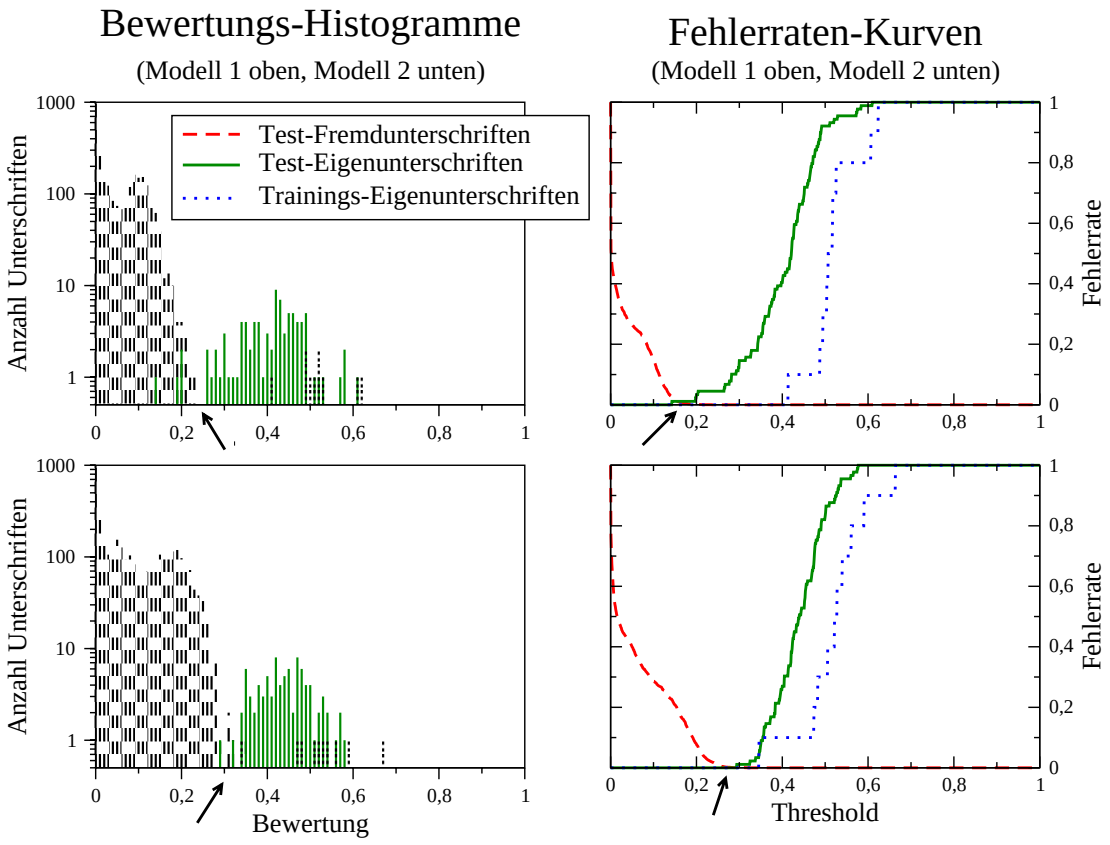

Abbildung 5.1: Verifikationsleistung für zwei Modelle des gleichen Probanten

Unterschriften werden hier blau gepunktet dargestellt. Sie haben ohne Ausnahme einen deutlichen Abstand zu sämtlichen Fremdunterschriften, und besitzen in den meisten Fällen auch im Vergleich zu den Test-Eigenunterschriften eine relativ hohe Bewertung.

Das Schaubild rechts oben gibt die Fehlerraten für einen variablen Separations-Threshold $\tau \in [0, 1]$ wieder, wenn alle Unterschriften u mit einer Bewertung $\alpha_u < \tau$ vom System abgelehnt werden (vgl. 2.6.1.1.3). Wie zu erwarten fällt die FAR-Kurve (rot gestrichelt) mit wachsendem Threshold τ ab, d.h. Fremdunterschriften werden immer häufiger korrekt abgelehnt, während die beiden FRR-Kurven für den Trainingsfehler (blau gepunktet) und den Testfehler (grün durchgezogen) dann zunehmend ansteigen. So wie im Histogramm links die Trainingsunterschriften im Vergleich zu den Testunterschriften sehr hohe Bewertungen bekommen haben, so liegt bei festgewähltem Threshold der Trainingsfehler durchgehend unterhalb des Testfehlers. Die Kurven für Test-FAR und Test-FRR kreuzen sich ungefähr an derjenigen Stelle (Pfeil rechts), wo man im Histogramm eine Trennlinie zwischen Fremd- und Eigenunterschriften ziehen würde (Pfeil links). Die Kreuzungsstelle der Fehlerratenkurven wollen wir die *„modellspezifische EER“* nennen.

Die beiden Schaubilder unten stellen die analoge Situation für das zweite generierte Modell des Probanten dar. Im Vergleich zum ersten Modell fallen vor allem zwei Dinge auf. *Erstens:* Auch hier haben wir eine deutliche Unterbewertung aller Fremdunterschriften im Vergleich zu allen Eigenunterschriften zu verzeichnen, und die Trainingsunterschriften werden auch für dieses Modell in der Mehrzahl der Fälle relativ hoch bewertet. Diese Analogie der grundsätzlichen Verhältnisse spiegelt sich ebenso in den Fehlerratenkurven wider. *Zweitens:* Die „natürliche“ Trennstelle zwischen Fremd- und Eigenunterschriften (Pfeil links) bzw. die modellspezifische EER (Pfeil rechts) ist im Vergleich zum ersten Modell sichtbar zu höheren Bewertungen hin verschoben. Dieses Phänomen wird uns später die Aufgabe erschweren, eine geeignete *personenunabhängige* Trennstelle zu bestimmen.

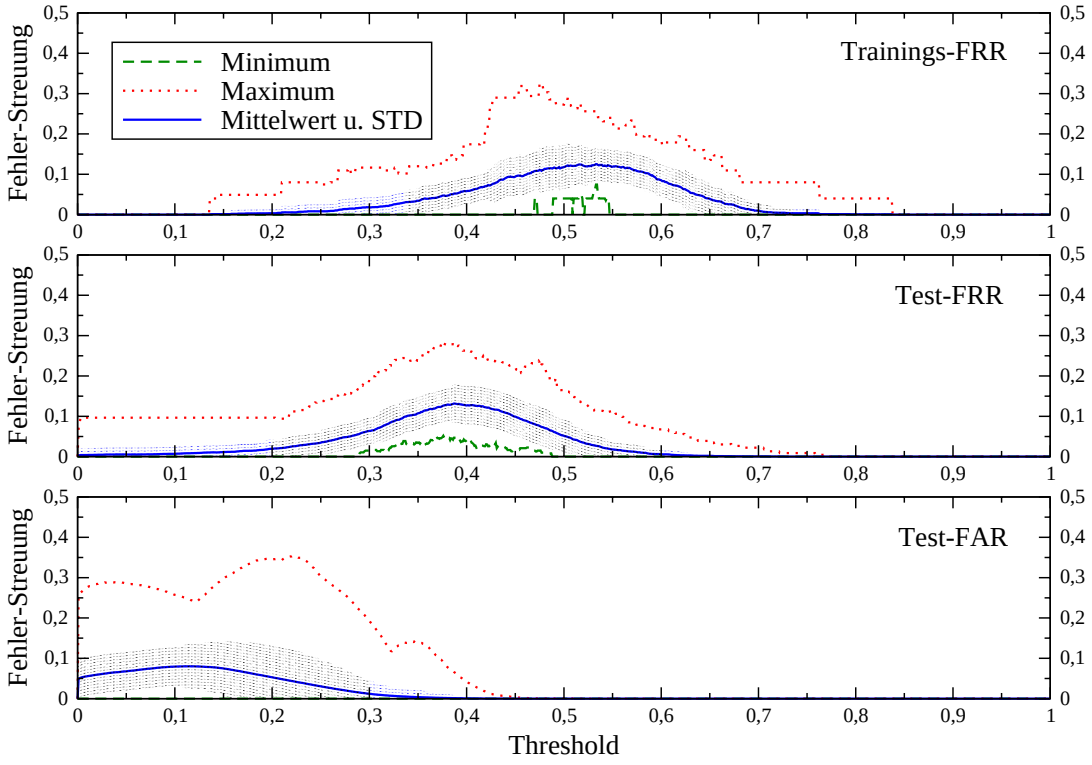

Abbildung 5.2: Allgemeiner Vergleich verschiedener Modelle des gleichen Probanten

Wir betrachten nun die allgemeine Situation. Im Sinne von Abb. 5.1 wurden für jedes Modell eines Probanten Fehlerratenkurven erstellt für die Trainings-FRR, die Test-FRR und die Test-FAR. Für Abb. 5.2, oben, wurde in einem ersten Schritt aus sämtlichen FRR-Kurven für die Trainingsdaten eines Probanten die Kurve der *punktweisen Standardabweichungen* gebildet, d.h. wir erhalten ein Maß für die *probantenspezifische Fehlerraten-Variabilität* in jeder Bewertungsstelle $\alpha \in [0, 1]$. Hieraus wurde nun für jede Bewertungsstelle jeweils die *minimale* (gestrichelte grüne Kurve), die *maximale* (gepunktete rote Kurve), sowie die *mittlere* Standardabweichung (durchgezogene blaue Kurve) über sämtliche 70 Probanten berechnet. Der schraffierte, blaue Kontext, welcher die Mittelwert-Kurve umgibt, gibt die Standardabweichung der Probantenkurven wieder.

Man stellt fest, daß die probantenspezifische Variabilität bei der Trainings-FRR für Bewertungen zwischen ca. 0.3 und 0.7 recht groß ist: Streuungen von mehr als 0.1 im Mittelwert sind hier nicht selten, einzelne Probanten erreichen für manche Bewertungen sogar Standardabweichungen in ihren Fehlerraten von 0.3 und mehr. Ähnliche Aussagen gelten für die Test-FRR (mittleres Schaubild): Hier liegt der besonders variable Bereich zwischen den Bewertungen 0.2 und 0.55. Für die Test-FAR (unteres Schaubild) gelten analoge Verhältnisse.

5.1.2 Inter-Probant-Variabilität

Wir vergleichen als nächstes die Verifikations-Ergebnisse verschiedener Probanten. Abb. 5.3 stellt den Ergebnissen des Modells aus Abb. 5.1 (oben) die Verifikationsleistung des Modells eines zweiten Probanten (Schaubilder unten) gegenüber.

Wie schon im vorangegangenen Abschnitt ähneln sich sowohl die Histogramme als auch die Fehlerraten-Kurven beider Modelle hinsichtlich ihrer groben Struktur. In dem hier betrachteten Beispiel befinden sich die modellspezifischen EERs der beiden Probanten an recht ähnlichen Positionen. Einen generelleren Blick liefert Abb. 5.4. Für das obere Schaubild wurde in einem ersten Schritt aus sämtlichen Trainings-FRR-Kurven eines Probanten die Kurve der *punktweisen Mittelwerte* gebildet, womit die „tatsächliche“ Fehlerratenkurve dieses Pro-

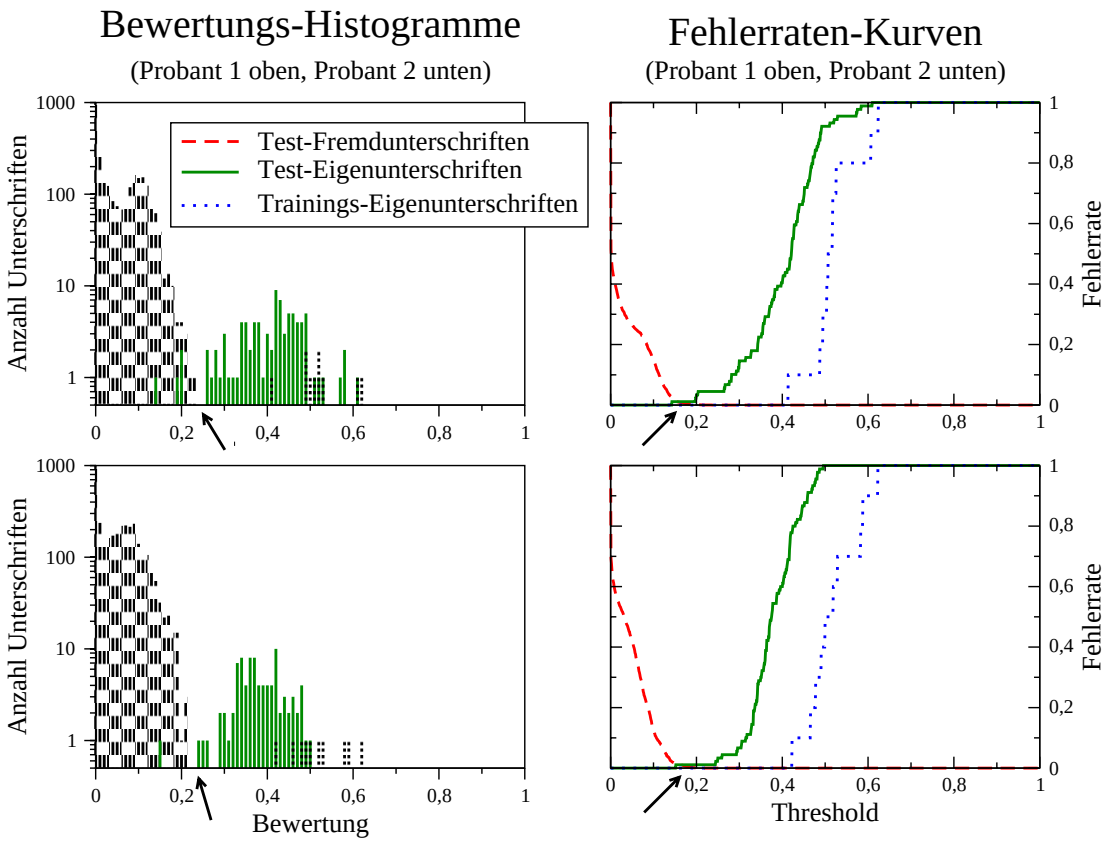

Abbildung 5.3: Verifikationsleistung für zwei Modelle verschiedener Probanten

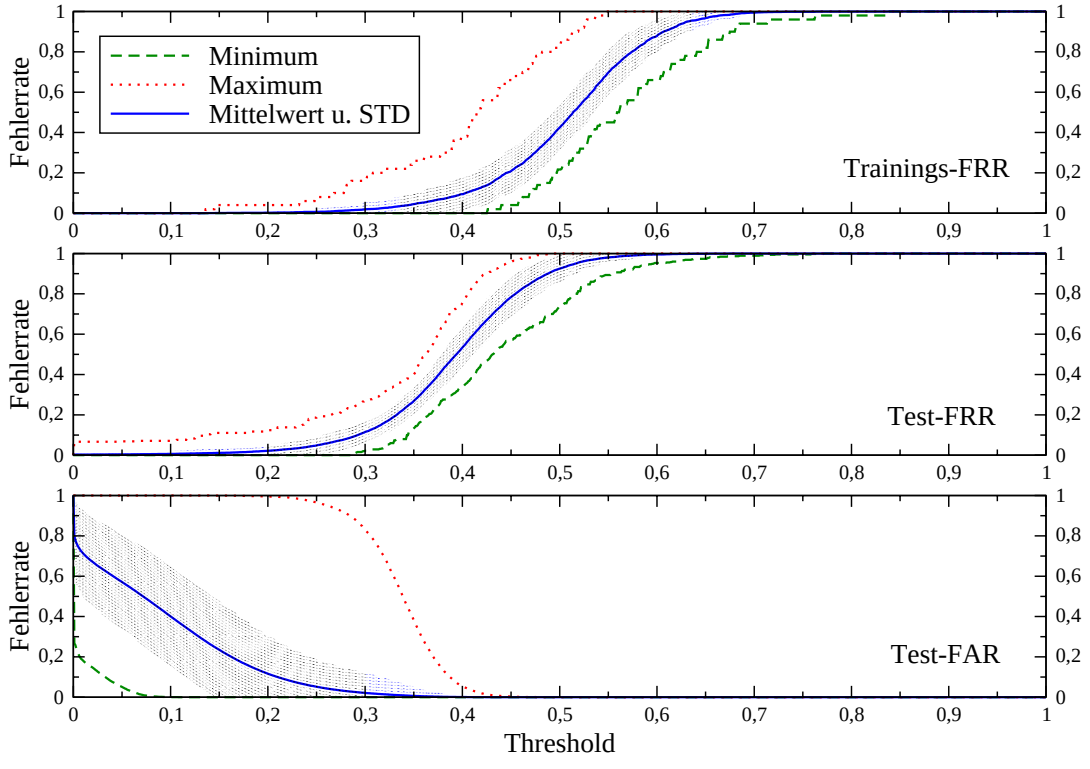

Abbildung 5.4: Allgemeiner Vergleich verschiedener Probanten

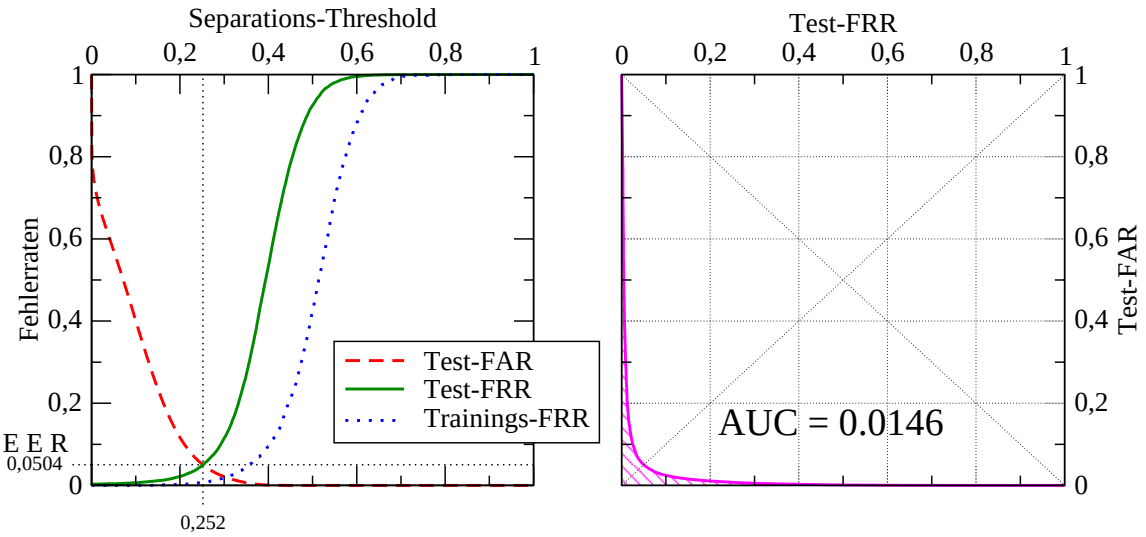

Abbildung 5.5: Fehlerraten-Kurven und ROC-Kurve des Systems

banten repräsentiert werden sollte. Hieraus wurde nun wieder wie in 5.1.1 für jede Bewertungsstelle $\alpha \in [0,1]$ jeweils der *minimale*, *maximale* und *mittlere* Wert zusammen mit der Standardabweichung über alle 70 Probanten berechnet. Gleiches wurde für die Test-FRR (mittleres Schaubild) und die Test-FAR (unten) durchgeführt.

Wir erkennen auch für *verschiedene* Probanten wieder recht starke Variabilitäten in allen drei Fehlerraten, die sich hier vor allem durch die Breite der Standardabweichung um die Mittelkurve zeigen. Die drei Mittelkurven selbst lassen sich nach obiger Konstruktion mit einigem Recht als „charakteristische“ Fehlerratenkurven des Verifikationssystems auffassen. Für die Beurteilung der Minimums- und Maximums-Kurven sei daran erinnert, daß diese in jeder Bewertungsstelle durch Minimum- bzw. Maximum-Bildung über sämtliche Probantenkurven entstanden sind. Es handelt sich also nicht etwa um die Kurven besonders leistungsstarker bzw. -schwacher Probantenmodelle.

5.1.3 Gesamtleistung

Für die Gesamtleistung des Systems wurden aus den bereits in 5.1.2 zugrundegelegten Mittel-Fehlerkurven für die Trainings-FRR, die Test-FRR und die Test-FAR drei Gesamt-Fehlerkurven durch Mittelwertbildung berechnet (Abb. 5.5, links).

Der Trainingsfehler auf Eigenunterschriften (gepunktete blaue Kurve) liegt durchgehend unterhalb des entsprechenden Testfehlers (durchgezogene grüne Kurve). Im Threshold-Bereich zwischen 0.4 und 0.5 sind die Unterschiede erheblich und betragen dort häufig 0.5 und mehr. Dies ist, wie in 2.5.4.4 bemerkt wurde, ein Indiz dafür, daß das hier präsentierte System zu Overfitting neigt.

Die Test-FRR besitzt lediglich bei Threshold 0 den Wert 0 und erreicht den Wert 0.01 bei Threshold 0.137. Bei Threshold 0.575 wird der Wert 0.99 angenommen, keine zum Testen verwendete Eigenunterschrift erhielt eine Bewertung größer als 0.772.

Alle Trainingsunterschriften erhalten Bewertungen von mindestens 0.135, die Fehlerrate von 0.001 wird bei Threshold 0.158 erreicht, die Rate 0.01 bei Threshold 0.269, die Rate 0.99 bei Threshold 0.685 und die Rate 0.999 bei Threshold 0.763. Die höchste Bewertung einer Eigenunterschrift beträgt 0.837.

Bei einem Threshold von 0.072 werden 50% aller zum Testen eingesetzten Fremdunterschriften korrekt als Fälschungen eingestuft, bei Threshold 0.145 sind es 75%, bei Threshold 0.211 90%, bei Threshold 0.336 99% und bei Threshold 0.397 99.9%. Keine Fremdunterschrift bekam eine Bewertung von 0.556 oder mehr.

Die EER für den Testfehler liegt bei Threshold 0.252 und beträgt 5.04%. An der Stelle der 1%-Fehlerrate für Fremdunterschriften (0.336) beträgt die Fehlerrate für Eigenunterschriften 21.1%, bei 0.1% FAR (Threshold 0.397) ist FRR=51.2%. Einen vollständigen Vergleich zwischen den Testfehlerraten bietet die ROC-Kurve (Abb. 5.5, rechts). Der AUC-Wert beträgt 0.0146. In Anhang B ist die ROC-Kurve in Tabellenform aufgeführt.

5.2 Laufzeitverhalten

Im Folgenden werden Angaben über das gemessene Laufzeitverhalten der Implementierung des Modellierungs- und des Verifikationsprozesses gemacht. Das System wurde in der Programmiersprache Java™ (`java.sun.com`) unter Java Development Kit (JDK) 1.4.1 implementiert, mit Optimierung übersetzt und bei abgeschalteten Assertions unter Java Runtime Environment (JRE) 1.4.1 ausgeführt. Der für die Messungen eingesetzte Computer war ein handelsüblicher PC mit 1.1GHz AMD Athlon™ Prozessor (`www.amd.com`) und 256MB DDR-RAM Hauptspeicher. Das Betriebssystem war Debian GNU/Linux 3.0 (`www.debian.org`) mit Linux-Kernel 2.4.22 (`www.kernel.org`). In den Messungen sind keine Festplattenzugriffe zum Laden und Speichern von Daten inbegriffen. Der Rechner war während der Experimente ansonsten unausgelastet.

5.2.1 Laufzeit des Enrollments

Zur Abschätzung der Laufzeiten des Enrollment-Prozesses wurde für 10 zufällig ausgewählte Probanten je ein Modell generiert. Jede dieser Modellierungen geschah mit 10 zufällig ausgewählten Unterschriften des jeweiligen Probanten. Die die Laufzeit des Enrollments dominierenden Systemparameter sind die *Populationsgröße* N_{GA} und die *Generationsanzahl* G_{GA} des Genetischen Algorithmus zur Prädiktorgenerierung, sowie die *Analysebreite* β_{Part} des Partitionierungsalgorithmus für den Alignmentgraphen (vgl. 4.2.4).

Tab. 5.1 gibt Minimum, Maximum, Mittelwert und Standardabweichung der gemessenen Laufzeiten in Sekunden an. Grob gesagt schwankt die Dauer zur Erzeugung eines Modells zwischen etwa einer halben Stunde und knapp zwei Stunden, wobei sich im Mittel eine Laufzeit von rd. einer Stunde ergibt.

	min	*max*	μ	σ
Dauer (Sek)	2086	6465	3950	1524

Tabelle 5.1: Statistik zur Laufzeit des Enrollments

5.2.2 Laufzeit der Verifikation

Der für die Laufzeit der Verifikation dominierende System-Parameter ist die *Rotationsphasen-Granularität* ω_{Rot} für den Rotationsausgleich (vgl. 4.5, (4.29)). Es wurde je ein Modell von jedem der 70 Probanten gegen 10 zufällig aus der Gesamtheit sämtlicher Signaturen gewählter Unterschriften verifiziert. Gemäß Tab. 5.2 beträgt die mittlere Verifikationsdauer rd. 10s mit wenigen Sekunden Streuung nach oben und unten. Für lediglich 13 der 700 Unterschriften wurde eine Zeit von mehr als 20s benötigt. Das gemessene Maximum von über 32s besitzt somit keine hohe Repräsentativität.

	min	*max*	μ	σ
Dauer (Sek)	2.49	32.48	10.34	4.31

Tabelle 5.2: Statistik zur Laufzeit des Verifikationsprozesses

5.3 Speicherplatzbedarf

Der Java-Interpreter in Version 1.4.1 bietet mit der Option `-Xrunhprof:heap=all` eine Möglichkeit des *Speicherplatz-Profilings*, um Informationen über den Arbeitsspeicherbedarf des ausgeführten Prozesses zu erhalten. Es läßt sich hierdurch eine ungefähre obere Schranke für die maximal zu einem Zeitpunkt allozierte Speichermenge im Verlauf der Programmausführung ermitteln. Die Ergebnisse sind möglicherweise nicht allzu genau, und sollten daher lediglich als grobe Richtschnur dienen, den Ressourcenanspruch des Systems zu beurteilen. In den folgenden Statistiken wird die gemessene Speichermenge in Byte angegeben.

5.3.1 Platzbedarf des Enrollments

Es wurde je ein Modell für 9 zufällig ausgewählte Probanten generiert. Jede Modellierung wurde anhand 10 zufällig gewählter Eigenunterschriften des Probanten durchgeführt.

	min	max	μ	σ
Platz (Byte)	2 618 128	7 795 880	5 306 159	1 755 318

Tabelle 5.3: Statistik zum Arbeitsspeicherbedarf des Enrollments

5.3.2 Platzbedarf der Verifikation

Es wurde je ein Modell von jedem der 70 Probanten gegen 10 zufällig aus der Gesamtheit sämtlicher Signaturen gewählter Unterschriften verifiziert.

	min	max	μ	σ
Platz (Byte)	518 672	2 265 376	1 194 803	282 571

Tabelle 5.4: Statistik zum Arbeitsspeicherbedarf des Verifikationsprozesses

Kapitel 6

Fazit und Ausblick

In dieser Arbeit wurde ein neues System zur Unterschriftenverifikation präsentiert. Die zu vergleichenden Unterschriften liegen als Zeitsequenzen von Schreibdruckwerten vor, die von einem speziellen Eingabegerät, dem BiSP-Pen, aufgezeichnet werden. Ein Hauptaspekt lag dabei auf der *adaptiven Segmentierung* einer solchen Unterschriftenzeitreihe, die zusammen mit einer geeigneten Segmentcodierung für ein hohes Maß an Vergleichbarkeit zwischen Signaturproben derselben Person sorgen sollte. Ein weiterer Kernpunkt war die Generierung einer *Modellsequenz* mit dem Ziel, die charakteristische Segmentfolge eines Probanten gemeinsam mit der personenspezifischen Variabilität in jedem Segment zu repräsentieren.

Für das Enrollment werden ausschließlich Eigenunterschriften der zu modellierenden Person verwendet. Dadurch werden die üblicherweise mit Fälschungen als Trainingsdaten einhergehenden Probleme, wie sie verschiedentlich in dieser Arbeit erwähnt wurden, von vornherein ausgeschlossen. Die grundsätzliche Idee war, daß ein spezifisch für eine Person generiertes Modell verglichen mit einer Fremdunterschrift, welche i.a. eine völlig andere Struktur besitzt als die Unterschriften des modellierten Probanten, zu einer niedrigen Ähnlichkeitsbewertung führen wird. Für die Realisierung dessen wurden verschiedene Methoden angewandt. Zu nennen seien das *segmentweise Alignment* sowohl beim Enrollment als auch bei der Verifikation, die ausschließliche Verwendung von *Segmentfeatures mit geringer Streurate* in Abschnitt 4.4.3, oder auch die Abweisung von Unterschriften mit einer überproportionalen *Längenabweichung* in 4.4.4. Über die Mindestanzahl an nötigen Trainingsdaten kann hier keine Aussage getroffen werden. In sämtlichen Tests wurden zur Modellierung stets 10 Unterschriften eingesetzt; eine Menge, die in der praktischen Anwendung gerade noch akzeptabel erscheint. Es ist aber zu erwarten, daß sich für deutlich kleinere Trainingsmengen die Leistungsfähigkeit des Systems stark verschlechtern wird, denn für eine *statistische* Modellierung, wie sie in dieser Arbeit erfolgt ist, bedarf es notwendigerweise einer gewissen Mindestmenge an auswertbaren Daten. Das in 4.3 vorgestellte Modell-Update ermöglicht allerdings prinzipiell den *iterativen* Aufbau von Modellen ausgehend von einem „Grobmodell“, sodaß dieses Problem weniger schwerwiegend erscheint. Die erzeugten Modelle sind relativ kompakt: In allen bisherigen Untersuchungen übertraf ihr Datenvolumen nie eine Größe von 50KB, womit die Speicherung auf Smartcards möglich wird.

In einem umfangreichen, für zusammengenommen ca. 4500 Unterschriften von 70 Probanten durchgeführten Test (Kapitel 5) hatte sich eine ROC-Kurve mit einem AUC-Wert von unter 1.5% und einer EER von rd. 5% auf den Testdaten ergeben. Bei dem kürzlich durchgeführten internationalen Wettbewerb SVC 2004[1], der speziell auf den Vergleich von Verifikationssystemen für die Unterschriftendynamik ausgerichtet war, und an dem sich etwa 30 Entwicklergruppen aus Forschung und Industrie beteiligten, hätte das hier präsentierte System mit den berichteten Leistungsdaten im vorderen Bereich des Teilnehmerfeldes gele-

[1] Auf der Homepage des Wettbewerbes (`www.cs.ust.hk/svc2004`) sind Beispiele für das Datenmaterial, wie es von allen Teams zu verwenden war, sowie die Ergebnisse des Vergleichs in Tabellenform und als ROC-Kurven erhältlich.

gen. Ein solcher Vergleich hinkt zugegebenermaßen, da weder die verwendeten Daten noch die Durchführung der Tests exakt übereinstimmen. Der Autor plant daher, sich mit seinem System an einer zukünftigen Durchführung des Wettbewerbs zu beteiligen.

Die gemessenen Fehlerraten des Systems lassen sich möglicherweise mit überschaubarem Aufwand verringern. In Kapitel 5 war deutlich geworden, daß die erzeugten Modelle jeweils für sich betrachtet zwar häufig eine weitgehende Trennung zwischen Eigen- und Fremdunterschriften herbeiführen können, daß aber die entsprechenden Trennstellen stark modellabhängig sind. Weitere Untersuchungen des Autors lassen diesen vermuten, daß jenes Phänomen der „dynamischen" Trennposition weniger von der Wahl der Trainingsunterschriften als vielmehr vom jeweils für das Modell generierten *Prädiktor* abhängt. Sollte es gelingen, die Trennstelle für alle Modelle weitgehend anzugleichen, oder sie zumindest mit Hilfe der Trainingsdaten adäquat abzuschätzen, so ließe sich allein durch eine solche Maßnahme eine Verbesserung der EER bis auf das Niveau des *Mittelwertes der modellspezifischen EERs* erreichen, welcher 1.94% beträgt. Auch dieser Wert ist wohl weiter reduzierbar, denn es hat sich gezeigt, daß noch recht häufig Eigenunterschriften eine sehr schlechte Verifikationsbewertung erhalten, wohingegen für immerhin 12.5% aller Modelle eine perfekte Trennung (modellspezifische EER = 0) gegeben ist, und für mehr als 30 von hundert Modellen liegen die modellspezifischen EERs bei unter 0.5% (vgl. Tab. B.2 in Anhang B). Bei der Untersuchung einiger Fälle schlecht bewerteter Eigenunterschriften hat sich herausgestellt, daß die Unterschriften gelegentlich entgegen den Vorgaben mit *rotiertem* Stift erzeugt worden waren. Die implementierte Rotationsbeseitigung aus 4.5 findet zwar anscheinend in recht zuverlässiger Weise den richtigen Rotationswinkel auf, die Bewertungen rotationskorrigierter Unterschriften fallen jedoch zumeist relativ niedrig aus. Möglicherweise könnte dies an der zusätzlichen *Stiftneigung* liegen: Selbst wenn diese für die Unterschriften einer Person stets gleich ist, so wird durch sie bei der Rückrotation das Signal jedoch verzerrt. Durch die Verwendung der neuen Version des BiSP-Pens, welcher auch die Stiftneigung erfaßt, sollte sich diese Frage klären lassen.

Zukünftige Entwicklungen hätten die zur Verifikation eingesetzten *Features* zum Thema. Diese waren identisch mit den für das Alignment von Segmentsequenzen verwendeten statistischen Merkmalen gewählt worden, wozu allerdings prinzipiell keine Notwendigkeit besteht. Als reizvolle Idee erscheint es, Featuresets *probantenspezifisch* zu wählen, d.h. nach solchen Vergleichsmerkmalen Ausschau zu halten, die für den zu modellierenden Probanten besonders hohe Bewertungen versprechen, mit der Hoffnung, daß Fremdunterschriften bei der Verifikation dadurch benachteiligt werden. Dies wäre eine konsequente Weiterverfolgung der bisherigen personenspezifischen Modellierung.

Auch die *Segmentierung* verlangt nach weitergehender Untersuchung. Die in 4.2.1 beobachtete Kuriosität, wonach es für die Form der Prädiktionsfehlersequenz beinahe irrelevant zu sein scheint, welchen Wert die Zielfunktion des Genetischen Algorithmus dem Siegerprädiktor gibt, genau wie die Feststellung, daß ein guter Prädiktor auch für Fremdunterschriften gute Prädiktionsergebnisse liefert, hat der Autor bislang überhaupt nicht verstanden. Bei der hohen Bedeutung der Segmentierung für die Leistungsfähigkeit des Gesamtverfahrens lohnt es sich darüber hinaus sicherlich auch, über *nichtlineare* Prädiktoren, oder gar über völlig anders geartete Segmentierungsstrategien nachzudenken.

In dieser Arbeit wurden keine *qualifizierten* Fälschungen für die Tests verwendet, und daher ist nicht endgültig geklärt, in wieweit das präsentierte Verfahren vor allem gegenüber Schriftbildkopien empfindlich ist. Eine interessante Ergänzung eines Verifikationssystems wäre hierfür eine Methode zur *direkten* Erkennung von Unterschriften als Fälschungen. Ein solches System würde, wie zu Beginn von Kapitel 2 bereits angedeutet, eine Unterschriftenprobe ohne Hinzuziehung eines Vergleichsmodells auf allgemeine Fälschungsartefakte hin untersuchen und eine Bewertung für die Vertrauenswürdigkeit der Unterschrift abgeben. Eine naheliegende Realisierung eines solchen Systems wäre ein mit qualifizierten Originalunterschriften und Fälschungen trainiertes *Neuronales Netz* [Bra95]. Die Bewertungen dieses Fälschungsdetektors und des Verifikationssystems könnten dann in eine gemeinsame Gesamtbewertung einfließen.

Schließlich läßt sich ein Verifikationssystem grundsätzlich auch zur Personen*identifikation* einsetzen, indem eine vorgelegte Einzelunterschrift, deren Erzeuger namentlich ermittelt werden soll, mit sämtlichen Probantenmodellen einer Datenbank verglichen wird. Das hier vorgestellte System ist dazu jedoch bereits für mittelgroße Datenbestände unbrauchbar, da eine Verifikation mit aktueller Rechnertechnik mehrere Sekunden dauert (vgl. Kapitel 5). Eine deutliche Beschleunigung läßt sich dadurch erreichen, daß zur Rotationskorrektur eine geringere Anzahl an Phasendrehungen als in 4.5 angegeben durchgeführt wird, ohne daß darunter die Leistungsfähigkeit allzu stark zu leiden hätte. Und sollte es doch noch gelingen, einen Rotationsangleich als direkten Vorverarbeitungsschritt im Sinne der in 4.1.6 geführten Diskussion zu entwickeln, so würde sich im Idealfall eine Laufzeitsteigerung von mehr als zwei Zehnerpotenzen ergeben. Für wirklich umfangreiche Personen-Datenbanken ist aber ein sequenzielles Durchsuchen aller Modelle grundsätzlich kein gangbarer Weg. Hier wäre die Entwicklung einer *Indexierungsmethode* von Interesse, über die es mit sublinearem Aufwand möglich sein sollte, sämtliche potentiellen Eigner einer gegebenen Unterschrift ausfindig zu machen.

Ansonsten steht die Übertragbarkeit der vorgestellten Methode auf andere Erfassungssysteme als den BiSP-Pen als Frage im Raum. Prinzipiell sollten sich die BiSP-Pen-spezifischen Eigenschaften nur auf die *Parametrisierung* des Systems ausgewirkt haben. Der eigentliche funktionale Kern des Verfahrens sollte offen sein für andere Architekturen – evtl. sogar offen für andere Zeitreihen-Anwendungen als die der Unterschriftenverifikation.

Anhang A

Die Implementierung

Die Implementierung des in dieser Arbeit vorgestellten Systems wurde in der Programmiersprache Java™ (java.sun.com) unter Java Development Kit (JDK) 1.4.1 erstellt. Der Umfang des Quellcodes beträgt ca. 6000 Zeilen.

Verwendet wird das System über einen einzigen JAR-Container namens `sigcheck.jar`. Der JAR-Container ist ausführbar, die wichtigsten Funktionen des Systems lassen sich damit wie folgt benutzen:[1]

- Enrollment:

  ```
  java -jar sigcheck.jar enroll NEWMODEL SIGFILE [SIGFILE ...]
  ```

`NEWMODEL`	Ziel-Dateiname des neu zu generierenden Modells
`SIGFILE`	Dateiname einer Trainings-Unterschrift (mehrere erlaubt)

- Verifikation:

  ```
  java -jar sigcheck.jar verify MODELFILE SIGFILE [THRESHOLD]
  ```

`MODELFILE`	Dateiname eines Referenz-Modells
`SIGFILE`	Dateiname einer Query-Unterschrift
`THRESHOLD`	Separations-Threshold (optional; Default=0.0)

- Update (Modell-Erweiterung bzw. -Aktualisierung):

  ```
  java -jar sigcheck.jar update-grow OLDMODEL NEWMODEL SIGFILE
  java -jar sigcheck.jar update-refresh OLDMODEL NEWMODEL SIGFILE
  ```

`OLDMODEL`	Dateiname eines zu überarbeitenden Modells
`NEWMODEL`	Ziel-Dateiname des neu zu generierenden Modells
`SIGFILE`	Dateiname einer Update-Unterschrift

Jede Unterschriften-Datei besteht aus Zeilen von durch Leerzeichen '␣' getrennten Fließkommazahlen, wie in folgendem Beispiel dargestellt:

```
-0.675␣1.187␣0.036↩
```

Dabei repräsentiert jede Zeile ein Sample und jede Zahl eine Signalkomponente.

Neben dem JAR-Container existiert HTML-Dokumentation (generiert durch JavaDoc) für die gesamte System-API. Auf die System-API läßt sich ebenfalls über den JAR-Container

[1] Die hier gemachten Angaben über die Verwendungsweise des Verifikationssystems als JAR-Container gelten für übliche Unix/Linux-Installationen. Die Verwendung unter anderen Betriebssystemen ist prinzipiell möglich, sofern eine Java-Laufzeitumgebung JRE in Version 1.4 oder höher vorliegt. In welcher Weise dort die Verwendung zu erfolgen hat, ist der Dokumentation des jeweiligen Betriebssystems zu entnehmen.

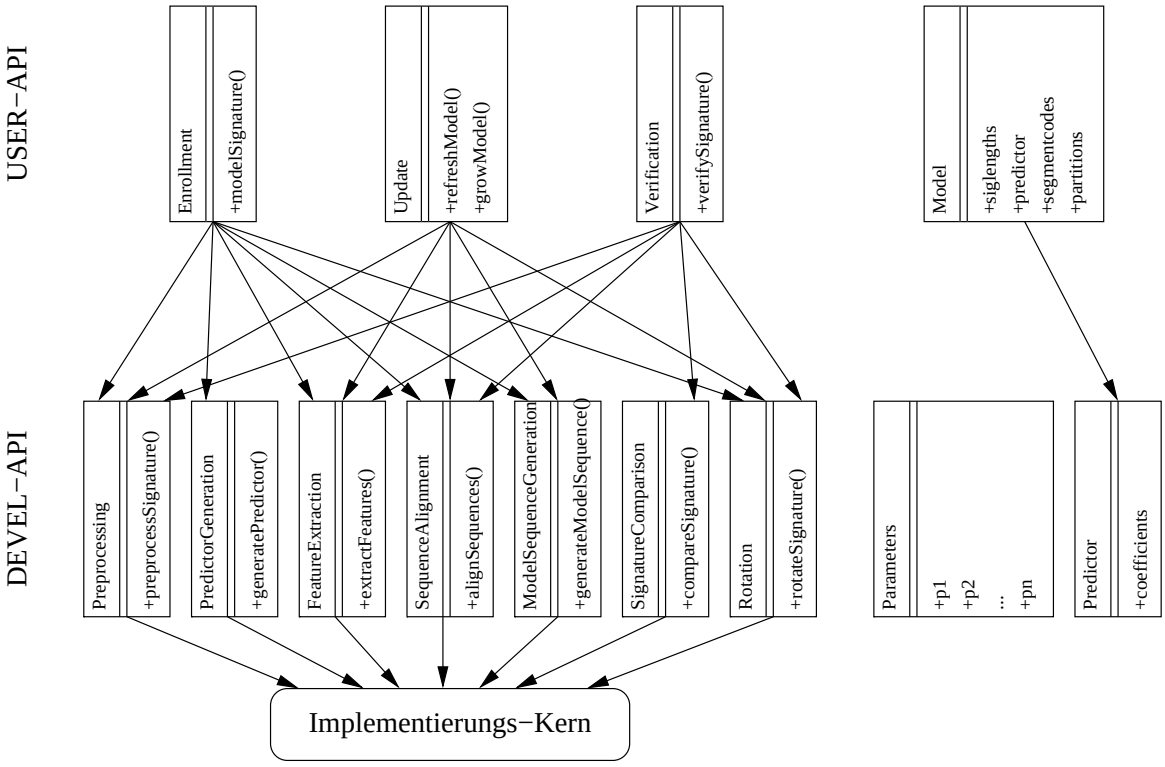

Abbildung A.1: Aufbau der Implementierung

zugreifen. Der Name des zugehörigen Java-Packages ist `sigcheck`. Es wird zwischen einer *„User-API“* und einer *„Developer-API“* unterschieden (vgl. Abb. A.1).

Die *User-API* stellt die wesentliche Anwendungsfunktionalität bereit. Sie setzt sich aus Klassen für das Enrollment (`Enrollment`), das Modell-Update (`Update`) und die Verifikation (`Verification`) zusammen. Eine Datenklasse `Model` repräsentiert Modelle. Eine zusätzliche Hilfsklasse `UtilSignature` (nicht dargestellt) unterstützt das Einlesen von Unterschriften-Dateien. Die zuvor beschriebene Verwendung des JAR-Containers als ausführbares Programm läßt sich als Zugriffsmethode auf die User-API auffassen.

Die *Developer-API* ermöglicht den Zugriff auf die Teilprozesse des Systems und erlaubt deren Parametrisierung in weiten Grenzen. Die Klasse `Preprocessing` dient der Datenvorverarbeitung. Sie wird von mehreren Klassen der User-API verwendet. Mittels `PredictorGeneration` wird ein Prädiktor generiert, der durch die Datenklasse `Predictor` repräsentiert wird. `FeatureExtraction` faßt die Schritte Segmentierung und Segmentcodierung zusammen. Durch `SequenceAlignment` gelingt der Zugriff auf den DTW-Algorithmus. Für Aufbau und Manipulation des Alignmentgraphen bis hin zur Erzeugung der statistischen Modellsequenz ist die Klasse `ModelSequenceGeneration` verantwortlich. Der Vergleich von Query-Unterschrift und Modell erfolgt in `SignatureComparison`. Die Klasse `Rotation` stellt den Prozeßklassen der User-API eine Möglichkeit der manuellen Rotation einer Unterschrift zur Verfügung. `Parameters` enthält sämtliche Systemparameter als Klassenkonstanten, auf die die Defaultkonstruktoren der Klassen der Developer-API zugreifen.

Die Klassen der Developer-API delegieren ihre eigentliche Arbeit an den *Implementierungskern*, der aus rd. 30 Klassen besteht, und auf den kein Zugriff von außerhalb des Packages möglich ist. Der Implementierungskern realisiert die in Kapitel 4 vorgestellten Funktionalitäten des vom Autor dieser Arbeit entwickelten Verifikationssystems. Auf eine nähere Beschreibung der Architektur des Implementierungskerns wird hier aus Platzgründen verzichtet.

Anhang B

Tabellen

Tabelle B.1: **ROC-Kurve des Verifikationssystems.** Die Einträge in dieser Tabelle liegen Abb. 5.5, rechts, auf S. 99, zugrunde. Angegeben sind die jeweils einander zugeordneten Paare aus False-Rejection-Rate (FRR) und False-Acceptance-Rate (FAR).

FRR	*FAR*	*FRR*	*FAR*	*FRR*	*FAR*	*FRR*	*FAR*	*FRR*	*FAR*
0.0000	1.0000	0.0057	0.4504	0.0088	0.3074	0.0129	0.1987	0.0194	0.1319
0.0023	0.8011	0.0058	0.4470	0.0091	0.3040	0.0132	0.1961	0.0199	0.1298
0.0026	0.7806	0.0059	0.4399	0.0092	0.3007	0.0133	0.1935	0.0201	0.1279
0.0029	0.7666	0.0060	0.4364	0.0094	0.2974	0.0134	0.1909	0.0204	0.1259
0.0030	0.7312	0.0061	0.4295	0.0096	0.2873	0.0138	0.1884	0.0208	0.1241
0.0031	0.7136	0.0062	0.4223	0.0098	0.2842	0.0140	0.1831	0.0210	0.1222
0.0032	0.6857	0.0064	0.4042	0.0100	0.2777	0.0143	0.1805	0.0213	0.1203
0.0033	0.6778	0.0065	0.3972	0.0101	0.2745	0.0146	0.1781	0.0216	0.1185
0.0034	0.6701	0.0066	0.3936	0.0102	0.2713	0.0147	0.1756	0.0219	0.1166
0.0036	0.6485	0.0067	0.3830	0.0103	0.2682	0.0149	0.1730	0.0221	0.1148
0.0037	0.6192	0.0069	0.3795	0.0106	0.2589	0.0151	0.1707	0.0224	0.1131
0.0040	0.6095	0.0071	0.3761	0.0108	0.2497	0.0152	0.1684	0.0228	0.1114
0.0041	0.5806	0.0072	0.3727	0.0109	0.2468	0.0154	0.1661	0.0230	0.1097
0.0042	0.5490	0.0073	0.3692	0.0112	0.2408	0.0157	0.1615	0.0236	0.1080
0.0043	0.5424	0.0074	0.3657	0.0114	0.2379	0.0161	0.1593	0.0239	0.1065
0.0045	0.5357	0.0075	0.3622	0.0115	0.2350	0.0162	0.1570	0.0241	0.1048
0.0046	0.5293	0.0076	0.3588	0.0116	0.2292	0.0165	0.1547	0.0243	0.1031
0.0048	0.5058	0.0077	0.3554	0.0117	0.2263	0.0167	0.1525	0.0246	0.1016
0.0049	0.5023	0.0078	0.3451	0.0118	0.2235	0.0171	0.1482	0.0252	0.1000
0.0050	0.4990	0.0081	0.3416	0.0119	0.2207	0.0179	0.1460	0.0256	0.0985
0.0051	0.4888	0.0082	0.3348	0.0121	0.2179	0.0180	0.1439	0.0262	0.0969
0.0052	0.4819	0.0083	0.3278	0.0122	0.2123	0.0183	0.1418	0.0268	0.0954
0.0053	0.4750	0.0084	0.3244	0.0123	0.2095	0.0184	0.1399	0.0271	0.0940
0.0054	0.4646	0.0085	0.3209	0.0125	0.2040	0.0188	0.1378	0.0278	0.0925
0.0055	0.4611	0.0086	0.3142	0.0126	0.2013	0.0193	0.1339	0.0282	0.0911

ROC-Kurve des Verifikationssystems (Fortsetzung nächste Seite)

FRR	*FAR*	*FRR*	*FAR*	*FRR*	*FAR*	*FRR*	*FAR*	*FRR*	*FAR*
0.0286	0.0896	0.0553	0.0460	0.1090	0.0225	0.2180	0.0095	0.4078	0.0025
0.0291	0.0882	0.0565	0.0452	0.1109	0.0221	0.2212	0.0092	0.4137	0.0024
0.0293	0.0867	0.0574	0.0444	0.1125	0.0217	0.2251	0.0090	0.4199	0.0023
0.0296	0.0853	0.0580	0.0437	0.1152	0.0213	0.2297	0.0088	0.4259	0.0022
0.0305	0.0839	0.0586	0.0429	0.1166	0.0209	0.2335	0.0085	0.4326	0.0021
0.0307	0.0826	0.0598	0.0422	0.1181	0.0205	0.2374	0.0083	0.4378	0.0020
0.0310	0.0812	0.0608	0.0414	0.1199	0.0201	0.2414	0.0081	0.4421	0.0019
0.0312	0.0800	0.0623	0.0407	0.1214	0.0198	0.2452	0.0078	0.4478	0.0018
0.0318	0.0786	0.0632	0.0400	0.1241	0.0194	0.2495	0.0077	0.4583	0.0017
0.0324	0.0773	0.0639	0.0393	0.1269	0.0190	0.2531	0.0074	0.4636	0.0016
0.0329	0.0761	0.0652	0.0386	0.1285	0.0186	0.2560	0.0072	0.4692	0.0015
0.0333	0.0748	0.0669	0.0379	0.1307	0.0183	0.2596	0.0070	0.4805	0.0014
0.0338	0.0736	0.0678	0.0373	0.1329	0.0179	0.2640	0.0068	0.4856	0.0013
0.0343	0.0725	0.0690	0.0367	0.1345	0.0176	0.2688	0.0066	0.4963	0.0012
0.0346	0.0713	0.0700	0.0361	0.1372	0.0173	0.2736	0.0063	0.5010	0.0011
0.0351	0.0702	0.0712	0.0355	0.1393	0.0169	0.2782	0.0061	0.5120	0.0010
0.0360	0.0690	0.0722	0.0348	0.1418	0.0166	0.2826	0.0059	0.5280	0.0009
0.0366	0.0679	0.0731	0.0342	0.1448	0.0162	0.2869	0.0058	0.5392	0.0008
0.0371	0.0668	0.0740	0.0336	0.1471	0.0159	0.2918	0.0056	0.5556	0.0007
0.0377	0.0657	0.0752	0.0330	0.1493	0.0156	0.2966	0.0054	0.5737	0.0006
0.0384	0.0647	0.0762	0.0324	0.1526	0.0152	0.3005	0.0052	0.5911	0.0005
0.0393	0.0636	0.0775	0.0318	0.1562	0.0149	0.3062	0.0050	0.6128	0.0004
0.0398	0.0626	0.0790	0.0313	0.1588	0.0146	0.3113	0.0049	0.6342	0.0003
0.0405	0.0615	0.0801	0.0307	0.1615	0.0143	0.3164	0.0047	0.6656	0.0002
0.0415	0.0605	0.0815	0.0302	0.1647	0.0140	0.3222	0.0045	0.7144	0.0001
0.0423	0.0595	0.0830	0.0296	0.1668	0.0137	0.3277	0.0044	0.7880	0.0000
0.0428	0.0585	0.0844	0.0291	0.1698	0.0134	0.3321	0.0042	1.0000	0.0000
0.0437	0.0575	0.0859	0.0286	0.1727	0.0131	0.3368	0.0040		
0.0442	0.0566	0.0873	0.0281	0.1754	0.0128	0.3424	0.0039		
0.0452	0.0556	0.0892	0.0276	0.1787	0.0125	0.3469	0.0038		
0.0462	0.0547	0.0903	0.0270	0.1822	0.0122	0.3522	0.0037		
0.0466	0.0538	0.0922	0.0266	0.1858	0.0119	0.3576	0.0035		
0.0471	0.0529	0.0939	0.0261	0.1886	0.0116	0.3629	0.0034		
0.0481	0.0520	0.0956	0.0256	0.1924	0.0113	0.3680	0.0033		
0.0497	0.0511	0.0975	0.0252	0.1950	0.0110	0.3728	0.0032		
0.0505	0.0502	0.0993	0.0247	0.1986	0.0108	0.3784	0.0030		
0.0510	0.0494	0.1011	0.0242	0.2028	0.0105	0.3837	0.0029		
0.0519	0.0485	0.1033	0.0238	0.2068	0.0103	0.3906	0.0028		
0.0528	0.0477	0.1049	0.0234	0.2109	0.0100	0.3967	0.0027		
0.0540	0.0468	0.1070	0.0230	0.2147	0.0098	0.4014	0.0026		

ROC-Kurve des Verifikationssystems (Schluß)

Tabelle B.2: **Modellspezifische Equal-Error-Rates (EER).** Für alle Probanten, von denen mehr als 10 Unterschriftenproben vorlagen (51 von 70; vgl. Kapitel 3), wurde die modellspezifische EER für alle fünf in Kapitel 5 erzeugten Modelle berechnet. Die dreistelligen Zahlen unter *„Probant“* stellen die in den Tests verwendeten IDs der Probanten dar. *„EER n“* gibt die EER zum n-ten Modell des jeweiligen Probanten an.

Probant	*EER 0*	*EER 1*	*EER 2*	*EER 3*	*EER 4*
001	0.0029	0.0009	0.0215	0.0017	0.0000
002	0.0401	0.0690	0.0286	0.1330	0.1283
003	0.0000	0.0021	0.0000	0.0001	0.0333
005	0.1034	0.0690	0.2414	0.0345	0.0345
007	0.0779	0.0690	0.2069	0.1724	0.1034
008	0.0079	0.0000	0.0257	0.0012	0.0040
009	0.0667	0.0984	0.0667	0.1374	0.1000
010	0.0052	0.0047	0.0317	0.0000	0.0667
012	0.0026	0.0000	0.0000	0.0000	0.0026
014	0.0012	0.0265	0.0000	0.0333	0.0005
016	0.0000	0.0000	0.0000	0.0000	0.0000
101	0.0444	0.0333	0.0133	0.0667	0.0369
102	0.0058	0.0124	0.0093	0.0006	0.0173
103	0.0759	0.1266	0.1579	0.1963	0.1139
104	0.1202	0.1778	0.1451	0.1667	0.1113
105	0.0003	0.0138	0.0090	0.0038	0.0059
106	0.1111	0.1382	0.1235	0.1531	0.1358
107	0.0040	0.0000	0.0111	0.0003	0.0000
108	0.0444	0.0627	0.0436	0.0444	0.0556
109	0.0222	0.0333	0.0222	0.0222	0.0222
110	0.0444	0.0467	0.0642	0.0333	0.0404
111	0.0337	0.0213	0.0159	0.0225	0.0112
113	0.0222	0.0222	0.0444	0.0333	0.0111
114	0.0111	0.0111	0.0111	0.0088	0.0096
115	0.0889	0.1111	0.1013	0.0889	0.1000
116	0.0510	0.1065	0.0329	0.0721	0.0556
117	0.0111	0.0000	0.0111	0.0000	0.0008
118	0.0263	0.0095	0.0250	0.0057	0.0191
119	0.3427	0.3333	0.3000	0.3600	0.3664
120	0.0039	0.0114	0.0111	0.0062	0.0222
121	0.0112	0.0201	0.0007	0.0308	0.0046
122	0.0151	0.0125	0.0128	0.0125	0.0250
123	0.0017	0.0000	0.0000	0.0045	0.0000
124	0.0243	0.0250	0.0375	0.0125	0.0187
125	0.0000	0.0000	0.0000	0.0045	0.0000

Modellspezifische EERs (Fortsetzung nächste Seite)

Probant	*EER 0*	*EER 1*	*EER 2*	*EER 3*	*EER 4*
126	0.0143	0.0429	0.0081	0.0273	0.0286
127	0.0219	0.0111	0.0050	0.0031	0.0111
128	0.0111	0.0167	0.0111	0.0111	0.0111
129	0.0004	0.0000	0.0018	0.0078	0.0111
130	0.0009	0.0005	0.0004	0.0000	0.0000
131	0.0038	0.0444	0.0296	0.0124	0.0017
132	0.0222	0.0111	0.0018	0.0111	0.0222
133	0.0667	0.0599	0.0794	0.0444	0.0556
135	0.0556	0.0245	0.0398	0.0667	0.0333
136	0.0045	0.0143	0.0001	0.0001	0.0005
137	0.0016	0.0355	0.0006	0.0111	0.0260
139	0.0000	0.0042	0.0002	0.0035	0.0070
140	0.0889	0.0444	0.0444	0.1111	0.0222
141	0.0112	0.0022	0.0112	0.0011	0.0112
142	0.0000	0.0000	0.0019	0.0000	0.0025
143	0.0051	0.0125	0.0125	0.0125	0.0125

Modellspezifische EERs (Schluß)

Literaturverzeichnis

[BIS] Hompage des BiSP-Projektes: `www.bisp-regensburg.de` (09.09.2004).

[BLH99] Brause, R., Langsdorf, T. und Hepp, T.: *Neural Data Mining for Credit Card Fraud Detection.* In: *IEEE Int. Conf. on Tools with Art. Intell. ICTAI-99*, Seiten 103–106, 1999.

[Bra95] Brause, R.: *Neuronale Netze (2. Aufl.).* Teubner, 1995.

[DH73] Duda, R. und Hart, P.: *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[Eng93] Engesser, H. (Herausgeber): *Duden Informatik (2. Aufl.).* Dudenverlag, 1993.

[Fel68] Feller, W.: *An Introduction to Probability Theory and Its Applications, Vol. 1 (3rd Ed.).* John Wiley & Sons, 1968.

[FS97] Freund, Y. und Schapire, R.: *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of Computer and System Sciences, 55(1):119–139, 1997.

[Gre02] Grell, D.: *Kurven kriegen – Entzerrervorverstärker für Plattenspieler mit Magnetsystem zum Selbstbau.* C't Magazin für Computertechnik, Heft 13, Seite 224ff, 2002.

[Heu91] Heuser, H.: *Lehrbuch der Analysis, Teil 1 (9. Aufl.).* Teubner, 1991.

[HKS03] Hook, C., Kempf, J. und Scharfenberg, G.: *New Pen Device for Biometrical 3D Pressure Analysis of Handwritten Characters, Words and Signatures.* In: *ACM SIGMM 2003 Multimedia: Biometrics Methods and Application Workshop*, Seiten 38–44, Berkley, USA, 2003.

[HKS04] Hook, C., Kempf, J. und Scharfenberg, G.: *A Novel Digitizing Pen for the Analysis of Handwriting in Biometrics.* In: *Biometric Authentication. ECCV 2004 Int. Workshop*, Prague, 2004.

[HQ95] Heise, W. und Quattrocci, P.: *Informations- und Codierungstheorie (3. Aufl.).* Springer-Verlag, 1995.

[JGC01] Jain, A., Griess, F. und Connel, S.: *On-Line Signature Verification.* Technischer Bericht, Department of Computer Science and Engineering, Michigan State University, 2001.

[Knu73] Knuth, D.: *The Art of Computer Programming, Vol. 3: Sorting and Searching.* Addison Wesley, 1973.

[KR88] Kernighan, B. und Ritchie, D.: *The C Programming Language (2nd Ed.).* Prentice Hall, 1988.

[LP94] Leclerc, F. und Plamondon, R.: *Automatic Signature Verification: The State of the Art—1989–1993.* International Journal of Pattern Recognition and Artificial Intelligence, 8(3):643–660, 1994.

[Mic82] Michel, L.: *Gerichtliche Schriftvergleichung: eine Einführung in Grundlagen, Methoden und Praxis.* de Gruyter, 1982.

[Mil91] Miller, G.: *The Science of Words.* Scientific American Library, 1991.

[MRMK02] Mautner, P., Rohlík, O., Matoušek, V. und Kempf, J.: *Signature Verification Using ART-2 Neural Network.* In: *Proceedings of ICONIP'02 Conference*, Singapore, 2002.

[OW93] Ottmann, T. und Widmayer, P.: *Algorithmen und Datenstrukturen (2. Aufl.).* B.I. Wissenschaftsverlag, 1993.

[Pev00] Pevzner, P.: *Computational Molecular Biology.* MIT Press, 2000.

[Roh03] Rohlík, O.: *Handwritten Text Analysis.* Thesis, University of West Bohemia in Pilsen, Faculty of Applied Science – Department of Computer Science and Engineering, 2003.

[Say96] Sayood, K.: *Introduction to Data Compression.* Morgan Kaufmann Publishers, 1996.

[Sch93] Schwarz, H.: *Numerische Mathematik (3. Aufl.).* Teubner, 1993.

[Sch96] Schneier, B.: *Applied Cryptography (2nd Ed.).* John Wiley & Sons, 1996.

[Sch01] Schnitger, G: *Algorithmisches Lernen*, 2001. Skript zur Vorlesung im Sommersemester 2001, Johann-Wolfgang-Goethe-Universität Frankfurt/Main.

[Sch04] Schüler, P.: *„Friedrich Wilhelm" digital.* C't Magazin für Computertechnik, Heft 02, Seiten 172–175, 2004.

[ŠK03] Šoule, M. und Kempf, J.: *Handwritten Text Analysis through Sound. A new Device for Handwriting Analysis.* In: *Proceedings IWSSIP*, Seiten 254–257, Prague, 2003.

[SL01] Schmidt, C. und Lenz, J.: *Unterschriftenerkennung.* In: Behrens, M. und Roth, R. (Herausgeber): *Biometrische Identifikation – Grundlagen, Verfahren, Perspektiven*, Kapitel 10. Vieweg, 2001.

[Smi99] Smith, S.: *The Scientist and Engineer's Guide to Digital Signal Processing (2nd Ed.).* California Technical Publishing, 1999.

[Stö93] Stöcker, H. (Herausgeber): *Taschenbuch mathematischer Formeln und moderner Methoden (2. Aufl.).* Verlag Harri Deutsch, 1993.

[Str02] Strömer, T.: *Online-Recht (3. Aufl.).* dpunkt.verlag, 2002.

[SW90] Storch, U. und Wiebe, H.: *Lehrbuch der Mathematik, Band II: Lineare Algebra.* B.I. Wissenschaftsverlag, 1990.