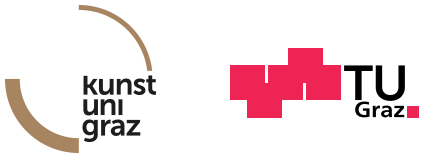


Masterarbeit

Der automatische Registrant

Erschließung von Score-Following zur Setzersteuerung von Orgeln

Markus Maier, BSc
(0873105)

Universität für Musik und darstellende Kunst Graz

Institut für Elektronische Musik und Akustik
Institut für Kirchenmusik und Orgel

Betreuer: Univ.-Prof. Dr. Gerhard Eckel, Mag. Martin Rumori

Studienrichtung: Masterstudium Elektrotechnik-Toningenieur (V 066 413)

Graz, Februar 2017

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....

(Unterschrift)

Kurzfassung

Score-Following ist ein Verfahren der Computermusik, um in Echtzeit maschinell die aktuell ausgeführte Position in einer Partitur zu erfassen. Es wird im Kontext der Live-Elektronik entwickelt und dient dort vor allem der automatisierten Synchronisation von Zuspielungen und Parametersteuerungen mit zeitlich flexibel aufgeführten Kompositionsteilen, zum Beispiel akustischen Soloinstrumenten. Die Spieldaten dieser Teile werden per MIDI oder als Audiosignal an den Score-Following-Algorithmus übermittelt.

Gegenstand dieser Arbeit ist die Anwendung von Score-Following bei Orgelmusik, etwa zur automatischen Steuerung der Registrierung. Dabei soll untersucht werden, inwieweit sich das Verfahren mit Standardrepertoire und damit verbundenen herkömmlichen Spieltechniken einsetzen lässt; die Anwendung ist also nicht auf den Bereich der Neuen Musik beschränkt. Das Projekt wird in Zusammenarbeit mit dem Institut für Kirchenmusik und Orgel durchgeführt, an dem sowohl geeignete Instrumente als auch die musikalische Expertise zur Verfügung stehen.

Ziel der Arbeit ist die Entwicklung eines Laborsystems, mit dem sich spezifische musikalische, technische sowie ergonomische Anforderungen und der gegenwärtige Entwicklungsstand des Score-Following gegenüberstellen und praktisch evaluieren lassen. Dazu müssen die Anforderungen der Anwendung hinsichtlich der wichtigsten Parameter, wie etwa der zeitlichen Granularität, analysiert werden. Davon ausgehend soll ein Workflow formuliert werden, der die prototypische praktische Anwendung des Laborsystems mit dem Instrument beschreibt und Bereiche wie Partitureingabe, die akustische oder elektronische Datenübermittlung und musikalisch-technische Grenzen der Anwendung umfasst.

Abstract

Score following is a procedure in computer music to automatically detect the currently performed position in a score in real-time. It is developed in context of live electronics where it is mainly used to automatically synchronize playbacks and control parameters with temporally flexible parts of compositions, e.g. acoustic solo instruments. The performance data of such parts is transmitted to the score following algorithm via MIDI or audio signal.

The subject of this thesis is the application of score following to organ music, for instance to automatically control the registration. In the process it is investigated to what extent the procedure can be employed in the context of standard repertoire and associated conventional performance techniques, thus the use is not limited to the area of contemporary music. The project is conducted in cooperation with the Institute for Church Music and Organ where both suitable instruments and the musical expertise are available.

The thesis aims at developing a laboratory system as a means to compare and evaluate specific musical, technical and ergonomic requirements as well as the current state of research and development regarding score following. To that end the application requirements concerning the most important parameters like, e.g., temporal granularity have to be analyzed. Based on that, a workflow shall be formulated which describes the prototypical hands-on application of the laboratory system and covers topics like score input, the acoustic or electronic transmission of data or the musical and technical limitations regarding the system's application.

Inhaltsverzeichnis

Einleitung

»Meine neue Orgel, sie ist ein Orchester!« Diese überlieferte Aussage des Organisten und Komponisten César Franck steht sinnbildlich für die Vielzahl von klanglichen Möglichkeiten, die das Instrument aufgrund seiner komplexen Bauweise bietet. Den Klangfarbenvorrat bilden sogenannte Register, Gruppen von Orgelpfeifen, deren Klänge sich abhängig von verschiedenen Bauformen und Funktionsprinzipien voneinander unterscheiden. Durch Auswahl bzw. Kombination von Registern sind Organistinnen und Organisten somit in der Lage, den Klang des Instruments individuell auf das jeweilige Musikstück abzustimmen. In diesem Zusammenhang werden sowohl der Zuordnungsvorgang von Registerkombinationen zu verschiedenen Passagen des Musikstücks als auch die einzelnen Kombinationen selbst als Registrierungen bezeichnet.

Bei rein mechanischen Orgeln konnten Registrierungen während eines Stückes nur geringfügig geändert werden, da jedes einzelne Register durch manuelle Betätigung des entsprechenden Registerzuges mechanisch zugeschaltet oder entfernt werden musste. Im Laufe der Zeit wurden deshalb pneumatische bzw. elektromagnetische Steuerungsmechanismen entwickelt und es entstanden zunächst festgelegte, gemeinsam schaltbare Registerkombinationen und in weiterer Folge sogenannte Setzereinrichtungen. Bei modernen Orgeln sind diese heute elektrisch bzw. elektronisch ausgeführt und ermöglichen dadurch die Speicherung von vorbereiteten Registrierungen sowie den Wechsel zwischen diesen auf Knopfdruck. Mit neuen Möglichkeiten entwickelt sich jedoch auch die Praxis weiter, sodass heute Registrierungswechsel nicht mehr nur an Satz- oder Phrasenübergängen stattfinden, wo diese in einfachen Fällen auch während der Darbietung von der Organistin bzw. dem Organisten selbst durchgeführt werden können. Vielmehr wird die Registrierung nun auch zur musikalischen Gestaltung in wesentlich feineren Abstufungen eingesetzt, was zu komplexeren Kombinationen und hoher Registrierungsfrequenz

führt. Aus diesem Grund müssen Registrierungswechsel in den meisten Fällen trotz der technischen Möglichkeiten nach wie vor von einer zweiten Person, der Registrantin bzw. dem Registrant, vorgenommen werden.

Genau an diesem Punkt setzt die vorliegende Arbeit an und soll die Erforschung neuer Registrierungsmöglichkeiten am Institut für Kirchenmusik und Orgel durch die Entwicklung eines automatischen Registranten unterstützen. Einerseits kann damit der praktische Umgang mit komplexen Registrierungen erleichtert werden, da alternative Registrierungsvarianten oder Wechsel an komplizierten Stellen auch ohne eine zweite Person erprobt werden können. Andererseits werden durch den Einsatz eines solchen Systems zukünftig noch komplexere Registrierungsmethoden denkbar, die von Hand gar nicht mehr durchführbar wären.

Als erster Schritt in diese Richtung ist es das Ziel der vorliegenden Arbeit, ein Laborsystem zu entwickeln, das in der Lage ist, den Notentext eines Musikstückes während dessen Darbietung mitzuverfolgen und an den vorgesehenen Stellen automatisch Registrierungswechsel auszulösen. Dazu wird zunächst das Forschungsfeld des Score-Following erschlossen, das sich mit der maschinellen Verfolgung einer musikalischen Darbietung beschäftigt (Kapitel 1), und im Zuge dessen ein geeignetes Softwarepaket ausgewählt, das diese Funktion als Herzstück des Laborsystems erfüllen kann. Anschließend sollen Aufbau und Funktionsweise dieses Pakets sowie dessen Verwendungsmöglichkeiten im Hinblick auf die geplante Anwendung untersucht werden (Kapitel 2). Darauf aufbauend wird das Laborsystem einerseits aus bestehenden Hardware- und Softwarekomponenten zusammengestellt, sowie andererseits durch im Rahmen dieser Arbeit entwickelte Programmteile komplettiert (Kapitel 3). Dieses soll neben seiner Funktion als automatische Setzersteuerung auch zur Untersuchung von spezifischen musikalischen und technischen Anforderungen an eine solche, sowie zur Analyse der Performance eines solchen Systems und verschiedener Einflüsse darauf dienen (Kapitel 4). In einem prototypischen Workflow wird abschließend die Verwendung des entwickelten Laborsystems zur automatischen Registrierungssteuerung von der Erstellung der für den Score-Follower lesbaren Partitur bis zur Benutzung des Systems bei der Darbietung beschrieben (Kapitel 5).

1 Score-Following

Zu Beginn dieser Arbeit soll dieses Kapitel als Einführung in das zugrunde liegende Thema Score-Following dienen. Dazu versucht Abschnitt 1.1 den Begriff selbst sowie die möglichen Anwendungsbereiche abzustecken und in Abschnitt 1.2 werden die bisherigen Entwicklungen auf diesem Gebiet zusammengefasst. Abschnitt 1.3 beschreibt anschließend häufig verwendete Basiskonzepte und Abschnitt 1.4 soll einen Überblick über derzeitige Anwendungen und zur Verfügung stehende Softwarepakete geben.

1.1 Definition

Der englische Begriff »score following« (dt. etwa »Partiturverfolgung«) entstand im technisch-informatischen Kontext und steht für das automatische Verfolgen einer Partitur durch Computeralgorithmen während der Darbietung des entsprechenden Musikstücks. Das Ziel dabei ist es, in Echtzeit jeweils jene Position in der Partitur zu bestimmen, die der Darbietung zum aktuellen Zeitpunkt entspricht.

Die Forschung auf diesem Gebiet war oft eng mit Musikkompositionen und deren Aufführung unter Verwendung der entwickelten Technologien verbunden, beispielsweise bei Systemen zur automatischen Begleitung von Musikerinnen und Musikern oder bei Stücken für instrumentale und elektronische Stimmen. Daher ist die Echtzeitfähigkeit ein wichtiger Aspekt des Score-Following. Vermutlich geht auch die Bezeichnung im Ursprung auf Roger Dannenberg zurück, der in seinem Artikel »An on-line algorithm for real-time accompaniment« [Dan84] die Formulierung »following the score« verwendete.

Es existieren aber auch sogenannte Offline-Algorithmen, die zwar ein Audiosignal in Relation zu einer Partitur setzen, aber nicht echtzeitfähig sein müssen, da sie andere Ziele verfolgen. Solche Algorithmen, wie etwa zum zeitlichen Abgleich zwischen verschiedenen

Aufnahmen desselben Musikstücks oder zur Unterstützung der Separation von mehreren Audioquellen durch Kenntnis der Partitur, werden meist nur mit dem Überbegriff »audio to score alignment« bezeichnet.

1.2 Historischer Überblick

Die folgenden Abschnitte sollen nun einen Überblick über die bisherigen Entwicklungen auf dem Gebiet des Score-Following geben. Da diesbezüglich bereits mehrere Texte existieren, z.B. [Jor07, Kapitel 2; Arz07, Abschnitt 2.2; Ace11; Wer14, Abschnitt 1.3], baut die nachfolgende Übersicht auf einer Zusammenschau dieser Texte auf, und erweitert sie mit zusätzlichen Publikationen. Die in den meisten der genannten Texte verwendete Gliederung nach verwendeten Basiskonzepten wird auch hier übernommen.

Um den zeitlichen Entwicklungsverlauf sichtbar zu machen, werden die in diesem Überblick erwähnten Publikationen zunächst in Abbildung 1.1 auf einem Zeitstrahl angeordnet, bevor sie in den nachfolgenden Abschnitten jeweils kurz beschrieben werden.

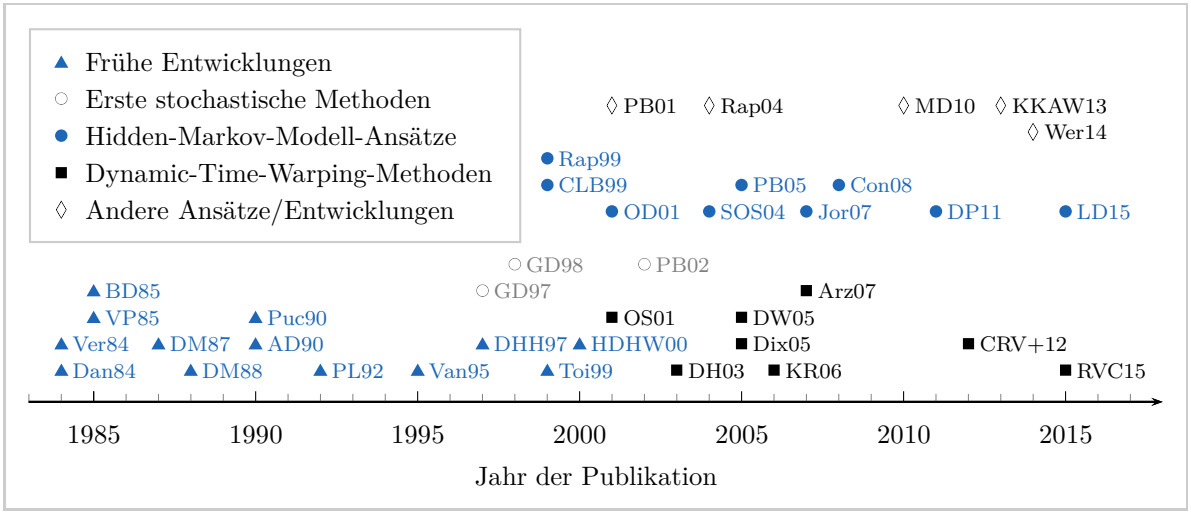


Abbildung 1.1: Darstellung der im historischen Überblick erwähnten Publikationen

1.2.1 Frühe Entwicklungen (ab 1984)

Die ersten bekannten Belege über Forschungen zum Thema Score-Following stammen aus dem Jahr 1984, wo zwei Forscher unabhängig voneinander Vorschläge für die automatische Begleitung von Solisten bei der International Computer Music Conference (ICMC) einreichten.

Roger Dannenberg beschrieb in »An on-line algorithm for real-time accompaniment« [Dan84] den Notentext und dessen Darbietung als zwei Ereignisfolgen und das Score-Following als Problem des Abgleichs derselben. Dabei wurde die Darbietungsfolge mittels Tonhöhendetektion aus dem einstimmigen Audiosignal einer Trompete gewonnen (unterstützt durch Hardware-Sensoren, da eine reine Analyse des Audiosignals zu dieser Zeit noch nicht schnell genug möglich war), und die Folge des Notentextes entsprach der erwarteten Tonhöhenfolge bei korrekter Darbietung. Anschließend ermittelte ein Algorithmus bei jedem neu eingetroffenen Darbietungsereignis die längste übereinstimmende Sequenz der beiden Folgen seit Beginn der Darbietung, wobei jede Erhöhung der Übereinstimmungslänge als Detektion des aktuellen Darbietungsereignisses und damit dessen Position im Notentext interpretiert wurde. Barry Vercoe verfolgte in »The Synthetic Performer in The Context of Live Performance« ähnliche Herangehensweisen: er unterstützte die Tonhöhendetektion mit optischen Sensoren an der verwendeten Flöte und verwendete Pattern-Matching-Techniken zum Abgleich zwischen Notentext und Darbietung [Ver84].

In den darauffolgenden Jahren versuchten beide Forschergruppen ihre Systeme weiterzuentwickeln. Vercoe und Miller Puckette beschäftigten sich mit der Verwendung von Aufzeichnungen aus vergangenen Proben, um den »Synthetic Performer« mit zusätzlichen Informationen über das Musikstück auszustatten und damit die Systemperformance bei zukünftigen Darbietungen zu verbessern [VP85]. Dannenberg arbeitete mit Joshua Bloch an der Verarbeitung von mehrstimmigen Audiosignalen, indem kurz aufeinanderfolgende Ereignisse, wie etwa die Anschläge einzelner Noten eines Akkordes auf einem Keyboard, zu Gruppen zusammengefasst wurden [BD85]. Mit Hirofumi Mukaino verfolgte er diesen Ansatz weiter und erweiterte die Gruppenbildung auf andere zusammengesetzte Ereignisse wie Triller und Glissandi; zudem wurde für unklare Fälle bei der Positionsdetektion die Möglichkeit geschaffen, das eingehende Signal vorübergehend mit

zwei Positionen in der Partitur gleichzeitig zu vergleichen und so die bessere der beiden Optionen feststellen zu können [DM88].

Außerdem betrieb Dannenberg Ende der 80er Jahre zusammen mit Bernard Mont-Reynaud bzw. Paul Allen auch Forschungen im Bereich des Beat-Tracking, um etwa Improvisationen in Jazz-Performances, für die keine expliziten Partituren existieren, über das Tracking der Schläge verfolgen zu können [DM87; AD90].

Einige Jahre nach seiner Zusammenarbeit mit Vercoe griff Puckette das Thema Score-Following im Zeitraum 1990 bis 1992 erneut auf und entwickelte in Kooperation mit Cort Lippe ein System mit einigen Verbesserungen: Es wurde etwa eine sogenannte »Skip List« eingeführt, welche die letzten nicht übereinstimmenden Ereignisse enthielt, um die einzelnen Noten eines Akkordes »einsammeln« zu können, oder ein Mechanismus vorgeschlagen, der Sprünge ohne vorheriges Eintreten eines bestimmten Ereignisses verhindern sollte [Puc90; PL92].

In der zweiten Hälfte der 90er Jahre tauchten dann zwei Versuche auf, das Problem aus einer anderen Richtung anzugehen: Jason Vantomme setzte auf den Rhythmus als primäres Feature für die Detektion, um auszunutzen, dass falsche Töne in diesem Fall irrelevant wären; die Tonhöhen wurden jedoch als Alternative verwendet, falls die Rhythmusdetektion kein ausreichendes Ergebnis lieferte [Van95]. Petri Toiviainen versuchte ein ausschließliches Beat-Tracking, womit auch Improvisationen verfolgt werden konnten, da der verwendete Algorithmus nur das Starttempo und keine sonstigen Informationen über das musikalische Material des Stückes benötigte [Toi99].

Der Großteil der Forscher hielt jedoch an der Tonhöhendetektion fest, so wie auch Peter Desain, Hank Heijink und ihre Kollegen. Sie versuchten, zusätzliche Informationen über die zeitliche Struktur des musikalischen Materials zu verwenden, um dem Algorithmus die Entscheidung zu erleichtern, ob eine von der Partitur abweichende Reihenfolge der eingehenden Ereignisse erlaubt ist, wie etwa bei Anschlägen von einzelnen Akkordtönen, oder ob dies als Fehler in der Darbietung gewertet werden soll, wie beispielsweise bei einem Melodieverlauf [DHH97; HDHW00].

1.2.2 Erste stochastische Methoden (ab 1997)

Am Ende der 90er Jahre versuchten sich Lorin Grubb und Dannenberg an einem neuen Ansatz, der Anwendung von stochastischen Methoden. Dabei wurde die Position in der Partitur mit einer Wahrscheinlichkeitsdichtefunktion dargestellt, die jedes Mal aktualisiert wurde, wenn ein neues Ereignis eintraf. Die Analyse der Ereignisse basierte auf einer Beobachtungsverteilung, mit der modelliert werden konnte, mit welcher Wahrscheinlichkeit die Tonhöhe des eintreffenden Ereignisses der in der Partitur eingetragenen, erwarteten Tonhöhe entspricht [GD97; GD98].

Auch wenn sich die meisten Forschergruppen ab dieser Zeit ausschließlich mit stochastischen Herangehensweisen beschäftigten, hielten einzelne zumindest teilweise noch an den bisher verwendeten Methoden fest. Bryan Pardo und William Birmingham versuchten etwa, den bisher üblichen Vergleich von symbolischen Ereignisrepräsentationen beizubehalten, strebten aber eine Leistungsverbesserung des Tonhöhendetektors mit stochastischen Methoden an. Dazu statteten sie den Detektor mit zuvor festgestellten Informationen darüber aus, wie häufig bestimmte Tonhöhen abhängig von der jeweiligen Stimmlage vorkommen und welche Fehler bei der Audioanalyse durch den Tonhöhendetektor wahrscheinlich auftreten würden [PB02]. Außerdem entwickelten sie den ersten Score-Follower für Lead-Sheets [PB01].

1.2.3 Hidden-Markov-Modell-Ansätze (ab 1999)

Ab 1999 hielt die Verwendung von Hidden-Markov-Modellen (für eine Einführung in das Thema siehe Abschnitt 1.3.1) Einzug in das Forschungsgebiet des Score-Following. Eine Gruppe um Pedro Cano verwendete in ihrem Algorithmus sechs Features zur Analyse des Audiosignals, darunter die Grundfrequenzen der Töne und die Signalenergie, sowie ein dreigeteiltes Hidden-Markov-Modell: Der erste Teil modellierte Noten mit drei Zuständen für Ansatz, stationäre Phase und Ausklang, der zweite Teil war ein einzelner Zustand für Pausen, und der dritte Teil stellte Klänge ohne definierte Tonhöhe mit ebenfalls drei Zuständen dar. Zur Schätzung jener Position im Notentext, die mit der höchsten Wahrscheinlichkeit der aktuellen Stelle der Darbietung entspricht, wurde

der Viterbi-Algorithmus (für eine Kurzbeschreibung siehe Abschnitt 1.3.2) verwendet [CLB99].

Auch Christopher Raphael beschäftigte sich zu dieser Zeit mit diesem Thema und benutzte dabei ebenfalls verschiedene Zustandsgruppen zur Modellierung von unterschiedlichen Klängen wie etwa langen bzw. kurzen Noten und Pausen. Außerdem verwendete er bei der Detektion mehrere Features, wie die Amplitude des aktuellen Audiosignals oder aus einer Fourier-Transformation des Signals ermittelte Frequenzschätzungen [Rap99].

Nicola Orio und François Déchelle formulierten ihren Ansatz als »Hidden-Markov-Modell auf zwei Ebenen«, wobei in der oberen Ebene jedes Ereignis der Partitur jeweils durch einen Zustand abgebildet wurde, und jeder dieser Zustände wiederum aus mehreren Zuständen der unteren Ebene bestand, die das zugehörige Audiosignal modellierten. Außerdem enthielt das Modell sogenannte Geisterzustände, um auch mit fehlerhaften, hinzugefügten oder übersprungenen Ereignissen umgehen zu können [OD01].

Im Jahr 2004 tat sich Diemo Schwarz mit Orio und anderen zusammen, um Orios Entwicklungen aufzunehmen und einen Score-Follower zu entwickeln, der MIDI-Daten als Eingabewerte verwendete und die entsprechenden Klänge intern mittels Hüllkurven modellierte. Sie entschieden sich für diese Variante, da das System mit polyphonen Inputs arbeiten sollte, sie die Verarbeitung von polyphonen Audiosignalen zu dieser Zeit aber als zu schwierig erachteten [SOS04].

Auch Pardo und Birmingham nahmen sich schließlich der Hidden-Markov-Modelle an, und versuchten, auch von der Partitur abweichende Verläufe eines Musikstücks durch zusätzliche Zustandsübergänge im Modell zu unterstützen, um beispielsweise das Auslassen von Wiederholungen zu ermöglichen [PB05].

Im Jahr 2007 verfasste Anna Jordanous ihre Masterarbeit mit dem Titel »Score Following: An Artificially Intelligent Musical Accompanist«, in der sie ein automatisches Begleitungssystem in mehreren Varianten basierend auf Hidden-Markov-Modellen und Tonhöhendetektion entwickelte [Jor07].

Ebenfalls 2007 entstand am IRCAM[1] das bisher wohl umfangreichste System für Score-Following und automatische Begleitung: Antescofo. Es arbeitet auf Basis von Hidden-Hybrid-Markov/Semi-Markov-Modellen und unterstützt zur Detektion neben der Tonhöhe auch weitere Audiofeatures sowie MIDI-Daten [Con08]. Für weitere Informationen zu Antescofo, das bis heute weiterentwickelt wird, siehe Abschnitt 1.4.1 sowie Kapitel 2.

Einige Zeit später entwickelte Pardo zusammen mit Zhiyao Duan das Computersystem »Soundprism«, welches in der Lage war, mit Unterstützung eines Score-Followers mehrere Audioquellen in Echtzeit zu separieren. Der darin integrierte Score-Follower basierte auf einem Hidden-Markov-Modell, verwendete die Tonhöhen als Feature für das Tracking, und die Position im Notentext sowie das Tempo wurden mittels eines Partikelfilter-Ansatzes[2] bestimmt [DP11].

Bochen Li und Duan arbeiteten 2015 noch an einer Verbesserung dieses Algorithmus, indem sie Fehlinterpretationen bei Klavierklängen aufgrund der Verwendung des Fortepedals zu vermindern versuchten. Sie bewerkstelligten dies durch das Unterdrücken von weiterklingenden Teiltönen nach einem neuen Ansatz eines Tons bzw. Akkords [LD15].

1.2.4 Dynamic-Time-Warping-Methoden (ab 2001)

Ungefähr zur selben Zeit, in der Orio mit Déchelle an einem Score-Follower auf Basis von Hidden-Markov-Modellen arbeitete, beschäftigte er sich auch gemeinsam mit Schwarz mit einem anderen Ansatz, der sich als recht vielversprechend herausstellen und auch für andere Forscher interessant werden sollte: die Verwendung von Dynamic-Time-Warping (für eine Einführung in das Thema siehe Abschnitt 1.3.3) um das Audiosignal und den Notentext als erwartetes Signal abzugleichen. Als Vergleichsfeature verwendeten sie Frequenzbereichsrepräsentationen des Eingangssignals und verglichen diese mit generierten Spektren, die entsprechend den erwarteten Klängen aus Rechteckfiltern an den Positionen der jeweiligen Teiltonfrequenzen zusammengesetzt wurden [OS01].

[1] Das »Institut de Recherche et Coordination Acoustique/Musique«, kurz IRCAM, ist eine Forschungseinrichtung in Paris, Frankreich.

[2] Bei Partikelfilter-Methoden werden Aufenthaltswahrscheinlichkeitsverteilungen nicht komplett berechnet, sondern sogenannte Partikel zufällig über den Zustandsraum verteilt und deren Positionen entsprechend der Systemeigenschaften und Beobachtungen rekursiv aktualisiert, bis sie am wahrscheinlichsten Aufenthaltsort im Zustandsraum konvergieren.

Dannenberg und Ning Hu verwendeten ebenfalls Dynamic-Time-Warping, setzten jedoch auf diskrete Chromagramme (für eine Kurzbeschreibung siehe Abschnitt 1.3.4) zur Gewinnung von Vergleichsfeatures, die aus dem eingehenden Audiosignal und aus dem (mit Hilfe der Software »Timidity«) zu einem Audiosignal synthetisierten MIDI-Notentext berechnet wurden [DH03].

Da alle bisher vorgeschlagenen Umsetzungen des Dynamic-Time-Warping nicht echtzeitfähig waren, entwickelte Simon Dixon 2005, teilweise auch zusammen mit Gerhard Widmer, eine Echtzeit-Variante, das Online-Dynamic-Time-Warping. Damit machten sie es möglich, diesen Ansatz nun auch in Systemen für Echtzeitanwendungen benutzen zu können [Dix05; DW05].

Eine andere Variante schlugen Hagen Kaprykowsky und Xavier Rodet vor. Sie entwickelten das sogenannte Short-Time-Dynamic-Time-Warping, das es erlaubte, den globalen optimalen Übereinstimmungspfad auch in Abschnitten zu berechnen und damit Ressourcen einzusparen [KR06].

Im Jahr 2007 griff Andreas Arzt das Thema Online-Dynamic-Time-Warping in seiner Masterarbeit mit dem Titel »Score Following with Dynamic Time Warping – An Automatic Page-Turner« wieder auf, in der er einen auf Score-Following basierenden Umblätter-Automaten entwickelte. Im Zuge dessen baute er auf den Algorithmus von Dixon auf und erarbeitete mehrere Verbesserungsvorschläge, wie etwa die zusätzliche Verwendung der in der Partitur gegebenen Zeitpunkte für Notenansätze, um den per Online-Dynamic-Time-Warping geschätzten Pfad weiter zu verfeinern, oder den vorübergehenden Einsatz von drei Vergleichsprozessen gleichzeitig, um zu erkennen, ob eine Wiederholung ausgelassen oder übersprungen wurde [Arz07].

Auch eine Gruppe um Julio Carabias-Orti, Francisco Rodríguez-Serrano und Pedro Vera-Candeas erarbeitete einige Versionen von Score-Followern auf Basis von Dynamic-Time-Warping und reichte diese in der Zeit von 2012 bis 2015 beim Music Information Retrieval Evaluation eXchange (MIREX) ein [CRV+12; RVC15].

1.2.5 Ausgewählte andere Ansätze und Entwicklungen

Im Jahr 2004 versuchte sich Raphael an einer komplett anderen Herangehensweise, da die Probleme bei den bis dato bekannten Methoden seiner Ansicht nach in der unzulänglichen Modellierung der Notendauer zu suchen wären. Er arbeitete deshalb an einem grafischen Modell, in dem die Notenlänge als abhängig vom aktuellen Tempo betrachtet und der Fokus deshalb auf die Tempomodellierung gelegt wurde [Rap04].

Robert Macrae und Dixon entwickelten 2010 einen Score-Follower, der auf einen völlig anderen Verwendungszweck ausgerichtet war: das Verfolgen von ausschließlich mit ASCII-Textzeichen dargestellten Gitarrentabulaturen, wie sie auf vielen Song-Plattformen im Internet üblich sind. Dazu verwendeten Sie zwei parallele Algorithmen: einerseits schätzten sie die Akkorde aus dem Audioinput und ordneten diese basierend auf Dannenbergs ursprünglichem Algorithmus den Tabulaturen zu, andererseits synthetisierten sie ein Audiosignal aus den Tabulaturen und verglichen das Ergebnis unter Verwendung von Online-Dynamic-Time-Warping mit dem Audio-Input [MD10].

Auch Filip Korzeniowski und seine Kollegen (u.a. Arzt) schlugen 2013 eine weitere Herangehensweise vor, in der sie ein Dynamic-Bayesian-Network[1] als Modell und eine Partikelfilter-Methode[2] zur Bestimmung der Position im Notentext verwendeten [KKAW13].

Im Jahr 2014 erschien eine weitere Masterarbeit zum Thema Score-Following mit dem Titel »Realisierung eines webbasierten Score-Followers für Tasteninstrumente mit Chroma-Features«, verfasst von Erik Werner. Er arbeitete darin an einer Web-Anwendung namens »Flowkey« (siehe auch Abschnitt 1.4.3) zum Erlernen des Klavierspiels, die auch eine Score-Following-Komponente enthielt. Für den Abgleich zwischen eingehendem und erwartetem Signal wurden Chromavektoren aus dem Mikrofonsignal gewonnen und mit aus dem Notentext generierten Chromavektormustern abgeglichen [Wer14].

[1] Dynamic-Bayesian-Networks sind eine Möglichkeit der grafischen Modellierung von kausalen zeitlichen Prozessen.
[2] Für eine Kurzbeschreibung des Begriffs Partikelfilter-Methode siehe Fußnote 2 auf Seite 16.

1.3 Basiskonzepte

Wie aus dem vorangegangenen historischen Überblick hervorgeht, wurden einige grundlegende Ansätze häufig als Basis für darauf aufbauende Systeme verwendet. Die wichtigsten davon sollen hier kurz erläutert werden.

1.3.1 Hidden-Markov-Modelle

Das Konzept von Hidden-Markov-Modellen wurde 1966 von Leonard Baum und Ted Petrie eingeführt [BP66] und gilt noch immer als ein wichtiges Instrument zur Modellierung von stochastischen Prozessen. Lawrence Rabiner charakterisiert diese Klasse von Modellen in »A tutorial on hidden Markov models and selected applications in speech recognition« wie folgt [Rab89, S. 258-261].

Ein Hidden-Markov-Modell ist ein Zustandsmodell eines (zeitdiskreten) Prozesses, der die Markov-Eigenschaft besitzt. Dies ist der Fall, wenn die Wahrscheinlichkeit des Fortschreitens zu einem nächsten Zustand nur vom aktuellen Zustand abhängt [Ros96, S. 163], d.h. wenn die Wahrscheinlichkeit unabhängig von Informationen über die Vergangenheit des Prozesses ist [Ros96, S. 213]. Diese Eigenschaft wird auch Gedächtnislosigkeit genannt [WS13, S. 11]. Die Zustände eines Hidden-Markov-Modells sind jedoch verborgen, daher der englische Begriff »hidden«. Sichtbar sind nur die sogenannten Beobachtungen oder Beobachtungssymbole, die aber nicht direkt jeweils einem Zustand entsprechen, sondern mit bestimmten Beobachtungswahrscheinlichkeiten von verschiedenen Zuständen ausgelöst werden können.

Ein vollständiges Hidden-Markov-Modell wird durch die folgenden fünf zu ermittelnden (bzw. zu schätzenden) Parametergruppen definiert: die unbeobachtbaren Zustände S_i, deren A-Priori-Wahrscheinlichkeiten π_i (benötigt zur Initialisierungszeit des Modells), die Übergangswahrscheinlichkeiten a_{ij} zwischen den Zuständen, die möglichen Beobachtungen B_k sowie die zugehörigen Beobachtungswahrscheinlichkeiten $b_i(k)$.

Zur Veranschaulichung soll der folgende Vorgang als einfaches Beispiel herangezogen werden (vgl. [Rab89, S. 260-261]): Hinter einem Vorhang befinden sich drei Urnen, die verschiedene Mengen von Kugeln in jeweils einer von bis zu vier Farben enthalten; d.h.

es können auch nur eine, zwei oder drei Farben in einer Urne vertreten sein. In mehreren Zügen wird jeweils eine Urne nach bestimmten Regeln ausgewählt und aus ihr zufällig eine Kugel gezogen, die dann vor dem Vorhang gezeigt und anschließend in die jeweilige Urne zurückgelegt wird. Dieser Prozess kann mit Hilfe eines Hidden-Markov-Modells modelliert werden, dessen Zustandsgraph in Abbildung 1.2 dargestellt ist.

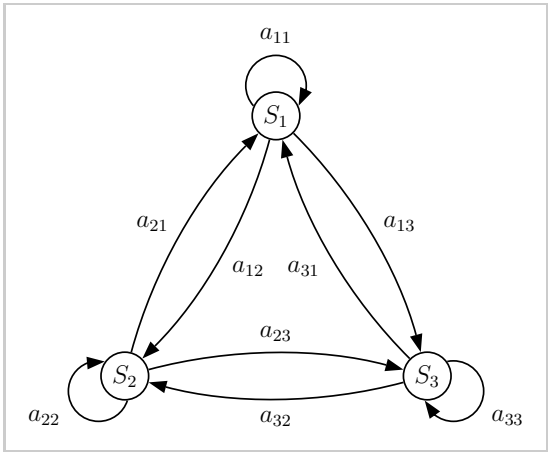


Abbildung 1.2: Zustandsgraph des Hidden-Markov-Modells für drei Urnen, aus denen farbige Kugeln gezogen und anschließend zurückgelegt werden

Die drei Zustände S_1 bis S_3 repräsentieren dabei die drei Urnen und die zugehörigen A-Priori-Wahrscheinlichkeiten π_1 bis π_3 geben an, wie oft eine Urne durchschnittlich als allererste ausgewählt wird. Welche Zustände auf einen vorherigen folgen können, wird durch die verbindenden Pfade angezeigt. Diesen sind Übergangswahrscheinlichkeiten a_{ij} zugeordnet, die modellieren, wie oft Urne j im Vergleich zu den anderen möglichen Urnen auf Urne i folgt.

Die möglichen Beobachtungen B_k des Modells entsprechen den vier Farben $k \in \{1, 2, 3, 4\}$ der gezeigten Kugeln. Jedem Zustand S_i sind deshalb zusätzlich vier Beobachtungswahrscheinlichkeiten $b_i(k)$ zugeordnet, deren Werte davon abhängen, wie viele Kugeln welcher Farbe sich in der jeweiligen Urne befinden. Enthält Urne 2 beispielsweise 3 rote ($k = 1$), 2 gelbe ($k = 2$), 4 grüne ($k = 3$) und keine blauen ($k = 4$) Kugeln, so würden sich als Beobachtungswahrscheinlichkeiten für Urne 2 Werte von $b_2(1) = \frac{3}{9}$, $b_2(2) = \frac{2}{9}$, $b_2(3) = \frac{4}{9}$ und $b_2(4) = 0$ ergeben.

Wie eben gezeigt, können die Wahrscheinlichkeiten für einzelne Beobachtungen auch Null sein, womit diese entsprechend dem modellierten Prozess als nicht möglich bzw. nicht erlaubt definiert werden können. Analoges gilt auch für Übergangs- und A-Priori-Wahrscheinlichkeiten. Dennoch müssen die Summen der jeweiligen Wahrscheinlichkeitsgruppen natürlich immer die volle Wahrscheinlichkeit von 100 % ergeben ($\sum_i \pi_i = 1$, $\sum_j a_{ij} = 1$, $\sum_k b_i(k) = 1$).

Auf Basis eines solchen Hidden-Markov-Modells ist es dann mit geeigneten Methoden möglich, etwa Fragen nach der Wahrscheinlichkeit einer bestimmten Beobachtungsfolge oder der wahrscheinlichsten den Beobachtungen zugrundeliegenden Zustandsfolge zu beantworten. Bezogen auf das Beispiel könnten solche Fragen etwa lauten: »Wie wahrscheinlich ist es, dass zuerst eine rote, dann eine blaue, eine grüne und wieder eine rote Kugel gezogen werden?«, oder »Aus welchen Urnen wurde eine bestimmte Folge von Farben höchstwahrscheinlich gezogen?«. Für Zweiteres kann z.B. der Viterbi-Algorithmus verwendet werden.

1.3.2 Viterbi-Algorithmus

Der bis heute verwendete Viterbi-Algorithmus wurde 1967 von Andrew Viterbi zur Dekodierung von Faltungscodes, einer Form von Codierung zur Übertragung von Bitstreams, entwickelt. Die folgende Zusammenfassung, basierend auf den Ausführungen von George Forney [For73, S. 268-273] sowie Matthew Ryan und Graham Nudd [RN93, S. 1-9], bietet einen Überblick über die Funktionsweise.

Zweck des Algorithmus ist es, aus einer Folge von (z.B. durch die Übertragung möglicherweise verfälschten) Beobachtungen eines endlichen, zeitdiskreten Zustandsautomaten die wahrscheinlichste zugrunde liegende Zustandsfolge zu schätzen, sofern der Zustandsautomat einen Prozess mit der Markov-Eigenschaft (siehe Abschnitt 1.3.1) darstellt. Im Optimalfall können so z.B. Übertragungsfehler auf der Empfängerseite korrigiert werden.

Zum besseren Verständnis der Funktionsweise wird oft das sogenannte Trellis-Diagramm herangezogen. Es besteht aus einem Gitter von Knotenpunkten, in dem die Zeilen für die Zustände und die Spalten für die diskreten Zeitpunkte stehen. Jeder Knotenpunkt

entspricht also einem Zustand zu einer bestimmten Zeit. Zusätzlich werden die Übergänge von jedem Zustand des aktuellen Zeitpunktes zu den laut Zustandsdiagramm möglichen folgenden Zuständen als Verbindungszweige dargestellt, wobei diesen jeweils Kosten entsprechend einer Kostenfunktion basierend auf der aktuellen Beobachtung sowie den Übergangs- und Beobachtungswahrscheinlichkeiten zugewiesen werden.

Der Algorithmus berechnet nun zu jedem Zeitpunkt jeweils für alle möglichen ankommenden Zweige jedes Zustandes die Summe der Kosten des jeweiligen Zweiges und aller ihm vorangegangenen Zweige (diese sind aus der vorherigen Iteration bekannt und werden Survival-Pfad genannt). Anschließend fügt er für jeden Zustand jenen Zweig, der zu den niedrigsten Gesamtkosten führt, als neuen Teil des Survival-Pfads hinzu. Nach genügend Iterationen entspricht dann jener Survival-Pfad mit den geringsten Gesamtkosten der wahrscheinlichsten zugrundeliegenden Zustandsfolge im beobachteten Zeitintervall.

1.3.3 Dynamic-Time-Warping

Die Idee des Dynamic-Time-Warping entstand Anfang der 70er Jahre im Kontext der Spracherkennung. Die generelle Vorgehensweise ist nachfolgend auf Basis der Erklärungen im Buch »Information Retrieval for Music and Motion« [Mül07, S. 69-74] überblicksmäßig zusammengefasst.

Aufgabe der Methode ist es, die sich entsprechenden Elemente zweier vorliegender Datenfolgen, die zwar ähnliche Inhalte haben, aber eventuell zu unterschiedlichen Zeiten unterschiedlich schnell verlaufen, einander möglichst optimal zuzuordnen.

Dazu wird zuerst das Übereinstimmungsmaß für jede mögliche Paarkombination von jeweils einem Datenpunkt jeder Folge mittels einer zuvor definierten Kostenfunktion ermittelt. Die Ergebnisse werden in Form einer Kostenmatrix festgehalten, in der die Zeilen den Elementen der ersten und die Spalten den Elementen der zweiten Folge entsprechen. Die bestmögliche Zuordnung der beiden Folgen zueinander entspricht dann dem Verlauf eines sogenannten optimalen Warping-Pfads durch diese Matrix, welcher sich als (bestimmten Kriterien genügende) Verbindungslinie jener Matrixzellen ergibt, deren Gesamtsumme der Kosten minimal wird.

Um diesen optimalen Pfad zu ermitteln, wird weiters auf Basis der Kostenmatrix eine sogenannte akkumulierte Kostenmatrix errechnet, die in jeder Zelle die (rekursiv berechenbaren) bisherigen Gesamtkosten des dort endenden optimalen Teilpfades enthält. Ausgehend von der letzten Zelle dieser Matrix[1], die dementsprechend die höchsten Gesamtkosten enthält und definitionsgemäß der Zuordnung der jeweils letzten Elemente der beiden Folgen zueinander entspricht, kann nun durch schrittweises Aufsuchen der jeweils (den Fortschreitungsregeln entsprechenden) vorangehenden Zelle mit den geringsten bisherigen Gesamtkosten der optimale Warping-Pfad Stück für Stück aufgebaut werden.

1.3.4 Chroma

Der Begriff Chroma steht im musikalischen Kontext in Zusammenhang mit der musiktheoretischen Sichtweise, dass die einzelnen Töne einer Tonleiter in jeder Oktave wiederkehren. Mark Bartsch und Gregory Wakefield geben in »Audio thumbnailing of popular music using chroma-based representations« [BW05, S. 97-98] einen Überblick über die Thematik, die sich wie folgt zusammenfassen lässt.

Schon früh wurde erkannt, dass bei der Wahrnehmung von Tonhöhen zwischen Tönen im Abstand von genau einer oder mehrerer Oktaven ein Zusammenhang besteht. Deshalb entstand ein Modell, das die Tonhöhe nicht auf einer eindimensionalen Gerade, sondern als zweidimensionale Größe auf einer Helix[2] darstellt (für Näheres dazu siehe z.B. [She64]). Dabei wurde die Dimension entlang der Mittelachse der Helix als Tonhöhe bezeichnet, die dem Rotationswinkel entsprechende Dimension als Chroma. In diesem Modell liegen beispielsweise die Punkte der beiden Töne a' (440 Hz) und a'' (880 Hz) auf zwei benachbarten Windungen der Helix entlang der Mittelachse gesehen direkt übereinander. Sie haben beide den gleichen Chromawert (oder in Zylinderkoordinaten gesehen den gleichen Winkel) und gehören demnach derselben Chromaklasse an.

[1] Mit der Bezeichnung »letzte Zelle einer Matrix« ist hier die Schnittzelle der jeweils letzten Zeile und Spalte der Matrix gemeint.

[2] Eine Helix, auch Schraubenlinie genannt, ergibt sich als jene Linie im Raum, die ein Punkt verfolgt, wenn man ihn gleichzeitig mit jeweils konstanter Geschwindigkeit sowohl entlang der Kreisbahn der Grundfläche eines Zylinders als auch in dessen Symmetrieachsenrichtung bewegt. Ein Beispiel für eine solche Linie wäre der Handlauf einer Wendeltreppe.

Auf Basis dieser Definition kann nun nach Vorbild des Frequenzspektrums ein sogenanntes Chromaspektrum berechnet werden, das die Signalenergieverteilung eines Audiosamples auf einer Chromaachse aufträgt. Öfter als das allgemeine wertkontinuierliche Chromaspektrum wird jedoch die Form des diskreten Chromaspektrums verwendet. Eine häufige Variante teilt den kontinuierlichen Wertebereich entsprechend den in westlicher Musik üblichen 12 chromatischen Tönen in 12 Chromabänder ein und fasst deren Inhalte zu Chroma-Bins zusammen.

Weiters kann analog zur Repräsentation des Frequenzverlaufs über die Zeit in Form eines Spektrogramms auch der Chromaverlauf eines Audiosignals als sogenanntes Chromagramm dargestellt werden.

1.4 Aktuelle Anwendungen und Softwarepakete

Zu den Anwendungsmöglichkeiten des Score-Followings zählen neben der automatischen Begleitung von Solisten, einem der ersten verfolgten Forschungsgründe, auch die Verwendung in den Bereichen Live-Visualisierung oder automatisches Umblättern von Noten. Außerdem wird Score-Following in jüngerer Zeit als Teil von kommerzieller Software zur Unterstützung beim Erlernen von Musikinstrumenten oder als Bereicherung beim Erleben von Musik eingesetzt.

Nachfolgend sollen nun exemplarisch einige aktuelle Anwendungen und Softwarepakete vorgestellt werden.

1.4.1 Automatische Begleitung

Die Bereitstellung einer automatischen Begleitung für Soloinstrumente war eine der ersten Motivationen, warum Forschung auf dem Gebiet des Score-Following betrieben wurde. Aktuell existiert im Wesentlichen nur ein einziges, voll funktionsfähig erhältliches Softwarepaket, sowie ein zweites, darauf aufbauendes System.

Antescofo ist aktuell das wohl umfangreichste und zugleich einzige System am Markt, das voll funktionsfähig und sofort anwendbar erhältlich ist. Es handelt sich dabei um ein External[1] für Max/MSP[2] bzw. Pure Data[2], das es ermöglicht, einer musikalischen Darbietung zu folgen, und zuvor an bestimmte Events gebundene Aktionen auszuführen. Begleitet wird es von einer eigenen Programmiersprache für die notwendigen Partituren und dem Zusatzprogramm AscoGraph, das sowohl als Partitur-Editor als auch für die Anzeige der aktuellen Position in einem Stück eingesetzt werden kann [Ant; Asc; CSSR07]. Für ausführliche Informationen zu Antescofo siehe Kapitel 2.

IMuSE (Integrated Multimodal Score-following Environment) ist eine Anwendung aufbauend auf dem Notensatzprogramm NoteAbility Pro sowie den Externals Antescofo und Gesture Follower. Die Softwareumgebung ermöglicht eine multimodale Verfolgung einer Darbietung inklusive Aktionsauslösung auf Basis der parallelen Verwendung sowohl von Audio-/MIDI-Signalen (durch Antescofo) als auch von Gestendaten (mittels Gesture Follower), die zuvor in der hybriden Partitur (in NoteAbility Pro) zusammen mit dem herkömmlichen Notentext festgehalten wurden [RHP13; Imu].

1.4.2 Automatisches Umblättern für digitale Noten

Seit sich die Verwendung von Tablets steigender Beliebtheit erfreut, haben sich auch Apps zur Anzeige von digitalen Noten entwickelt. Darunter existieren vereinzelt auch Vertreter wie etwa die beiden folgenden Apps, die über das eingebaute Mikrofon eine Darbietung verfolgen und den angezeigten Notentext automatisch umblättern können.

Tonara bietet einen eigenen Store mit kompatiblen Notentexten an, in denen neben anderen Funktionen auch während der Darbietung die aktuelle Position markiert und automatisch umgeblättert werden kann [Ton].

[1] Als Externals werden externe Erweiterungen bezeichnet, die in Max/MSP bzw. Pure Data verwendet werden können.
[2] Max/MSP und Pure Data sind zwei Varianten einer echtzeitfähigen, grafischen Programmiersprache für Multimedia-Anwendungen.

Autoflip arbeitet dagegen mit Notentexten im PDF-Format. Unter anderem kann nach einem Trainingsdurchgang, in dem das gewünschte Stück gespielt und dabei manuell umgeblättert wird, anschließend das Umblättern automatisch durchgeführt werden [Aut].

1.4.3 Erlernen von Musikinstrumenten

Auch zum Erlernen von Musikinstrumenten gibt es mittlerweile einige Apps am Markt, manche davon bedienen sich auch der Technik des Score-Followings, um das Nutzungserlebnis zu erhöhen.

Flowkey ist ein Programm zur Unterstützung beim Erlernen des Klavierspiels. Dabei werden der bereitgestellte digitale Notentext sowie ein zugehöriges Tutorial-Video angezeigt, in denen jeweils die gerade zu spielenden Noten bzw. Tasten hervorgehoben werden. Es bietet verschiedene Modi wie etwa langsames Mitspielen oder eine Loop-Funktion, sowie einen sogenannten Warte- oder Flow-Modus, in dem das Klavierspiel des Benutzers über das Mikrofonsignal (auch MIDI möglich) analysiert wird, und das Programm bei jeder Note bzw. jedem Akkord so lange wartet, bis der dem Notentext entsprechende Klang registriert wurde [Flo].

SmartPiano ist ein weiteres Klavierlernsystem, bestehend aus einem E-Piano und einer zugehörigen App, die mit dem Instrument verbunden wird. Auch hier wird eine Funktion ähnlich dem Warte-Modus von Flowkey geboten, welche die Tastenanschläge über die Verbindung zum E-Piano registriert, und beim Lernen der bereitgestellten Notentexte erst mit der nächsten Note bzw. dem nächsten Akkord fortsetzt, wenn die richtigen Tasten gedrückt wurden. Dabei wird der Benutzer durch LEDs an den Tasten unterstützt, die aufleuchten, wenn die jeweilige Taste entsprechend dem Notentext gedrückt werden soll [SmaA; SmaB].

1.4.4 Bereicherung des Musikerlebnisses

Außerdem werden in jüngster Zeit auch Forschungen durchgeführt, die darauf abzielen, das Erlebnis der Zuhörer bei einem klassischen Konzert multimedial zu unterstützen.

PHENICX (kurz für »Performances as Highly Enriched aNd Interactive Concert eXperiences«) ist ein u.a. von der Europäischen Union gefördertes Projekt, das sich seit 2013 mit Fragestellungen zum Thema Unterstützung und Bereicherung des Erlebens von Musik durch die Nutzung von aktuellen Technologien beschäftigt. Dabei sollen Konzertvorstellungen mit Hilfe einer App in multimediale Erlebnisse verwandelt und Möglichkeiten zur Interaktion vor, während und nach dem Konzert geboten werden. U.a. wird auch Score-Following eingesetzt, um etwa einen Notentext synchronisiert mit der Darbietung anzuzeigen, passende musikalische Informationen zur richtigen Zeit im Stück einzublenden oder gerade aktive Instrumentensektionen auf einem Orchesterplan hervorzuheben [Phe14].

2 Antescofo

Da Antescofo die wichtigste softwaretechnische Basiskomponente dieser Arbeit darstellt, widmet sich dieses Kapitel nun der näheren Beschreibung des Softwarepakets. Dazu gibt Abschnitt 2.1 einen Überblick über die Entstehung sowie die Intentionen und Anwendungsbereiche des Systems, während Abschnitt 2.2 dessen Architektur und zugrunde liegende Modelle beschreibt. Abschnitt 2.3 beschäftigt sich anschließend mit der begleitenden Programmiersprache zur Partiturerstellung und Abschnitt 2.4 geht auf die Anwendung der Software ein.

2.1 Überblick

Das Softwarepaket Antescofo besteht im Wesentlichen aus zwei Teilen [GCEM16, S. 1]: Einerseits enthält es einen polyphonen Score-Follower, mit dem passend zu einer musikalischen Darbietung die aktuelle Position in der zugehörigen Partitur in Echtzeit verfolgt werden kann. Andererseits wird eine eigens entwickelte synchrone Programmiersprache[1] bereitgestellt, die zur Erstellung der benötigten Antescofo-Partituren inklusive der Möglichkeiten zur Notation eventueller elektronischer Parts verwendet wird.

2.1.1 Entstehung

Das Paket entstand im Jahr 2007 aus der Zusammenarbeit des Komponisten Marco Stroppa mit dem Wissenschaftler Arshia Cont [GCEM16, S. 5], als sich während der Komposition von »... of Silence« [Con08, S. 8] für Saxophon und elektronische Parts die Notwendigkeit für ein System ergab, das sowohl geeignete Notationsmöglichkeiten

[1] Synchrone Programmiersprachen sind Sprachen zur Erstellung von reaktiven Systemen. Dies sind Systeme, die ständig auf ihre Umgebung reagieren, u.a. Echtzeitsysteme [BG92, S. 88].

in der Kompositionsphase bieten als auch die Ausführung der elektronischen Parts in der Konzertphase ermöglichen sollte. Seitdem wird Antescofo bis heute weiterentwickelt, aktuell vom über die Jahre entstandenen MuTant Projektteam in Frankreich, in dem Vertreter des Forschungsinstituts für Akustik/Musik (IRCAM), des Nationalen Zentrums für wissenschaftliche Forschung (CNRS), des Nationalen Instituts für Computerwissenschaft und angewandte Mathematik (INRIA) und der Universität Pierre und Marie Curie (UPMC) zusammenarbeiten.

2.1.2 Intentionen und Hauptanwendungsbereich

Ausgehend von dessen Entstehungsgeschichte sind die beiden Hauptintentionen von Antescofo die Erstellung von Partituren, welche sowohl die Stimmen von akustischen Instrumenten als auch die elektronischen Parts beinhalten, sowie die Bereitstellung einer Software, die eine Interaktion zwischen diesen beiden Seiten ermöglicht [GCEM16, S. 1].

Ersteres ist ein neuer Ansatz, da in bisherigen Systemen für Computermusikstücke die Notation bzw. Spezifikation der instrumentalen bzw. elektronischen Stimmen getrennt voneinander vorgenommen und extern miteinander verknüpft werden musste [GCEM16, S. 6]. Mit Antescofo wurde nun eine Programmiersprache entwickelt, die es erlaubt, nicht nur das Notenmaterial der akustischen Instrumente zu notieren, sondern auch die elektronischen Parts festzuhalten sowie die Interaktionen zwischen den Instrumenten und den computergenerierten Teilen direkt in einer Partitur zu definieren. Da bei Antescofo alles zusammengefasst notiert ist, kann das interne Zeitmanagement der Software als Basis für sowohl die Synchronisation der Parts als auch zur Steuerung der elektronischen Teile verwendet werden. Dies erleichtert einerseits das Zusammenspiel zwischen akustischen Instrumenten und elektronischen Parts, andererseits ermöglicht es die Verwendung von relativen Zeitangaben in Taktschlägen im Gegensatz zur sonst ausschließlichen Verfügbarkeit von absoluten Zeitangaben wie etwa Millisekunden. Für weitere Informationen zur Programmiersprache siehe Abschnitt 2.3.

Zum Zweck der Interaktion zwischen den instrumentalen und elektronischen Parts entstand ein Score-Follower in Form eines Externals für Max/MSP bzw. Pure Data, der

den zweiten und zugleich zentralen Teil des Pakets darstellt. Auf Basis einer vorliegenden Antescofo-Partitur eines Stücks ist die Software in der Lage, die Darbietung eines Instruments zu verfolgen und der aktuellen Position in der Partitur zuzuordnen [GCEM16, S. 1-2]. Zusätzlich wird auch das aktuelle Tempo der Darbietung detektiert, welches zusammen mit der aktuellen Position zur Synchronisation und Steuerung eventueller elektronischer Parts herangezogen wird. Damit ist es möglich, die Software als Bindeglied zwischen einem akustischen Instrument und laut der Partitur in Echtzeit generierten elektronischen Klängen bei Konzertsituationen einzusetzen, was den Hauptanwendungsbereich des Softwarepaketes darstellt.

2.2 Systemarchitektur

In diesem Abschnitt sollen nun die softwaretechnische Architektur von Antescofo sowie die verwendeten Modelle und Algorithmen auf Basis von [Con10, S. 3-10] beleuchtet werden (für Implementationsdetails und mathematische Herleitungen zu den verwendeten Algorithmen sei auf ebendiese Quelle verwiesen).

Grundsätzlich basiert die Softwarearchitektur auf dem Ansatz, dass das Audiosignal einer Darbietung theoretisch auch als Summe der Beobachtungen eines passend zur Partitur des Musikstückes konstruierten Zustandsmodells angesehen werden könnte. Auf dieser Basis kann der Score-Following-Prozess als die bei Zustandsmodellen häufig gestellte Frage interpretiert werden, welche Zustandsfolge höchstwahrscheinlich für die bisher festgestellten Beobachtungen verantwortlich ist (vgl. Abschnitt 1.3.1). Der letzte Zustand dieser Zustandsfolge entspricht dann der aktuellen Position in der Partitur.

In Abbildung 2.1 sind die wichtigsten Komponenten zur Umsetzung dieses Konzepts als Blockdiagramm dargestellt. Zunächst erstellt der Partitur-Parser (Abschnitt 2.2.1) in einem Vorverarbeitungsschritt die grundlegende Struktur des Zustandsmodells aus der zu verfolgenden Partitur. Zudem ist vor Beginn der Partiturverfolgung auch die Angabe jener Position notwendig, an der die Darbietung beginnen soll. Während der Echtzeitverwendung des Score-Followers versorgt der Input-Beobachter[1] (Abschnitt 2.2.2)

[1] Zum besseren Verständnis im Kontext des Zustandsmodells wird diese Komponente, die in [Con10] die Bezeichnung »Features« trägt, im Rahmen dieser Arbeit »Input-Beobachter« genannt.

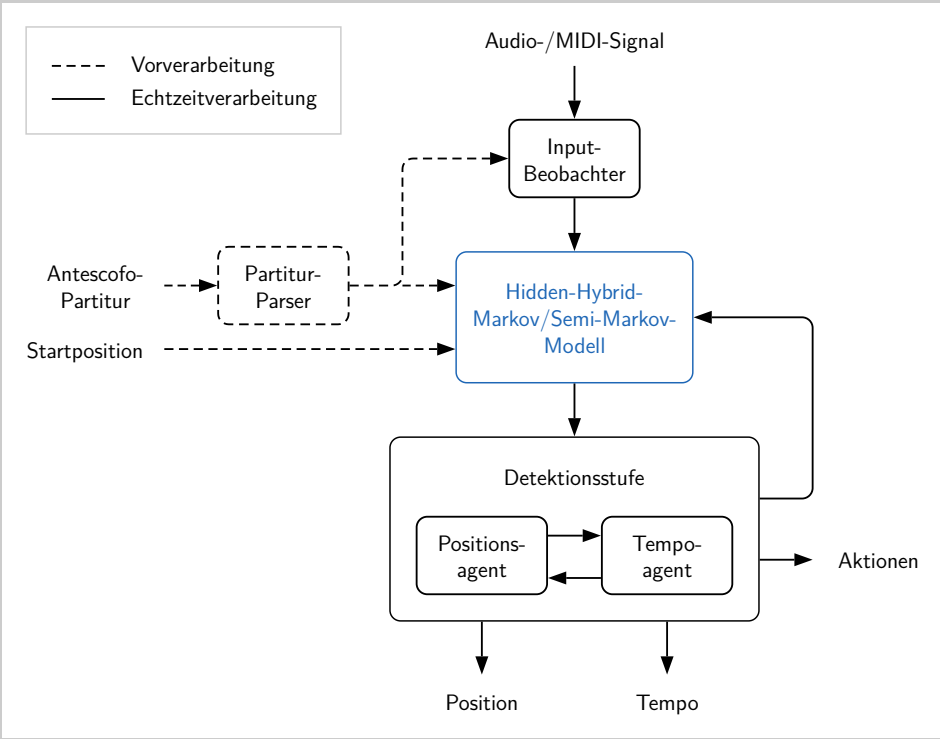


Abbildung 2.1: Blockdiagramm des generellen Softwareaufbaus von Antescofo (vgl. [Con10, S. 4])

das Modell mit Informationen über das aktuelle Eingangssignal, womit es dann in der Detektionsstufe vom Tempoagenten für die Ermittlung des aktuellen Tempos verwendet werden kann (Abschnitt 2.2.3). Auf Basis dieser Tempowerte werden wiederum die entsprechenden Parameter des Zustandsmodells aktualisiert, sodass der Positionsagent[1] schlussendlich die aktuelle Position in der Partitur bestimmen kann (Abschnitt 2.2.4).

2.2.1 Partitur-Parser: Erstellung der Modellstruktur

Der Partitur-Parser übernimmt zu Beginn die Aufgabe, ausgehend von der gegebenen Antescofo-Partitur eines Stücks die grundsätzliche Struktur eines für den Zweck der Partiturverfolgung geeigneten Zustandsmodells zu erstellen.

[1] Auf Hinweis des Antescofo-Projektleiters Arshia Cont während eines Gesprächs wurde die ursprüngliche Bezeichnung »Audio-Agent« [Con10] im Rahmen dieser Arbeit zu »Positionsagent« abgeändert, da diese Komponente, egal bei welcher Art von Input (Audio, MIDI etc.), die Position bestimmt.

Als Modelltyp wird im Hinblick auf die angestrebte Verwendung ein Hidden-Hybrid-Markov/Semi-Markov-Modell verwendet. Dieses stellt eine Kombination von Hidden-Markov-Modellen (siehe Abschnitt 1.3.1) und Hidden-Semi-Markov-Modellen[1] dar, in der neben den klassischen Markov-Zuständen auch sogenannte Semi-Markov-Zustände zur Verfügung stehen. Diese zeichnen sich durch die folgenden beiden Eigenschaften aus:

Beliebige, explizit definierbare Verweilzeiten für Semi-Markov-Zustände werden ermöglicht, da für die Modellierung der Verweildauern[2] beliebige Wahrscheinlichkeitsverteilungen gewählt werden können [FRK10, S. 112]. Für Markov-Zustände (entsprechend einer klassischen Markov-Kette) ist dies nicht möglich, da die Verweilzeiten hier immer genau einer Zeiteinheit entsprechen [Ros96, S. 213]. Das Verweilen in einem Zustand ist somit nur durch Übergänge zurück zum selben Zustand zu erreichen, wobei die summierte Verweildauer dabei als geometrisch verteilt interpretiert werden kann [Yu10, S. 215].

Die Möglichkeit mehrerer Beobachtungen pro Zustand entsprechend der jeweiligen Verweildauer ist ein weiteres Merkmal von Semi-Markov-Zuständen. Bei klassischen Markov-Zuständen ist dagegen nur eine Beobachtung pro Zustand möglich [Yu10, S. 216].

Der Partitur-Parser erstellt nun die Grundstruktur des Zustandsmodells, indem er jedes Event in der Antescofo-Partitur als ein oder mehrere Zustände darstellt. Durch die Wahl des Hidden-Hybrid-Markov/Semi-Markov-Modells als Typ des Zustandsmodells können die Vorteile beider Zustandsarten nebeneinander genutzt und die Art (und Anzahl) der Zustände passend zur Beschaffenheit des jeweiligen Events gewählt werden. Zu diesem Zweck werden im Wesentlichen die folgenden drei Event-Kategorien unterschieden.

[1] Die Bezeichnung »semi-Markov« leitet sich vom folgenden Sachverhalt ab [FRK10, S. 119]: Die Markov-Eigenschaft (siehe Abschnitt 1.3.1) trifft auf Semi-Markov-Prozesse im Allgemeinen nicht zu, da beliebige Verweilzeiten festgelegt werden können und die Übergangswahrscheinlichkeiten damit abhängig von der bisherigen Verweildauer im aktuellen Zustand sind; streng genommen ist dieses Vergangenheitswissen jedoch genau zum Übergangszeitpunkt nicht mehr nötig [Ros96, S. 213], weshalb die Markov-Eigenschaft dort erfüllt ist.

[2] Als Verweildauer oder Verweilzeit eines Zustandes wird jene Zeit bezeichnet, in welcher der Zustandsautomat in ebendiesem Zustand verbleibt.

Einfache Events mit expliziter Dauer sind Ereignisse wie etwa Noten oder Pausen, denen bestimmte Längen zugeordnet sind. Solche Events werden mit einem einzelnen Semi-Markov-Zustand modelliert, da diese Zustandsart die explizite Angabe der Event-Dauer erlaubt.

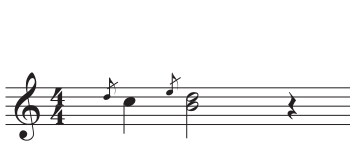
Einfache Events mit unbestimmter Dauer werden hingegen mit einzelnen Markov-Zuständen dargestellt, um den Vorteil zu nutzen, dass Dauer bzw. Wahrscheinlichkeitsverteilung nicht explizit definiert werden müssen. Vorschlagnoten wären beispielsweise Vertreter dieser Klasse.

Zusammengesetzte Events stellen eine zusammengehörige Struktur dar, die aus mehreren einfachen Events besteht, wobei nur die Dauer der gesamten Struktur definiert ist, jene der enthaltenen Elemente jedoch nicht. Eine solche Struktur, wie z.B. ein Triller oder ein Glissando, wird demnach als ein übergeordneter Semi-Markov-Zustand modelliert, der mehrere Markov-Zustände für die untergeordneten Elemente enthält.

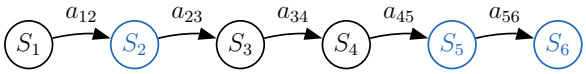
Jedem Zustand werden anschließend zwei Eigenschaften zugewiesen: eine fortlaufende Nummer, die der Position des modellierten Events innerhalb der Partitur entspricht, sowie die für dieses Event zu erwartenden Frequenzen laut Partitur (Pausen erhalten einen Frequenzwert von Null). Für zusammengesetzte Zustände bzw. Events erhalten nur die untergeordneten Zustände Werte für die Tonhöhen. Handelt es sich um einen Semi-Markov-Zustand, also ein Event mit definierter Dauer, wird diese Dauer als weitere Eigenschaft des Zustandes festgehalten.

Zur Veranschaulichung zeigt Abbildung 2.2 einen Beispielnotentext und die Struktur des daraus erstellten Zustandsmodells sowie die Eigenschaften[1] der Zustände. Darin ist zu sehen, dass der Partitur-Parser in diesem Fall jedes Event mit einem Zustand dargestellt hat. Dabei entsprechen die schwarzen Markov-Zustände S_1 und S_4 den beiden Vorschlagnoten, deren Tonhöhen als Eigenschaften $f_1 = \text{d''}$ bzw. $f_4 = \text{e''}$ festgehalten werden. Die Viertelnote, der Akkord und die Pause werden durch die blauen Semi-Markov-Zustände S_2, S_5 und S_6 modelliert, wobei diesen zusätzlich zu deren Tonhöhen $f_2 = \text{c''}$ und $f_5 = \{\text{h'}, \text{d''}\}$ bzw. $f_6 = 0$ für die Pause auch die Notendauern $l_2 = 1$,

[1] Obwohl die Tonhöheneigenschaft eigentlich den zu erwartenden Frequenzen entspricht, werden diese aus Gründen der Verständlichkeit im Rahmen dieses Beispiels mit deren Tonnamen dargestellt.



a. Notentext



i:	1	2	3	4	5	6
f_i:	d''	c''	0	e''	h', d''	0
l_i:	–	1	–	–	2	1

b. Struktur und Eigenschaften des erstellten Modells

Abbildung 2.2: Veranschaulichung eines Zustandsmodells mit Markov- (schwarz) und Semi-Markov-Zuständen S_i (blau) inklusive deren Eigenschaften Event-Nummer i, Tonhöhe(n) f_i und Notendauer l_i in Schlägen, sowie Übergangswahrscheinlichkeiten a_{ij} (vgl. [Con10, S. 6])

$l_5 = 2$ und $l_6 = 1$ (in Schlägen) als Eigenschaften zugeordnet werden. Wenn es ihm zur besseren Modellierung sinnvoll erscheint, fügt der Partitur-Parser zusätzlich sogenannte Dummy-Pausen ein, d.h. Pausen mit einer Dauer von Null. Im obigen Beispiel wurde etwa Zustand S_3 hinzugefügt, da ein Absetzen des Klanges vor der zweiten Vorschlagnote zu erwarten ist, und die Detektion damit erleichtert wird.

Außerdem stattet der Partitur-Parser das Zustandsmodell mit Übergangswahrscheinlichkeiten auf Basis der Partitur aus; wie genau dies geschieht, geht aus den Quellen jedoch nicht hervor. Aufgrund der verfügbaren Informationen über die Fähigkeiten des Systems ist aber anzunehmen, dass die in Abbildung 2.2b eingetragenen Übergangspfade a_{ij} zum jeweils nächsten Zustand im Vergleich zu den jeweiligen Alternativen mit den höchsten Wahrscheinlichkeiten belegt werden, da stark von einer Darbietung entsprechend des Stückverlaufs auszugehen ist. Die Tatsache, dass den Score-Follower auch kleine Bereiche übersprungener Events nicht aus der Bahn werfen, legt zudem entgegen der Beispieldarstellung nahe, dass auch Pfade existieren, die zu späteren Zuständen führen, wenn auch mit geringeren Wahrscheinlichkeiten. Wird die standardmäßig deaktivierte, aber verfügbare Möglichkeit zur Rückwärtskorrektur der detektierten Position verwendet, gilt dasselbe vermutlich auch für Pfade zu früheren Zuständen. In Fällen von in der Partitur notierten Sprüngen, wie etwa einer Wiederholung, deutet die Antescofo-Dokumentation [GCEM16, S. 20] des dafür vorgesehenen Event-Attributs (`@jump`, siehe Abschnitt 2.3.2) darauf hin, dass mehrere Pfade mit höheren Wahrscheinlichkeiten vom Ausgangszustand eines Sprungs ausgehen: im Fall einer Wiederholung beispielsweise einer, der zum ers-

ten Zustand des Wiederholungsbereichs führt, und ein zweiter als Übergang zum ersten Zustand nach der Wiederholung.

Damit sind am Ende der Vorverarbeitung durch den Partitur-Parser bereits zwei der sechs Parametergruppen eines Hidden-Hybrid-Markov/Semi-Markov-Modells (vgl. auch Abschnitt 1.3.1) definiert: die Zustände und die Übergangswahrscheinlichkeiten. Durch Initialisierung des Score-Following-Prozesses mit der gewünschten Startposition für die Verfolgung können von dieser zudem die A-Priori-Wahrscheinlichkeiten abgeleitet werden (auch hier geben die Quellen wenig Aufschluss darüber, wie dies passiert, es liegt jedoch nahe, dass jenem Zustand, der die Startposition darstellt, ein hoher Wahrscheinlichkeitswert zugeordnet wird, und den anderen Zuständen niedrige Werte). Außerdem stellt das Eingangssignal die Beobachtungen dar, womit nur noch zwei Parametergruppen zu ermitteln sind: die Beobachtungswahrscheinlichkeiten, welche durch den Input-Beobachter bestimmt werden (Abschnitt 2.2.2), und die Wahrscheinlichkeitsverteilungen für die Verweildauern der Semi-Markov-Zustände, die nicht nur von den in der Partitur angegebenen Notendauern, sondern (da diese relativ zum Tempo angegeben sind) auch vom aktuellen Tempowert abhängen, welcher wiederum vom Tempoagenten detektiert wird (Abschnitt 2.2.3).

2.2.2 Input-Beobachter: Analyse des Eingangssignals

Während des Score-Following-Prozesses hat der Input-Beobachter die Aufgabe, laufend das Eingangssignal zu beobachten, um das Zustandsmodell mit den notwendigen Beobachtungswahrscheinlichkeiten zu versorgen.

Dazu werden einerseits die vom Partitur-Parser bereits festgelegten Modellzustände herangezogen, um für jeden Zustand ein sogenanntes Frequenz-Template zu konstruieren. Diese Frequenz-Templates modellieren den Klang des jeweiligen Zustandes im Frequenzbereich, indem dort Normalverteilungen so positioniert werden, dass deren Mittelwerte genau auf den Grundtonfrequenzen der im Klang enthaltenen Töne sowie auf jenen der Obertöne zu liegen kommen. Anzahl und Zusammensetzung der modellierten Teiltöne sind einstellbar (siehe Abschnitt 2.4.2), wobei die Standardeinstellung den ersten zehn harmonischen Teiltönen (ein- bis zehnfache Grundfrequenz) entspricht. Die Varianzen

der Verteilungen entsprachen ursprünglich standardmäßig einem Halbton in temperierter Stimmung [Con10, S. 8], der Standardwert wurde jedoch später auf drei Zehntel eines Ganztons (also etwa einen Drittelton) reduziert [AntHV]. Bei Bedarf kann der Varianzparameter jedoch ebenfalls angepasst werden (siehe Abschnitt 2.4.2), um auf die Trennschärfe nebeneinander liegender Töne einwirken und damit die Akzeptanz des Score-Followers gegenüber etwaigen Tonhöhenabweichungen beeinflussen zu können.

Andererseits werden ebenfalls Frequenzbereichsdarstellungen des jeweils aktuellen Eingangssignals erstellt, die bezogen auf das Zustandsmodell der jeweils aktuellen Beobachtung entsprechen. Handelt es sich um Audio-Input, wird dieser blockweise einer Fast-Fourier-Transformation (FFT) unterzogen, deren Resultat direkt zu einer Repräsentation im Frequenzbereich führt. Wie mit MIDI als Input umgegangen wird, geht aus den Quellen nicht hervor. Dass dessen Repräsentationen jedoch auf Grundlage von symbolisch zur Verfügung stehenden Klangkomponenten konstruiert werden müssen, ist eine Parallele zu den Frequenz-Templates, da auch diese auf Basis der mit den Zuständen gespeicherten Tonhöhen erstellt werden. Somit liegt es nahe, dass sich die beiden Konstruktionsmethoden stark ähneln.

Um nun die nötigen Beobachtungswahrscheinlichkeiten für das Zustandsmodell zu ermitteln, wird während des Score-Following-Prozesses die Repräsentation des aktuellen Eingangssignals mit den einzelnen Frequenz-Templates verglichen und der jeweilige Grad der Übereinstimmung unter Verwendung der Kullback-Leibler-Divergenz, einem Maß für die Unterschiedlichkeit zweier Verteilungen, ermittelt. Da davon auszugehen ist, dass die Darbietung dem Stückverlauf folgt, wird die Anzahl der betrachteten Frequenz-Templates aus Effizienzgründen auf einen Bereich rund um die zuletzt detektierte Position im Stück beschränkt (bei geplanten Sprüngen im Notentext, wie etwa Wiederholungen, müssen natürlich trotzdem auch entferntere Sprungziele mitberücksichtigt werden). Die jeweiligen Vergleichsergebnisse sind nun ein Maß dafür, wie wahrscheinlich es ist, dass der Klang des aktuellen Eingangssignals dem Klang eines bestimmten Partitur-Events entspricht. Im Modellsinn ist dies aber genau die Funktion von Beobachtungswahrscheinlichkeiten, denn eine solche sagt aus, wie wahrscheinlich es ist, dass eine bestimmte Beobachtung von einem bestimmten Zustand ausgelöst wurde. Die benötigten Beobachtungswahrscheinlichkeiten für das Zustandsmodell des Score-Followers entsprechen also

den ermittelten Übereinstimmungsmaßen, wenn auch mit kleineren Anpassungen wie etwa einer Wertebereichstransformation.

Ein Hidden-Markov-Modell wäre damit vollständig bestimmt, denn an dieser Stelle wären nun alle dafür notwendigen Modellparameter bekannt. Da es sich im Fall von Antescofo aber um ein Hidden-Hybrid-Markov/Semi-Markov-Modell handelt, werden zusätzlich noch die Wahrscheinlichkeitsverteilungen für die Verweildauern der Semi-Markov-Zustände benötigt. Diese hängen jedoch vom aktuellen Tempo ab, weshalb der Tempoagent neben der Schätzung des aktuellen Tempos auch für die Bereitstellung der Verweildauerverteilungen zuständig ist.

2.2.3 Tempoagent: Schätzung des Tempos

Zur Schätzung des Tempos wird als Modell ein Einheitskreis herangezogen, in dem die Periode des Kreises[1] dem zuletzt geschätzten Tempo entspricht und die Auftrittszeiten der Partitur-Events als Phasenlagen dargestellt werden.

Um nun das aktuelle Tempo zu detektieren, erhält der Tempoagent vom Positionsagenten (siehe Abschnitt 2.2.4) in jedem Verarbeitungszyklus den Auftrittszeitpunkt des gerade detektierten Events (beim allerersten Verarbeitungszyklus wird die Tempoangabe in der Partitur verwendet). Der jeweilige Auftrittszeitpunkt wird anschließend in eine Phasenlage des Kreismodells umgerechnet und in Relation zum zuletzt geschätzten Tempo betrachtet. Weicht die berechnete Phasenlage zu stark von der aufgrund der Partiturangaben und der letzten Temposchätzung erwarteten Phasenlage ab, so hat sich das Tempo verändert und der Tempoagent berechnet entsprechend dem festgestellten Unterschied ein Update der Phasenlage. Darauf aufbauend wird auch die Periode des Kreises aktualisiert, um sie dem geänderten Tempo anzupassen. Die somit korrigierte Temposchätzung wird dann einerseits ausgegeben und andererseits zur Bestimmung der für das Hidden-Hybrid-Markov/Semi-Markov-Modell notwendigen Wahrscheinlichkeitsverteilungen für die Verweildauern der Zustände herangezogen.

[1] Die Periode des Kreises ist jene Zeitdauer, die ein Punkt braucht, um beim Entlangschreiten auf der Kreislinie wieder seine Startposition zu erreichen.

Die zu erwartenden Verweildauern im musikalischen Sinn wurden bereits vom Partitur-Parser als Zustandseigenschaften auf Basis der in der Partitur angegebenen Notenlängen festgehalten. Da diese jedoch in Schlägen angegeben sind, hängen sie vom aktuellen Tempo ab, weshalb dieses bei der Berechnung der entsprechenden Wahrscheinlichkeitsverteilungen mitberücksichtigt werden muss. Als Typ wird die Exponentialverteilung eingesetzt, bei der es sich einerseits um eine nur für positive Werte definierte Verteilung handelt, was dem Charakter einer Verweildauer entspricht, die sich vom Zeitpunkt der Zustandsbesetzung ausschließlich in die Zukunft erstreckt. Andererseits modelliert der exponentiell abfallende Verlauf der Verteilung das zeitliche Verhalten entsprechend der Erwartung, dass die Wahrscheinlichkeit eines Verbleibens im jeweiligen Zustand mit zunehmender Verweildauer immer weiter abnimmt.

Der genaue Verlauf der Verteilungen für verschiedene Zustände hängt nun von deren Notenlängeneigenschaften und dem aktuellen Tempo ab. Steht der jeweilige Zustand beispielsweise für ein Event mit längerem Notenwert, hat dies einen flacheren Verlauf zur Folge, was bedeutet, dass die Wahrscheinlichkeit für das Verbleiben langsamer mit der Zeit fällt, und damit ein Verbleiben im aktuellen Zustand in Relation wahrscheinlicher ist, als im umgekehrten Fall. Zusätzlich hat aber auch das aktuelle Tempo Einfluss auf die Steilheit des Verlaufs, denn abhängig vom Tempo wird auch die Notendauer in Bezug auf die absolute Zeit skaliert. Eine Halbe Note dauert beispielsweise bei halbem Tempo doppelt so lang wie zuvor, weshalb auch die Wahrscheinlichkeitsverteilung der Verweildauer mit niedrigerem aktuellen Tempo flacher ausfällt.

2.2.4 Positionsagent: Detektion der Partiturposition

Bis zum Einsatz des Positionsagenten innerhalb eines Verarbeitungszyklus haben die anderen Architekturkomponenten bereits alle Parameter des zugrunde liegenden Hidden-Hybrid-Markov/Semi-Markov-Modells festgelegt. Deshalb kann der Positionsagent nun auf diese Informationen zurückgreifen und auf deren Basis die aktuelle Position in der Partitur errechnen.

Diese aktuelle Position ist grundsätzlich mit jenem Event in der Partitur gleichzusetzen, das den aktuellen Klang der Darbietung darstellt. Events werden aber wiederum

durch die ihnen zugeordneten Zustände im Zustandsmodell repräsentiert, weshalb die aktuelle Position gleichzeitig auch durch den letzten Zustand der bisherigen Zustandsfolge dargestellt wird. Darum ist die Aufgabe der Positionsdetektion gleichbedeutend mit der Frage, welche Zustandsfolge höchstwahrscheinlich für die bisher festgestellten Beobachtungen verantwortlich ist. Da es sich dabei um eine im Kontext von Hidden-Markov-Modellen häufig auftretende Problemstellung handelt, existieren dafür bereits verschiedene Algorithmen. Im Fall von Antescofo wurde der Viterbi-Algorithmus gewählt (vgl. Abschnitt 1.3.2), welcher jedoch im Hinblick auf die geforderte Echtzeitfähigkeit entsprechend adaptiert wurde.

2.3 Partitursprache und -format

Wie bereits erwähnt, wird die Score-Follower-Software von einer Programmiersprache begleitet. Diese ist notwendig, um die Partitur des zu verfolgenden Stücks in einer für Antescofo verständlichen Form, der sogenannten Antescofo-Partitur, darzustellen. Abschnitt 2.3.1 gibt nun eine Einführung in das Format der Antescofo-Partitur und Abschnitt 2.3.2 beschäftigt sich mit den für diese Arbeit relevanten Teilen der Partitursprache. Für eine umfassende Beschreibung sei auf das Originaldokument [GCEM16] verwiesen, das die Grundlage dieser Abschnitte darstellt.

2.3.1 Antescofo-Partiturformat

Grundsätzlich ist eine Antescofo-Partitur eine gewöhnliche Textdatei. Ihr Inhalt ist aus einer Serie von Sprachelementen der zugehörigen Partitursprache aufgebaut, die der Reihe nach interpretiert werden.

Dafür stehen einerseits Einzelbefehle zur Verfügung, die jeweils einem Ereignis wie einer Note, Pause, Tempoangabe etc. in der Originalpartitur bzw. einer gewünschten Aktion (im Rahmen dieser Arbeit etwa das Umschalten der Registrierung) entsprechen, sowie andererseits Sprachkonstrukte zur Gruppenbildung bzw. Formulierung von komplexeren Funktionen (wie etwa einer Kurve für einen Schweller). Eine Beschreibung der für diese Arbeit relevanten Sprachelemente findet sich in Abschnitt 2.3.2.

Ein Befehl startet immer in einer neuen Zeile der Textdatei mit dem zugehörigen Schlüsselwort, gefolgt von den notwendigen und etwaigen optionalen Parametern. Bei Action-Befehlen kann dem Schlüsselwort zusätzlich ein sogenanntes Delay vorangestellt sein, um eine Verzögerung anzugeben. Gruppenbefehle enden mit der zugehörigen schließenden, geschwungenen Klammer, Einzelbefehle grundsätzlich mit dem Zeilenende, wobei für lange Zeilen ein Backslash (\) verwendet werden kann, um die nächste Zeile für den selben Einzelbefehl mitzuverwenden.

Befehle werden traditionell in Großbuchstaben geschrieben, es können jedoch wenn gewünscht auch Kleinbuchstaben verwendet werden. Außerdem können Leerzeichen und Tabulatoren sowie Leerzeilen beliebig im Sinne einer besseren Lesbarkeit verwendet werden, da sie für die Interpretation der Partitur nicht von Bedeutung sind.

2.3.2 Elemente der Partitursprache

Dieser Abschnitt gibt nun einen Überblick über den zur Verfügung stehenden Wortschatz der Partitursprache. Aufgrund des Umfangs der Sprache sollen allerdings nur die im Rahmen dieser Arbeit relevanten Befehlsgruppen und Befehle besprochen werden (für eine ausführliche Auflistung aller Möglichkeiten siehe [GCEM16]).

Kommentare

Mit Hilfe von Kommentaren besteht die Möglichkeit, zusätzliche Informationen zum besseren Verständnis oder zur leichteren Orientierung in die Antescofo-Partitur einzutragen, die keinerlei Effekt auf den Programmablauf haben. Einzeilige Kommentare beginnen entweder mit zwei Schrägstrichen (`//`) oder einem Semikolon (`;`), mehrzeilige Kommentare beginnen mit der Zeichenfolge `/*` und enden mit `*/`.

Events

Um den Notentext eines Stücks in der Antescofo-Partitur darzustellen, stehen sogenannte Event-Befehle zur Verfügung. Diese Gruppe bietet Kommandos für traditionelle

Partiturereignisse wie etwa Noten, Pausen oder Tempoangaben. In Tabelle 2.1 sind die zur Verfügung stehenden Befehle aufgelistet.

Der Befehl `BPM`[1] dient einerseits dazu, am Beginn der Antescofo-Partitur das zu erwartende Tempo des Stücks anzugeben. Andererseits wird es auch verwendet, um etwaige Tempowechsel innerhalb des Stücks einzutragen.

Einzelne Noten und Akkorde werden mit `NOTE` bzw. `CHORD` in der Partitur dargestellt. Für Pausen wird ebenfalls der Befehl `NOTE` verwendet, jedoch wird hier als Tonhöhe die Zahl Null angegeben. Wird hingegen zwar eine gültige Tonhöhe, aber ein Länge von Null Schlägen spezifiziert, wird dies als Vorschlagnote interpretiert.

Mit dem Kommando `TRILL` lassen sich Triller und ähnliche Elemente darstellen. Der Befehl kann dabei als Angabe eines Tonvorrats für die Dauer des Events verstanden werden. Im Zuge des Score-Following-Prozesses wird dann nur darauf geachtet, dass die gespielten Töne dem angegeben Tonvorrat entsprechen; Reihenfolge, Dauer und Wiederholungsanzahl der einzelnen Töne werden jedoch nicht berücksichtigt [Con10, S. 7]. So kann dieses Kommando sowohl für Triller als auch für Tremoli, Mordente, Doppelschläge etc. verwendet werden, indem Größe und Zusammensetzung des Tonvorrats entsprechend gewählt werden.

Der Befehl `MULTI` kann ebenfalls als Definition eines Tonvarrats gesehen werden. Allerdings wird dieser (anders als beim `TRILL`-Kommando) einerseits nicht durch einzelne Tonhöhen, sondern als Tonhöhenbereich angegeben, und andererseits wird die Reihenfolge der auftretenden Töne bei Verwendung dieses Kommandos vom Score-Follower berücksichtigt. Er ist etwa zur Spezifikation von Glissandi gedacht.

Zusätzlich können an alle Event-Befehle ein oder mehrere sogenannte Labels angehängt werden. Das sind benutzerdefinierte, alphanumerische Markierungen des Events, die etwa als Sprungziel, z.B. zur Darstellung von Wiederholungen, verwendet werden können.

Außerdem stehen für alle Events mit Ausnahme von `BPM` die in Tabelle 2.2 beschriebenen Attribute zur Verfügung, um die jeweiligen Events näher zu spezifizieren. Solche Attribute beginnen generell mit dem Zeichen `@` und können vor und/oder nach dem Label notiert werden.

[1] Die Abkürzung »bpm« steht für »beats per minute«, zu Deutsch »Schläge pro Minute«.

BPM	Tempoangabe	
Syntax	`BPM value`	
Parameter	value	Anzahl der Schläge pro Minute
Beispiele	`BPM 60` `BPM 90`	
NOTE	einzelne Note oder Pause	
Syntax	`NOTE pitch duration`	
Parameter	pitch duration	Tonhöhe als MIDI-Nr., MIDICent oder Tonname Event-Dauer in Schlägen

Beispiele	`NOTE 60 1.0` `// Viertelnote c' (1 Schlag)`
	`NOTE 6600 0.5` `// Achtelnote fis' (1/2 Schlag)`
	`NOTE Ab4 1/4` `// 16telnote as' (1/4 Schlag)`
	`NOTE E4 0` `// Vorschlagnote e'`
	`NOTE 0 1/2 "mylabel"` `// Achtelpause (1/2 Schlag)`	

CHORD	einzelner Akkord	
Syntax	`CHORD (pitchlist) duration`	
Parameter	pitchlist duration	Tonhöhen als MIDI-Nr., MIDICent oder Tonnamen Event-Dauer in Schlägen
Beispiele	`CHORD (D4 F4 A4) 4` `// Moll-Akkord auf d'`	
TRILL	Triller oder Tremolo	
Syntax	`TRILL (pitchlist) duration` `TRILL ( (pitchlist) (pitchlist) ) duration`	
Parameter	pitchlist duration	Tonhöhen als MIDI-Nr., MIDICent oder Tonnamen Event-Dauer in Schlägen

Beispiele	`TRILL (G4 A4) 2` `// Triller auf g'`
	`TRILL (F#4) 2` `// Tremolo auf fis'`
	`TRILL (F#4 E4 D4) 2` `// Doppelschlag auf e'`
	`TRILL ( (E4 G#4) (F#4 A4) ) 2` `// Akkord-Triller`	

MULTI	Glissando	
Syntax	`MULTI ( (pitchlist) -> (pitchlist) ) duration`	
Parameter	pitchlist duration	Tonhöhen als MIDI-Nr., MIDICent oder Tonnamen Event-Dauer in Schlägen
Beispiele	`MULTI ( (D4) -> (A4) ) 2` `// Glissando d' nach a'` `MULTI ( (D4 A4) -> (A4 E5) ) 2` `// Glissando in Quinten`	

Tabelle 2.1: Elemente der Partitursprache – Events

@fermata	Fermate
Syntax	`@fermata`
Erklärung	Ein Event mit Fermate kann länger dauern als notiert, ohne die Tempo-Detektion zu beeinflussen.
Beispiele	`NOTE C4 2.0 @fermata`
@pizz	Pizzicato
Syntax	`@pizz`
Erklärung	Streicher-Pizzicati können mit diesem Attribut versehen werden, um die Partiturverfolgung zu unterstützen.
Beispiele	`NOTE F5 1/2 @pizz "mylabel"`
@hook	Event-Detektion erzwingen
Syntax	`@hook`
Erklärung	Der Score-Follower wird gezwungen, an mit diesem Attribut markierten Events zu warten, bis sie detektiert werden.
Beispiele	`NOTE Bb3 1/4 @hook`
@jump	Sprungmöglichkeit
Syntax	`@jump label` `@jump label1, label2, ...`
Parameter	label ein für ein Ziel-Event definiertes Label
Erklärung	Auf ein Event mit diesem Attribut kann entweder das direkt danach notierte Event folgen, oder eines jener Events, die mit den angegebenen Labels versehen wurden.
Beispiele	`NOTE A4 1.0 @jump "mylabel"` `NOTE A4 1.0 @jump "mylabel", "myotherlabel"`

Tabelle 2.2: Elemente der Partitursprache – Event-Attribute

Actions

Um Aktionen zu einem bestimmten Zeitpunkt in der Partitur auszulösen, wie etwa die Umschaltung der Registrierung, werden sogenannte Action-Befehle verwendet. Einen Auszug der für die Zwecke dieser Arbeit interessanten Befehle zeigt Tabelle 2.3.

Actions werden grundsätzlich zeitgleich mit der Detektion des direkt davor notierten Events oder dem Definitionszeitpunkt der direkt vorangehenden Action ausgeführt, es

Message	Nachricht an Max/MSP bzw. Pure Data	
Syntax	`receiver value [value [...]]`	
Parameter	receiver	Name des Ziel-Receive-Objekts in Max/MSP bzw. Pure Data
	value	zu sendender Wert

Beispiele	`receiver1 4`
	`receiver2 "this is a message"`
	`receiver3 this is also valid`	

OSCSEND	Ausgehenden OSC-Kanal definieren	
Syntax	`OSCSEND name [host] : port prefix`	
Parameter	name	Name des OSC-Kanals
	host	IP-Adresse des Empfängers (optional)
	port	Ziel-Port des Empfängers
	prefix	OSC-Adresse
Beispiele	`OSCSEND mychannel/mymsg www.example.com : 3456 "mymsg"`	
OSC Message	OSC-Nachricht senden	
Syntax	`name value`	
Parameter	name	Name eines deklarierten ausgehenden OSC-Kanals
	value	zu sendender Wert
Beispiele	`mychannel/mymsg "test osc message"`	
CURVE	Kontinuierliche Werteveränderung	
Syntax	`CURVE receiver start, end duration`	
Parameter	receiver	Name des Ziel-Receiver-Objekts in Max/MSP bzw. Pure Data
	start	Startwert der Kurve
	end	Zielwert der Kurve
	duration	Dauer der Werteveränderung von start nach end
Beispiele	`CURVE receiver1 0.5, 1.3 2s`	
Anmerkung	Für komplexere Syntaxvarianten und Konfigurationsmöglichkeiten (z.B. Interpolationsmethoden etc.) siehe [GCEM16, Kapitel 5.5].	
GROUP	Gruppierung mehrerer Actions	
Syntax	`GROUP [name] [attr [...]], { actions }`	
Parameter	name	Name der Gruppe
	attr	Attribute (z.B. Synchronisationsstrategie etc.)
	actions	zu gruppierende Actions, eine je Zeile
Beispiele	`GROUP @tight { 2.5 receiver1 4 }`	

Tabelle 2.3: Elemente der Partitursprache – Actions

sei denn, es wird dem jeweiligen Action-Befehl ein sogenanntes Delay vorangestellt. In diesem Fall wird die Ausführung der Action um das angegebene Delay verzögert. Solche Delays können entweder in Sekunden bzw. Millisekunden (z.B. `1s` bzw. `1000ms`) oder in Schlägen (z.B. `2`) angegeben werden.

Außerdem können Actions ähnlich wie Events mit Labels markiert werden. Im Falle von Actions wird dazu das Attribut `@name` verwendet, z.B. `@name "myaction"`.

Es besteht auch die Möglichkeit, mehrere Actions in einer übergeordneten `GROUP`-Action zusammenzufassen. Das Verhalten von Actions in solchen Gruppen kann dann zusätzlich mittels verschiedener Attribute beeinflusst werden. Im Rahmen dieser Arbeit sind besonders zwei dieser Einstellmöglichkeiten interessant: die explizite Wahl sogenannter Synchronisations- und Fehlerbehandlungsstrategien (für weiterführende Informationen sei auf die Quelle der nachfolgenden Beschreibungen [GCEM16, S. 105-114] verwiesen).

Zur Beeinflussung der Auslösungssynchronisation von Actions mit der aktuellen Position im Stück stehen einige Attribute zur Verfügung, wovon die folgenden drei im Rahmen dieser Arbeit relevant erscheinen.

`@loose` Die Auslösung der Actions orientiert sich ausschließlich an den Temposchätzungen, die mit jedem detektierten Event aktualisiert werden. Dadurch werden relativ zum Tempo angegebene Delays, die über mehrere Events hinweg andauern, immer wieder angepasst. Dies ist die Standardstrategie und gilt implizit für einzelne Actions außerhalb einer Gruppe, sowie für Gruppen, bei denen keine andere Strategie angegeben ist.

`@tight` In diesem Fall richtet sich die Auslösung nach den Detektionszeitpunkten der Events. Fällt das Ende der angegebenen Delay-Dauer mit dem Zeitpunkt eines Events laut Partitur zusammen, wird die Action bei der Detektion dieses Events ausgelöst; ist die Delay-Dauer etwas länger, findet die Auslösung entsprechend nach dem Event statt. Das Verhalten ist also ähnlich jenem einer einzelnen Action, die in der Partitur direkt nach dem Event eingetragen wäre.

`@target` Diese Strategie aktualisiert das interne Tempo der `GROUP`-Action nicht direkt, sondern gleicht es durch Interpolation über eine definierbare Zeitspanne hinweg an das detektierte Tempo der Darbietung an.

Die Fehlerbehandlungsstrategie kann mit den Attributen `@global` und `@local` eingestellt werden. Sie entscheidet darüber, wie mit Actions umgegangen wird, wenn das Event (oder eine vorherige Action), an das sie gebunden wurden, nicht detektiert wurde. Im Fall von `@local` wird die Action ignoriert, bei `@global` wird sie beim nächsten detektierten Event nachgeholt. Im Rahmen dieser Arbeit ist die Verwendung dieser Attribute jedoch nicht nötig, da ein Nachholen der verpassten Actions generell als sinnvoll erscheint, und dies ohnehin der Standardstrategie für Gruppen der obersten Ebene sowie einzelner Actions entspricht. Verschachtelte Gruppen erben dabei von den jeweils umschließenden [AntOD].

Variablen

Wie jede Programmiersprache erlaubt auch die Antescofo-Partitursprache die Verwendung von Variablen, um konstante oder berechnete Ausdrücke zu speichern. Variablen beginnen immer mit dem Zeichen `$` und sind standardmäßig global in der ganzen Partitur verfügbar (die Deklaration von lokalen Variablen ist in bestimmten Fällen durch Voranstellen von `@local` möglich). In Tabelle 2.4 sind die wichtigsten Variablenoperationen aufgelistet.

Zuweisung	Zuweisung eines Wertes zu einer Variable
Syntax	`LET` *`$var := expression`*
Parameter	var Name der Variable expression zuzuweisender Ausdruck
Beispiele	`LET $x := 4` `LET $text := "variable y is"`
Verwendung	Verwendung des Wertes einer Variable
Syntax	*`$var`*
Parameter	var Name der Variable
Beispiele	`LET $y := $x+1 // $y enthält Wert 5` `receiver1 $text $y // Nachricht: "variable y is 5"`

Tabelle 2.4: Elemente der Partitursprache – Variablen

Da Variablenzuweisungen eine Spezialform von Actions darstellen, kann auch hier ein Delay vorangestellt werden, um die Ausführung zu verzögern.

Steuerkommandos

Um die Parameter des Score-Followers nicht nur von Max/MSP bzw. Pure Data aus setzen, sondern auch direkt aus der Partitur heraus ändern zu können, existieren für diesen Zweck zusätzlich zu den Sprachelementen für Notentextdarstellung und Aktionsdefinition auch Steuerkommandos.

Diese Steuerkommandos lauten grundsätzlich gleich wie ihre Gegenparts in Max/MSP bzw. Pure Data, mit dem einzigen Unterschied, dass ihnen die Zeichenfolge `antescofo::` vorangestellt wird, also z.B. `antescofo::nextevent`. Eine Ausnahme stellt der Varianzparameter (siehe Abschnitt 2.2.2) dar, dessen Befehl `VARIANCE` lautet. Für eine Auflistung der verfügbaren Kommandos sei auf die Beschreibungen der Steuermessages in Abschnitt 2.4.2 und Abschnitt 2.4.3 verwiesen.

2.4 Verwendung

Während der Verwendung von Antescofo zur Partiturverfolgung an sich sind im Optimalfall keine Eingriffe mehr nötig, es sei denn, es treten Detektionsschwierigkeiten oder dergleichen auf, die einer Korrektur bedürfen, denn die eigentliche Arbeit ist in der Vorbereitungsphase zu leisten.

In einem ersten Schritt muss die Partitur des zu verfolgenden Stücks in das Antescofo-Partiturformat überführt und die Zeitpunkte für gewünschte Aktionen wie etwa Registrierungsumschaltungen dort eingetragen werden (Abschnitt 2.4.1). Anschließend sind die Parameter des Score-Followers auf das Stück und das zu verfolgende Instrument abzustimmen sowie, falls ein Audiosignal verwendet wird, eine Kalibrierung durchzuführen (Abschnitt 2.4.2), bevor der Score-Follower schlussendlich gestartet und verwendet werden kann (Abschnitt 2.4.3).

2.4.1 Erstellen der Antescofo-Partitur

Zur Erstellung einer Antescofo-Partitur für das zu verfolgende Stück gibt es zwei Möglichkeiten. Zum einen kann diese händisch erstellt werden, indem ein Texteditor verwendet wird, um das Notenmaterial der zu verfolgenden Stimmen (inklusive relevanter Vortragszeichen) Note für Note, Akkord für Akkord, durch geeignete Event-Kommandos auszudrücken. Diese Vorgangsweise ist jedoch sehr aufwändig, deshalb bietet das Softwarepaket zum anderen unter Verwendung des Zusatzprogramms AscoGraph die Möglichkeit, Partituren in einem bereits vorliegenden, digitalen Format zu importieren und dabei automatisch in das Antescofo-Format umzuwandeln [GCEM16, S. 21].

Partitur-Import

Für einen Partitur-Import werden grundsätzlich MIDI- und MusicXML-Dateien als Ausgangsmaterial unterstützt. Die Antescofo-Dokumentation empfiehlt jedoch, wenn möglich MusicXML-Dateien zu verwenden, da diese auch explizite Darstellungen von komplexeren Ereignissen wie etwa Trillern oder Glissandi unterstützen, und diese damit auch korrekt konvertiert werden können.

Zum Importieren einer Datei genügt es, sie einfach per Drag-and-Drop in die Programmoberfläche von AscoGraph zu ziehen, allerdings sollte bei den verwendeten MusicXML-Dateien auf einige Dinge geachtet werden, um eine möglichst fehlerfreie Umwandlung zu ermöglichen: Es sollten keine speziellen Notenköpfe (wie etwa rautenförmige für Flageolett-Notation) verwendet, und grafische sowie versteckte Elemente vermeiden werden (für nähere Informationen siehe [GCEM16, S. 21][1]).

Da die Umwandlung einerseits vom Inhalt der MusicXML-Dateien, andererseits jedoch auch vom für deren Erstellung verwendeten Programm abhängt (der Konverter ist auf das Notensatzprogramm »Finale« optimiert), wird empfohlen, die Antescofo-Partitur nach der Konversion zu überprüfen und etwaige Fehler vor allem bei komplexeren Ereignissen wie Trillern o.Ä. manuell zu korrigieren.

[1] Zumindest der dortige Hinweis auf eventuelle Probleme mit mehreren Ebenen ist jedoch laut einem Gespräch mit Antescofo-Projektleiter Arshia Cont inzwischen überholt.

Eintragen der Aktionen

In die erstellte Antescofo-Partitur können anschließend die gewünschten Aktionen eingetragen werden. Im üblichen Umfeld von Antescofo wären dies Informationen zur Steuerung der elektronischen Parts eines Musikstücks, im Fall dieser Arbeit sind die gewünschten Aktionen die Steuerung von Orgelparametern wie etwa die Umschaltung der Registrierung.

Für die Registrierungsumschaltung einer digitalen Orgel ist es notwendig, zum richtigen Zeitpunkt die korrekte MIDI-Nachricht an die Orgel zu senden. Da diese Nachricht im Laborsystem generiert wird (siehe dazu Abschnitt 3.1.3), muss nur der richtige Auslösezeitpunkt innerhalb der Antescofo-Partitur definiert werden. Dafür genügt es prinzipiell, eine Message-Action (siehe Abschnitt 2.3.2) zwischen jenen beiden Events einzufügen, zwischen denen die Umschaltung geschehen soll. Bei Ausführung dieser Action wird die Message dann in Max/MSP bzw. Pure Data empfangen und löst den Sendevorgang aus.

Dabei ist allerdings zu beachten, dass der genaue Zeitpunkt der Ausführung enorm relevant ist. Im Normalfall ist es wünschenswert, einen Registrierungswechsel kurz nach der letzten Note eines Registrierungsabschnitts vorzunehmen, um den Ausklang der Note nicht zu beeinflussen. Um dies zu erreichen, muss einerseits die Reaktionszeit der Orgel bekannt sein und der Auslösezeitpunkt darauf abgestimmt werden, andererseits ist jedoch auch anzunehmen, dass das Gehör eine gewisse Toleranz in Bezug auf den Zeitpunkt der Klangveränderung erlaubt (siehe dazu Abschnitt 4.3.1 bzw. Abschnitt 4.2).

2.4.2 Einstellen des Score-Followers

Neben der Erstellung der Antescofo-Partitur ist auch der Score-Follower an sich vorzubereiten. Um ihn verwenden zu können, muss zuerst das entsprechende Objekt `antescofo~` in Max/MSP bzw. Pure Data instanziiert und eingestellt werden. Nachfolgend werden die im Rahmen dieser Arbeit relevanten Einstellungen auf Basis der im Softwarepaket enthaltenen Tutorials für Max/MSP [AntTS] beschrieben.

Inlets und Outlets

Wie die meisten Objekte in Max/MSP bzw. Pure Data kommuniziert auch das Antescofo-Objekt primär über sogenannte Inlets und Outlets[1]. Welche der möglichen Inlets und Outlets für die Verwendung zur Verfügung stehen sollen, wird mit Argumenten bei der Instanziierung des Objektes festgelegt.

Mit Hilfe des Arguments `@inlets`, gefolgt von den gewünschten Optionen, wird eingestellt, welche Art von Input man an das Objekt senden wird. Gibt man keinen Parameter an, wird standardmäßig Audio-Input angenommen. Zur Verwendung von MIDI-Input wäre das Objekt beispielsweise mit `antescofo~ @inlets MIDI` zu instanziieren. Zur Auswahl stehen die in Tabelle 2.5 genannten Parameter.

Parameter	erwartetes Eingangssignal
`KL`	Audiosignal
`MIDI`	MIDI-Nachrichten
`myinput`	externes Pitch-Detektor-Ergebnis (mit benutzerdefinierter Bezeichnung)

Tabelle 2.5: Instanziierungsargumente für Inlets

Ähnlich verhält es sich mit der Definition von Outlets. Abgesehen von den standardmäßig vorhandenen Outlets (Haupt-Outlet, aktuelles Tempo, aktuelles Label und Partiturladestatus) wird mit dem Argument `@outlets` eingestellt, welche zusätzlichen Outlets mit welchen Daten zur Verfügung gestellt werden sollen. Um zum Beispiel zusätzliche Outputs für die aktuelle Partiturposition und ein Endsignal zu erhalten, würde man das Objekt mit `antescofo~ @outlets beatnum endbang` erstellen. Tabelle 2.6 zeigt eine Liste der möglichen Optionen.

[1] In Max/MSP bzw. Pure Data sind Inlets bzw. Outlets eingehende bzw. ausgehende Objektschnittstellen, die dazu dienen, Objekte miteinander zu verbinden und so Daten zu transportieren.

Argument	ausgegebene Daten
`notenum`	aktuelle Tonhöhe als MIDI-Nummer
`beatnum`	Partiturposition in Schlägen
`anteIOI`	erwartete Dauer bis zum nächsten Event
`certainty`	Wert für die Sicherheit der aktuellen Schätzung
`scoretempo`	notiertes Tempo für die aktuelle Stelle
`endbang`	Signal beim Erreichen des letzten hörbaren Events
`velocity`	aus Ansatzenergie geschätzte MIDI-Velocity
`midiout`	aus dem aktuellen Event generierte MIDI-Message
`TDIST`	Dauer bis zur nächsten Action

Tabelle 2.6: Instanziierungsargumente für Outlets

Parametereinstellungen

Ist das Objekt instanziiert, akzeptiert es, wie in Max/MSP bzw. Pure Data üblich, an seinem ersten Inlet sogenannte Messages[1], um Einstellungen vorzunehmen. Tabelle 2.7 listet einen Auszug der möglichen Messages auf, die im Rahmen dieser Arbeit relevant erscheinen. Für eine vollständige Liste sei auf die Max/MSP-Tutorials verweisen.

Syntax	Wertebereich	Initialwert	Einstellung
`calibration` *value*	0, 1	0 (aus)	Kalibriermodus aktivieren
`calibration 1` *value*	0 - 12700	6900 (a')	Referenzton per MIDI-Nr./MIDICent setzen
`tune` *value*	positiv	440.0	Stimmung des Score-Followers in Hz
`nofharm` *value*	positiv	10	Anzahl harmonischer Teiltöne für Analyse
`harmlist` *list*	Liste	—	Liste der Teiltöne für Analyse
`temposmoothing` *value*	0 - 1	0.1	Einbeziehung vergangener Temposchätzungen
`preventzigzag` *value*	on, off	on (ein)	Positionskorrekturen nur vorwärts
`info`	—	—	Ausgabe der aktuell gesetzten Werte
`version`	—	—	Ausgabe der verwendeten Version

Tabelle 2.7: Messages zur Einstellung des Score-Followers

Außerdem existiert ein Parameter namens »Varianz«, mit dem die Toleranz der Detektion gegenüber Tonhöhenabweichungen beeinflusst werden kann (siehe Abschnitt 2.2.2). Ein Setzen dieses Parameters mit der in den Quellen angegebenen Max/MSP-Message hat jedoch (zumindest in der verwendeten Antescofo-Version) keine Auswirkung, wes-

[1] Messages sind eine Form des Datenaustauschs und des Zugriffs auf Objektfunktionen in Max/MSP bzw. Pure Data.

halb dies mit dem Befehl `VARIANCE`, gefolgt vom gewünschten Varianzwert, am Beginn der verwendeten Antescofo-Partitur erfolgen muss. Dabei sind Werte zwischen 0 und 1 möglich, als Standardeinstellung wird ein Varianzwert von 0.3 verwendet.

Kalibrierung

Vor Beginn der Verwendung ist es im Fall von Audio-Input außerdem notwendig, eine Kalibrierung durchzuführen, um die bestmögliche Performance des Score-Followers zu gewährleisten. Dazu muss zunächst mit der Message `calibration 1` der Kalibriermodus aktiviert werden. Dies hat zu Folge, dass am Haupt-Outlet des Externals zusätzlich eine Liste mit Kalibrationswerten ausgegeben wird, die mit dem Objekt `route calibration` herausgefiltert werden können.

Der erste Wert dient zur Einstellung des Audiosignalpegels. Der Kalibrationswerten sollte beim Erklingen von Nutzsignal einen Wert von über 0.75 aufweisen, ist gerade kein Nutzsignal zu hören, sollte er sich unter 0.5 bewegen. Ist dies nicht der Fall, so sollte der Pegel des Audiosignals angepasst werden.

Mit dem zweiten Wert steht ein Hilfsmittel zur Überprüfung der Stimmung des verwendeten Instruments zur Verfügung. Standardmäßig verwendet Antescofo den Kammerton a' (MIDICent-Wert 6900) als Referenzton entsprechend der mit der `tune`-Message einstellbaren Stimmung (Standardwert 440.0 Hz). Dieser Referenzton kann jedoch beispielsweise mit der Message `calibration 1 7000` auf b' geändert werden. Spielt das Instrument diesen Ton, so sollte der Kalibrationswert bei korrekter Stimmung etwa bei 0.5 liegen. Falls nicht, muss entweder das Instrument nachgestimmt oder eventuell die Stimmung des Score-Followers angepasst werden.

2.4.3 Steuerung des Score-Followers

Damit das gewünschte Musikstück verfolgt werden kann, muss zunächst die zugehörige Antescofo-Partitur mit der Message `score` geladen werden. Falls der Score-Follower nicht ohnehin bereits aktiviert ist, sollte dies mit `suivi 1` erledigt werden. Anschließend startet man den Score-Follower mit der Message `start`.

Sollte man nicht am Beginn des Stückes starten oder im Falle von Proben an eine bestimmte Stelle springen wollen, stehen dafür ebenfalls eine Reihe von Messages, wie etwa `startfromlabel` oder `gotolabel`, bereit.

Die folgende Tabelle 2.8 zeigt eine Zusammenfassung der verfügbaren Messages zur Steuerung des Score-Followers.

Syntax	Wertebereich	Initialwert	Einstellung
`score [file]`	—	—	Partitur laden (Fenster oder *file* direkt)
`read [file]`	—	—	Partitur laden (Fenster oder *file* direkt)
`reloadscore`	—	—	Partitur erneut laden
`suivi value`	0, 1	1 (ein)	Score-Follower de-/aktivieren
`start`	—	—	Verfolgung starten
`stop`	—	—	Verfolgung stoppen
`actions value`	on, off	on (ein)	Ausführung von Actions de-/aktivieren
`startfromlabel label`	Zeichenfolge	—	Verfolgung bei *label* starten[1]
`scrubtolabel label`	Zeichenfolge	—	zu *label* vorspulen[2]
`gotolabel label`	Zeichenfolge	—	zu *label* springen[3]
`startfrombeat beatnr`	positiv	—	Verfolgung bei *beatnr* starten[1]
`scrubtobeat beatnr`	positiv	—	zu *beatnr* vorspulen[2]
`gotobeat beatnr`	positiv	—	zu *beatnr* springen[3]
`nextevent`	—	—	zum nächsten hörbaren Event springen
`previousevent`	—	—	zum vorhergehenden hörbaren Event springen
`nextlabel`	—	—	zum nächsten Label springen
`previouslabel`	—	—	zum vorhergehenden Label springen
`nextaction`	—	—	zur nächsten Action springen
`getcues`	—	—	Ausgabe einer Liste der definierten Labels
`ascograph value`	open, close	—	AscoGraph-Editor öffnen/schließen

Tabelle 2.8: Messages zur Steuerung des Score-Followers

[1] Vorangegangene Actions werden ausgewertet aber nicht ausgeführt.
[2] Dazwischen liegende Actions werden sowohl ausgewertet als auch ausgeführt.
[3] Dazwischen liegende Actions werden weder ausgewertet noch ausgeführt.

3 Laborsystem

Aufbauend auf Antescofo wurde ein Laborsystem zusammengestellt, das sowohl aus Hardware-Komponenten als auch aus bereits existierenden sowie im Rahmen dieser Arbeit entwickelten Software-Komponenten besteht.

Abschnitt 3.1 widmet sich nun der Beschreibung der einzelnen Komponenten, während in Abschnitt 3.2 die verschiedenen Verwendungsszenarien vorgestellt werden, in denen das Laborsystem eingesetzt wird.

3.1 Komponenten

Als erste Übersicht des Laborsystems zeigt Abbildung 3.1 dessen Komponenten. Die Verbindungen der einzelnen Blöcke stellen dabei eine Zusammenfassung der verwendeten Verschaltungsmöglichkeiten dar, wobei je nach Verwendungsszenario nur die dafür nötigen Komponenten verwendet und verschaltet wurden (siehe dazu Abschnitt 3.2).

Genaue Daten zu den im nachfolgenden Text erwähnten Geräten und Software-Programmen, wie etwa Modellbezeichnungen oder Software-Versionen, sind in Abschnitt A.2 zusammengefasst.

3.1.1 Hardware

Insgesamt umfasst das Laborsystem vier Hardware-Komponenten: eine digitale Orgel, ein zugehöriges Wiedergabesystem, ein Audio-/MIDI-Interface sowie einen Laborcomputer zur Ausführung der Software-Komponenten.

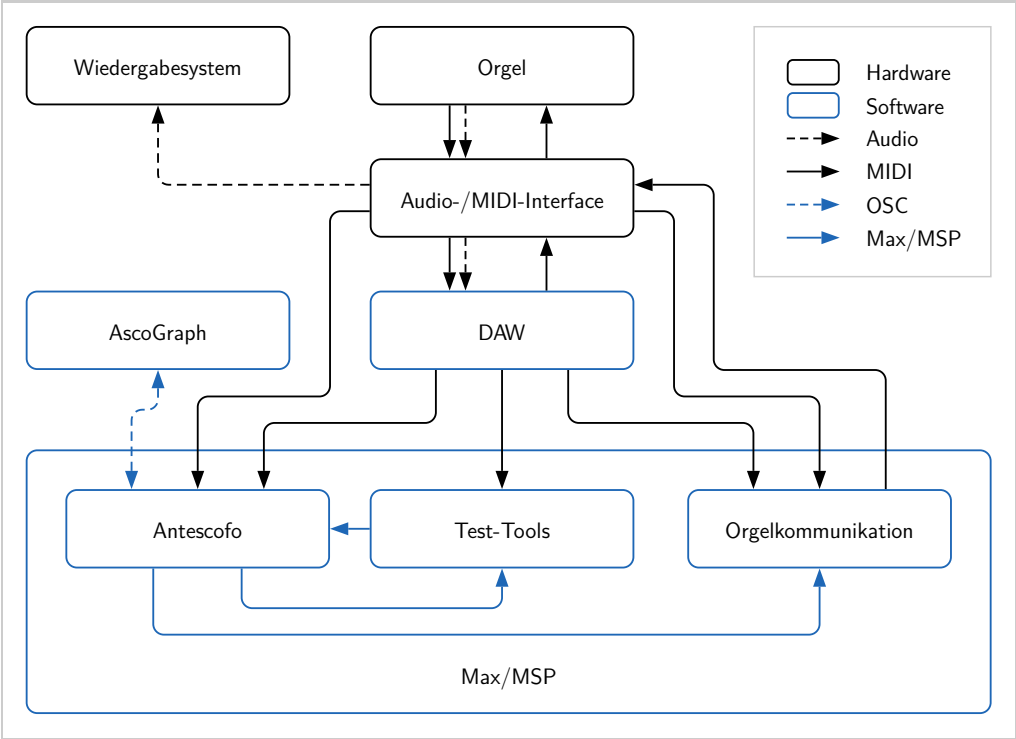


Abbildung 3.1: Blockdiagramm der Laborsystemkomponenten

Orgel

Die wichtigste Hardware-Komponente ist natürlich die Orgel selbst, deren automatische Registrierungsumschaltung Ziel dieser Arbeit ist. Sie stellt im Anwendungsszenario (vgl. Abschnitt 3.2) einerseits den Startpunkt der Verschaltungskette dar, indem die entsprechend dem Spiel einer Organistin bzw. eines Organisten generierten Audio- und MIDI-Signale von ihr ausgehen. Andererseits müssen jedoch auch die Steuerungssignale wieder zu ihr zurück gelangen, um die Registrierungsumschaltung auszulösen.

Verwendet wurde im Rahmen dieser Arbeit eine digitale Konzertorgel der Firma Rodgers mit der Modellbezeichnung Trillium Masterpiece 968, die sich am Institut für Kirchenmusik und Orgel der Universität für Musik und darstellende Kunst Graz befand.

Grundsätzlich ist die Verwendung der entwickelten Registrierungsumschaltung jedoch bereits in der derzeitigen Version mit jeder beliebigen elektronischen Orgel möglich,

solange diese in der Lage ist, Steuerungsdaten zur Registrierungsumschaltung zu empfangen sowie die Tasten- und Pedalanschläge als MIDI-Daten auszugeben (derzeit wird nur MIDI als Eingangssignal für das entwickelte System unterstützt, in Zukunft könnte eventuell auch Audio verwendet werden, siehe dazu Abschnitt 4.1).

Audio-/MIDI-Interface

Als Schnittstelle zwischen der digitalen Orgel und dem Laborcomputer zur Ausführung der Software-Komponenten war zudem ein Audio- und MIDI-Interface notwendig.

Praktischerweise wurde die Rodgers-Orgel im Rahmen der Orgelforschung am Institut für Kirchenmusik und Orgel vor einiger Zeit mit einem MADI-Wandler und einem MIDI-Embedder erweitert. Aufgrund dessen stand über eine optische Schnittstelle ein digitaler MADI-Datenstrom zur Verfügung, der Audiokanäle und MIDI-Signal gleichzeitig übertrug. Daher konnte ein RME Madiface als Interface verwendet werden, um alle notwendigen Daten gesammelt am optischen Eingang entgegen zu nehmen und in den Laborcomputer einzuspeisen.

Außerdem bot das RME Madiface eine zusätzliche MADI-Schnittstelle mit BNC-Steckverbindern, die verwendet werden konnte, um das Audiosignal der Orgel mit Hilfe des zugehörigen Softwaremixers ohne Umweg durch den Laborcomputer zum Wiedergabesystem durchzuschleifen.

Laborcomputer

Als Laborcomputer zur Ausführung der Software-Komponenten war im Rahmen dieser Arbeit ein Gerät der Firma Apple notwendig, da das Programm AscoGraph nur unter Macintosh als Betriebssystem (kurz MacOS) lauffähig ist.

Je nach Verwendungsszenario wurden zwei verschiedene Geräte verwendet. Für das Aufnahme- sowie das Anwendungsszenario (vgl. Abschnitt 3.2) bot sich der schon im Orgelsaal befindliche Mac Mini an (Mac mini A), für die Durchführung der Tests außerhalb des Orgelsaals wurde ein anderes Gerät (Mac mini B) benutzt, das jedoch eine nur leicht abweichende Konfiguration aufwies.

Sollte man sich jedoch entscheiden, bei einer zukünftigen Version des Systems (zumindest im Anwendungsszenario) auf die grafische Darstellung der Partitur und des Score-Following-Vorganges mittels AscoGraph zu verzichten, könnte auch ein beliebiger Computer mit Linux-Betriebssystem verwendet werden, da ebenso eine Antescofo-Version für die Verwendung mit Pure Data unter Linux existiert. Dazu müssten allerdings die als Teil dieser Arbeit für Max/MSP entwickelten Programmteile nach Pure Data portiert werden.

Wiedergabesystem

Zur üblichen Ausstattung der Rodgers-Orgel gehört zudem ein Wiedergabesystem, das auch aus dem Hause des Herstellers stammt. Dieses wurde ebenfalls im Rahmen der Orgelforschung am Institut für Kirchenmusik und Orgel mit einem zwischengeschalteten Rechner erweitert, der eine flexiblere Konfiguration des Wiedergabesystems ermöglichen soll.

Normalerweise ist die Orgel direkt über die optische MADI-Leitung mit der Wiedergabeeinheit verbunden. Für die entwickelte Registrierungsumschaltung muss jedoch auch der Laborrechner das Orgelsignal empfangen können. Deshalb wurde die optische Verbindung an das Audio-/MIDI-Interface angeschlossen und das Audiosignal der Orgel über die zusätzliche BNC-Schnittstelle an das Wiedergabesystem weitergeleitet.

Eine solche Verschaltung ermöglicht demnach sowohl die Live-Wiedergabe des Orgelklanges, als auch die Verwendung des Laborcomputers für die Wiedergabe von aufgezeichnetem Material zu Testzwecken.

3.1.2 Bestehende Software

Neben der Verwendung des zentralen Softwarepakets Antescofo und dessen Begleitprogramm AscoGraph wurden für Aufgaben, die von bereits existierenden Computerprogrammen übernommen werden konnten, ebendiese Programme herangezogen.

Max/MSP

Als wichtigster Teil dieser Gruppe ist zunächst die Programmier- und Laufzeitumgebung Max/MSP zu nennen, in der durch grafische Verbindung von sogenannten Objekten mittels Cords (dt. Kabel) sogenannte Patches erstellt und danach ausgeführt werden können.

Dabei werden zwei Signaltypen unterschieden, Audiosignale und sogenannte Messages für den Austausch sonstiger Daten, die quasi von oben nach unten durch die Kabel fließen und mit Hilfe der Objekte bearbeitet werden können. Dasselbe gilt auch für die im Rahmen dieser Arbeit nicht verwendete, aber sehr ähnliche Umgebung Pure Data.

Zudem können die beiden Umgebungen jeweils mit von Dritten bereitgestellten Objekten, sogenannten Externals, erweitert werden. Auch Antescofo wurde als ein solches External für die Verwendung mit Max/MSP (unter MacOS) bzw. Pure Data (unter Linux) entwickelt. Da AscoGraph, die grafische Begleitanzeige zu Antescofo, jedoch nur auf MacOS lauffähig ist, fiel die Entscheidung auf Max/MSP, bei Verzicht auf diese Komponente wäre ein Wechsel zu Pure Data jedoch durchaus möglich (vgl. dazu auch Abschnitt 3.1.1).

DAW

Um die Versuchsphase möglichst effizient zu gestalten, hat eine Aufnahme stattgefunden, um die Versuche in der Folge unabhängig von einer Organistin bzw. einem Organisten mit dem aufgezeichneten Material durchführen zu können (siehe dazu Abschnitt 3.2.1 bzw. Abschnitt 3.2.2).

Dazu wurde sowohl bei dieser Aufnahme, als auch für Wiedergaben und weitere Aufnahmen in der Testphase die DAW Nuendo bzw. dessen kleinerer Bruder Cubase verwendet, die zueinander direkt kompatibel sind. Neben den üblichen Funktionen wie Aufnahme und Wiedergabe von Audio- und MIDI-Material, erwiesen sich die folgenden Funktionen als besonders hilfreich:

Zum einen bieten Track-Lanes die Möglichkeit, mehrere Takes in der selben Spur aufzunehmen und dennoch untereinander anzuzeigen, was das Anlegen mehrerer Spuren

mit identischen Einstellungen erspart und zudem die zeitlichen Unterschiede zwischen verschiedenen Takes gut erkennen lässt.

Zum anderen konnten die vorhandenen MIDI-Features gut genutzt werden. Nuendo und Cubase sind nicht nur in der Lage, MIDI-Start/Stop-Messages zu senden, die als Ereignisse in den Test-Tools verwendet werden konnten (siehe Abschnitt 3.1.3), sie beinhalten auch den sogenannten Logical Editor. Dieser erlaubt es, MIDI-Daten zu filtern und zu verändern, was die Arbeit beim Erstellen der Testbeispiele für die empirische Bestimmung des Toleranzbereichs für den Umschaltzeitpunkt deutlich erleichterte (siehe Abschnitt 4.2).

3.1.3 Entwickelte Programmteile

Außerdem war es notwendig, einige Programmteile in Form von Max/MSP-Patches zu entwickeln, um die Kommunikation zwischen den einzelnen Komponenten des Laborsystems herzustellen.

Abbildung 3.2 zeigt die Oberfläche des Laborsystems, mit der alle im Rahmen dieser Arbeit benötigten Funktionen von einem zentralen Ort aus gesteuert werden können. Dabei sind die einzelnen Teile der Oberfläche ihrer Funktion nach farbcodiert.

Antescofo-Peripherie

Die Module mit blauer Hintergrundfarbe betreffen all jene Dinge, die mit dem Funktionsumfang von Antescofo an sich zu tun haben.

Das *Input*-Modul bietet vor allem die Möglichkeit, die gewünschte Signalquelle auszuwählen und an das Antescofo-External weiterzuleiten. Als Quellen können entweder die Eingänge eines Audio-Interfaces, eine Audiodatei oder MIDI-Daten verwendet werden. Für die beiden Audiovarianten stehen zusätzlich eine Lautstärkeregelung und ein Monitorschalter zum Mithören des Signals zur Verfügung, sowie ein Noisegate, das zur Rauschunterdrückung in Passagen ohne Nutzsignal eingesetzt werden kann. Außerdem kann per Doppelklick auf das Feld *calibration* ein Fenster zur Kalibrierung des Score-Followers geöffnet werden (für Informationen zur Durchführung siehe Abschnitt 2.4.2).

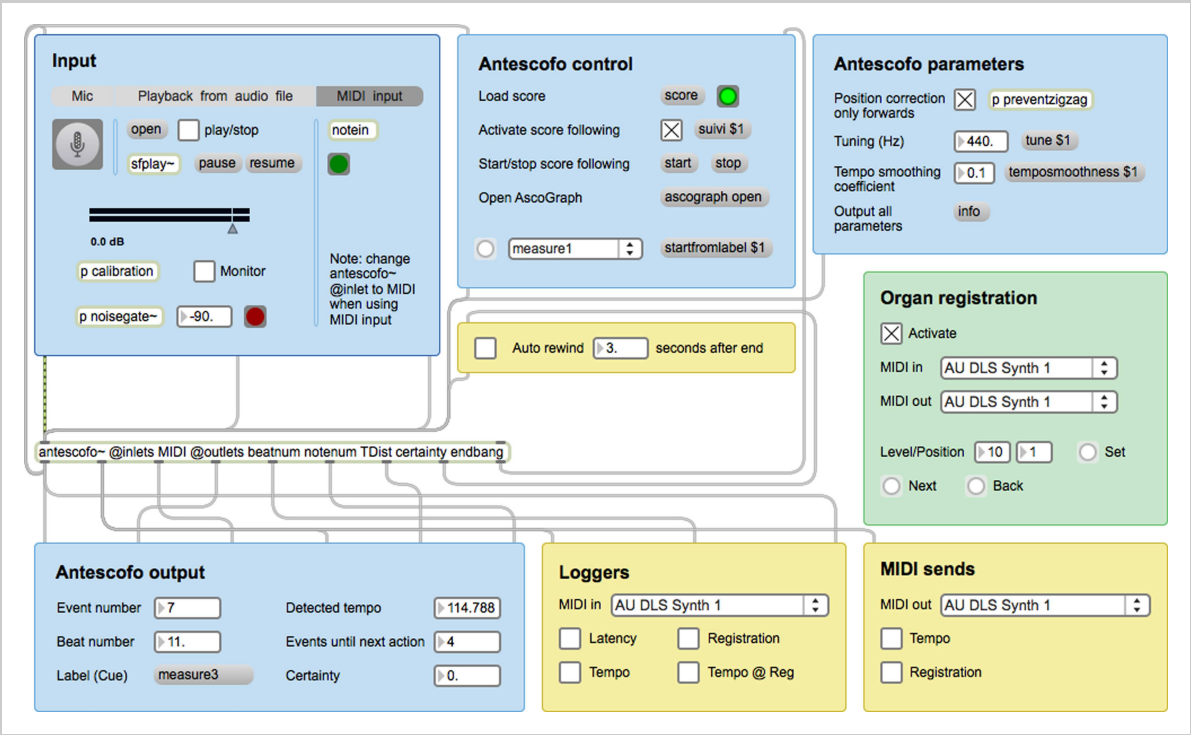


Abbildung 3.2: Oberfläche des Laborsystems

Der Block *Antescofo control* macht die wichtigsten Möglichkeiten zur Steuerung von Antescofo, wie etwa das Laden einer Partitur oder das Starten der Verfolgung, leicht zugänglich.

Mit dem Bereich *Antescofo parameters* können die für die Verwendung relevantesten Einstellungen, wie beispielsweise die Stimmung, vorgenommen werden.

Am linken unteren Rand der Oberfläche werden im Modul *Antescofo output* die für die Beobachtung der Systemfunktion sowie für die Auswertungen der Tests interessanten Werte angezeigt, die bei Betrieb des Score-Followers ausgegeben werden.

Test-Tools

Blöcke mit gelber Hintergrundfarbe stellen hingegen unterstützende Funktionen für die Testphase bereit.

Zum einen wurde ein *Loggers*-Modul erstellt, um das Sammeln von Daten für die verschiedenen Tests zu vereinfachen und die Log-Einträge mit Zeitstempeln zu versehen. Zusätzlich kann über dieses Modul eingestellt werden, welche Log-Einträge tatsächlich im Max/MSP-Programmfenster ausgegeben werden sollen, und mit einem Dropdown-Feld ist der zu verwendende MIDI-Input explizit wählbar. Außerdem werden, wenn die DAW aufgezeichnetes Material als virtuelle Quelle wiedergibt (siehe auch Abschnitt 3.2.2), von ihr ausgehende MIDI-Start/Stop-Messages dazu verwendet, den Zeitstempelgenerator mit dem Zeitpunkt der MIDI-Start-Message auf Null zu setzen sowie die Log-Ausgabe nur während der Durchführung der Versuche freizuschalten, um unerwünschte Log-Einträge zu vermeiden.

Mit dem Bereich *MIDI sends* wurde eine Möglichkeit geschaffen, bestimmte Ereignisse wie zum Beispiel Registrierungsumschaltungen neben der Ausgabe im Log auch an die DAW senden zu können. So können diese Werte dort wiederum aufgezeichnet und damit auch optisch untereinander verglichen werden. Analog zum *Loggers*-Block kann auch hier der gewünschte MIDI-Output ausgewählt werden.

Beim gelben Block unter dem Modul *Antescofo control* handelt es sich um einen zusätzliche Steuerungsmechanismus, um den Score-Follower nach dem Erreichen des Partiturendes automatisch wieder auf den Partituranfang zurückzusetzen. Dadurch können mehrere Durchläufe des selben Tests auch ohne Benutzerinteraktion durchgeführt werden, was vor allem bei langen Tests von erheblichem Vorteil sein kann.

Orgelregistrierung

Das Modul mit grünem Hintergrund ist schlussendlich für die Kommunikation mit der Orgel selbst zuständig. Sobald es aktiviert und die korrekten MIDI-Inputs und -Outputs eingestellt sind, zeigt das Modul zum einen die zur Zeit aktivierte Memory-Position inklusive Memory-Level an und ermöglicht zum anderen das Vor- und Zurückschalten sowie das direkte Setzen dieser Werte vom Laborcomputer aus.

Unter der Oberfläche spielt dieses Modul jedoch noch eine zweite, viel zentralere Rolle: Da es für die Weitergabe von Steuersignalen an die Orgel verantwortlich ist, definiert es

durch seine Implementierung de facto das zusätzliche, im Rahmen dieser Arbeit notwendige Partiturkommando für Registrierungswechsel. Dies liegt daran, dass gewünschte Registrierungsumschaltungen als Actions (siehe Abschnitt 2.3.2) in die Antescofo-Partitur eingetragen werden müssen, und die *Message*-Action (siehe Tabelle 2.3) als einfachstes aber ausreichendes Mittel zur Übertragung der angegebenen Registrierungsparameter an Max/MSP gewählt wurde. Das Kommando einer Message lautet nun genau auf den Namen des gewünschten Ziels, in diesem Fall des Receiver-Objekts `orgReg` in Max/MSP, weshalb die Wahl von dessen Bezeichnung gleichzeitig auch den Namen des Umschaltbefehls in der Partitur festlegt.

Das Modul ist nun für die Verarbeitung der Befehlsparameter zuständig, generiert die entsprechenden Steuersignale und sendet diese per MIDI an die Orgel. Damit ermöglicht es die in Tabelle 3.1 zusammengefassten Funktionen.

Syntax	Wertebereiche		Umschaltung auf
	Wert 1	Wert 2	
`orgReg next`	—	—	nächste Registrierungsposition
`orgReg prev`	—	—	vorherige Registrierungsposition
`orgReg back`	—	—	vorherige Registrierungsposition
`orgReg memory` *`lvl pos`*	1 - 99	1-6	Level *`lvl`* und Position *`pos`*
`orgReg none`	—	—	keine Registrierungsposition

Tabelle 3.1: Partiturbefehle zur Registrierungsumschaltung

3.2 Verwendungsszenarien

Durch den modularen Aufbau des Laborsystems stellt es in seiner Gesamtheit alle nötigen Funktionen für die drei im Rahmen dieser Arbeit auftretenden Verwendungsszenarien bereit. Werden bestimmte Teile für einzelne Szenarien nicht benötigt, können diese einfach unbenutzt bleiben, ohne das restliche System zu beeinflussen.

3.2.1 Aufnahme

Da für alle durchgeführten Versuche prinzipiell Input von Seiten der Orgel notwendig war, wurde vor Beginn der Versuchsphase eine Aufnahme-Session mit dem Organisten Stjepan Molnar durchgeführt, in der drei Teststücke (siehe Abschnitt A.1) in Form von Audio- und MIDI-Daten aufgezeichnet wurden.

Dieser Fall stellt in Bezug auf die Anzahl der verwendeten Laborsystemmodule das kleinste verwendete Szenario dar, denn es waren dafür nur die Hardware-Komponenten sowie die DAW im Einsatz.

3.2.2 Versuche

Bei der Durchführung der verschiedenen Versuche im Rahmen des Entwicklungsprozesses (siehe Kapitel 4) waren zwei Szenarien zu unterscheiden.

Für Versuche, die ausschließlich mit virtuellem Input durchgeführt werden konnten (wie beispielsweise die Bestimmung von Software-Latenzen), war der Einsatz der Hardware-Komponenten Orgel, Interface und Wiedergabesystem nicht nötig. Dies erleichterte die Arbeit insofern erheblich, als dass diese Versuche unabhängig von der Orgel und damit von den zugehörigen Räumlichkeiten stattfinden konnten.

Möglich war dieses Szenario durch die unter MacOS bereits standardmäßig vorhandene Möglichkeit, virtuelle MIDI-Interfaces zu erstellen, mit denen MIDI-Signale innerhalb des Laborcomputers von einem Programm zum anderen geschickt werden können (unter MacOS mittels des sogenannten IAC-Treibers, in anderen Betriebssystemen wäre Ähnliches durch Zusatzprogramme möglich). Die DAW agierte also in diesen Fällen als Quelle, und nahm damit sozusagen den Platz eines virtuellen Organisten ein.

Für gewisse Tests, wie etwa die empirische Bestimmung des Toleranzbereichs für den Umschaltzeitpunkt, war jedoch ein zweites Versuchsszenario mit Verwendung aller Komponenten des Laborsystems (wie in Abbildung 3.1 dargestellt) gegeben.

3.2.3 Anwendung

Beim schlussendlichen realen Einsatz des Systems zur Registrierungsumschaltung während des Orgelspiels kommen auch nahezu alle Komponenten zum Einsatz. Lediglich Test-Tools und DAW sind in diesem Fall nicht mehr notwendig, und die Verwendung von AscoGraph als Anzeige der aktuellen Position im Stück ist optional.

4 Entwicklungsprozess

Im Entwicklungsprozess wurde das eben beschriebene Laborsystem herangezogen, um Entwicklungsfragen zu untersuchen und Versuche zu verschiedenen Teilen des Systems durchzuführen. Abschnitt 4.1 befasst sich dabei mit der Verwendbarkeit von Audio- und MIDI-Signalen, Abschnitt 4.2 untersucht den Toleranzbereich für den Zeitpunkt der Registrierungsumschaltung und in Abschnitt 4.3 werden verschiedene zeitliche Systemgrößen bestimmt. Aufgrund der Ergebnisse eines ersten Testlaufs (Abschnitt 4.4), werden in Abschnitt 4.5 Einflüsse auf die Genauigkeit der Tempodetektion sowie deren Auswirkungen diskutiert, bevor in Abschnitt 4.6 der abschließende Testlauf und die ermittelten Resultate beschrieben werden.

4.1 Audio- vs. MIDI-Input

Ein erster informeller Test sollte die Frage klären, welche der sowohl an der Orgel als auch bei Antescofo verfügbaren Schnittstellen, nämlich Audio und MIDI, für den angestrebten Zweck verwendbar wären. Dazu wurde die Reaktionszeit und der Detektionserfolg von Antescofo beim Verfolgen des Präludiums in C-Dur von J. S. Bach beobachtet.

Die ersten Tests erfolgten dabei mit einem Klavier und einem MIDI-Keyboard. Diese ergaben, dass der Score-Follower nicht nur bei MIDI-Input erwartungsgemäß schnell und präzise arbeitete, auch bei Audio-Input (mit entsprechender Signalqualität und ausreichendem Signal-Rausch-Abstand) konnten gute Ergebnisse in Bezug auf Reaktionszeit und Detektionserfolg (kaum Verpassen einzelner Events) erzielt werden. Ein anfangs zu geringer Signal-Rausch-Abstand bei diesen Tests gab auch den Ausschlag, das System mit einem Noisegate auszustatten.

In der hoffnungsvollen Annahme, dass diese Ergebnisse auch auf den Fall der Orgel als Input-Quelle zutreffen würden, wurde der selbe Test an der Rodgers-Orgel am Institut für Kirchenmusik und Orgel der Universität für Musik und darstellende Kunst Graz wiederholt. Wie zu erwarten, waren die MIDI-Tests ebenfalls erfolgreich, bei Testung von Audio als Input offenbarte sich jedoch der entscheidende Unterschied zwischen den beiden Instrumenten.

Denn obwohl Tests mit nur einem, in der dem Notentext entsprechenden Oktavlage aktivierten Register ähnliche Ergebnisse zu jenen der Klaviertests lieferten, fielen Tests bei Zuschaltung von immer mehr Registern zunehmend schlechter aus. Waren bei einer zusätzlichen Oktavlage auch nur kleine Verlängerungen der Detektionszeiten festzustellen, so hatte eine Zuschaltung von mehreren Oktaven und einzelnen zusätzlichen Intervallverhältnissen wie Terz- und Quintregister schlussendlich zur Folge, dass die Event-Detektion durch Antescofo quasi nicht mehr möglich war.

Bei näherer Betrachtung ist diese Tatsache jedoch nicht unlogisch, wenn man bedenkt, dass Antescofo ausschließlich den Notentext mit dem eingehenden Signal abgleicht, und bei Übereinstimmung das jeweilige Event detektiert. Kommen nun aber zusätzliche Oktavlagen bzw. Terzen und Quinten dazu, werden diese (sofern sie nicht leise genug sind, um als Teiltöne eines notierten Tons angesehen zu werden) als eigenständige Töne detektiert. Damit passt jedoch der eingehende Klang im engsten Sinn nicht mehr zum notierten Notentext, und ein Zusammenhang ist für den Score-Follower nicht mehr feststellbar.

Da die Verwendung von Registerkombinationen jedoch einen fundamentalen Bestandteil des Instrumentes Orgel darstellt, musste demnach die Verwendung von Audio-Input ausgeschlossen werden, und MIDI verblieb damit als zumindest derzeit einzige Input-Möglichkeit. Bei einem Gespräch mit Arshia Cont, dem Leiter des Antescofo-Projektes, wurde dieser auf den Sachverhalt aufmerksam gemacht. Er stellte daraufhin in Aussicht, dass in Zukunft eventuell eine Möglichkeit geschaffen werden könnte, um dem Score-Follower zusätzliche Klangbestandteile mittels Partiturkommandos mitteilen zu können. Sobald dies geschieht, könnte dann auch Audio wieder als Input in Betracht gezogen werden.

4.2 Toleranzbereich für den Umschaltzeitpunkt

Der Zeitpunkt für die Auslösung eines Registrierungswechsels ist für eine zufriedenstellende Funktion des Systems von enormer Wichtigkeit. Zum einen hängt dieser Zeitpunkt vom Reaktionsverhalten der Orgel ab, zum anderen war anzunehmen, dass es in Zusammenhang mit dem Gehör eine gewisse Toleranz bezüglich des Schaltzeitpunktes gibt. Deshalb wurde einerseits ein informeller Hörversuch entworfen, um den Toleranzbereich für den Auslösezeitpunkt empirisch festzustellen, und andererseits die Reaktionszeit der Orgel für eine theoretische Abschätzung bestimmt (siehe Abschnitt 4.3.1).

Für den Hörversuch wurden zwei verschiedene MIDI-Testsequenzen erstellt, die jeweils an einem Akkordübergang einen Registrierungswechsel enthalten. Für Testsequenz A (siehe Abbildung A.1) wurde die MIDI-Aufnahme einer realen Beispielstelle aus dem aufgenommenen Testmaterial verwendet, in der ein Ton liegen bleibt, während der einspielende Organist einen Registerwechsel vorgenommen hat. Als Testsequenz B (siehe Abbildung A.2) wurde eine MIDI-Sequenz neu erstellt, die das vollständige Legato-Spiel von zwei aufeinander folgenden Akkorden simuliert.

Außerdem wurden zwei Registrierungen festgelegt, die mit den entsprechenden MIDI-Kommandos aktiviert werden konnten: eine eher weich klingende unter ausschließlicher Verwendung von Lippenpfeifen (Registrierung A[1]), und eine darauf aufbauende, jedoch mit hinzugefügten Zungenpfeifen und höheren Oktavlagen eher schärfer klingende (Registrierung B[2]).

Aus diesen Bausteinen wurden anschließend mit Hilfe des in der DAW enthaltenen Logical Editors für jede der beiden Testsequenzen jeweils zwei Testreihen generiert. In der einen sollten die Töne vor dem Wechsel mit Registrierung A klingen, und jene danach mit Registrierung B, für die jeweils andere Testreihe wurden die beiden Registrierungen vertauscht. Als Testfälle innerhalb der Testreihen wurde nun das Kommando für den Registrierungswechsel an verschiedenen Stellen jeweils im Abstand von 5 ms im Bereich von 0 bis 200 ms vor dem Beginn des dem Wechsel folgenden Akkordes positioniert.

[1] Grande Orgue: Flûte harmonique 8, Bourdon 8, Gambe 8; Pédale: Soubasse 16, Flûte 8, Bourdon 8, Violon celle 8

[2] Grande Orgue zusätzlich: Flûte 4, Salicional 4, Trompette 8, Clairon 4; Pédale zusätzlich: Trompette 8, Clairon 4

Bei Durchführung des Tests durch den Autor selbst wurde dann die subjektive Qualität der Klangumschaltung in den einzelnen Testfällen bewertet. Dafür wurden die Testfälle nach der Hörbarkeit von Artefakten aufgrund der Umschaltung eingestuft: keine Artefakte, leichte Artefakte (z.B. als ungenauer Anschlag des Akkordes interpretierbar) oder störende Artefakte.

Anhand der grafischen Darstellung in Abbildung 4.1 ist gut erkennbar, dass Bereiche rund um -65 ms (also 65 ms vor dem Akkordübergang und damit dem optimalen, gewünschten Wechselzeitpunkt) in allen vier Testreihen (obere vier Balken) gut bewertet wurden. Darauf aufbauend wurde jener Bereich ermittelt, in dem in allen vier Testreihen keine oder nur leichte Artefakte auftraten. Dieser ermittelte Toleranzbereich (unterster Balken) stellt jenen Bereich mit einer Breite von ca. 25 bis 30 ms dar, in dem laut den durchgeführten Tests das Steuerungskommando ausgelöst werden sollte, damit bei der tatsächlichen Klangumschaltung in möglichst vielen Fällen keine störenden Artefakte auftreten.

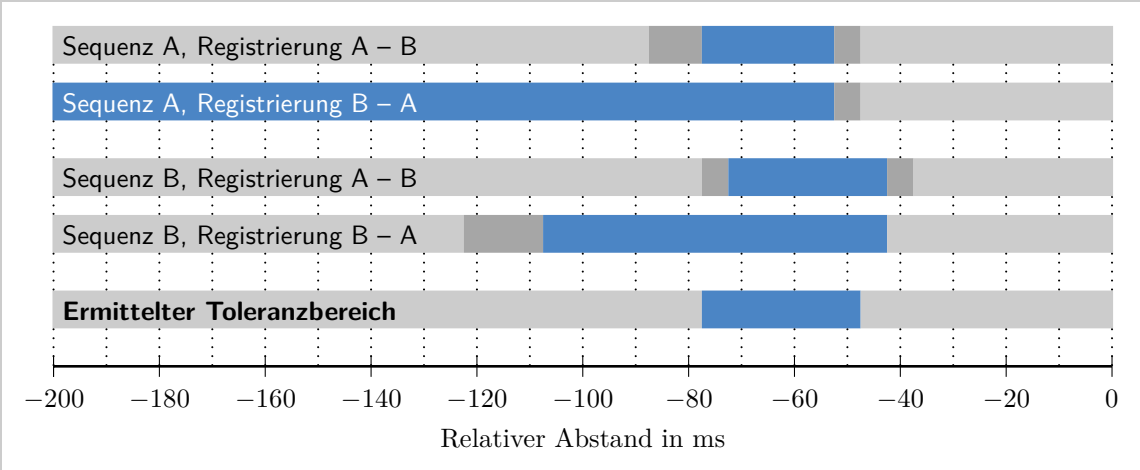


Abbildung 4.1: Testergebnisse der Toleranzbereichsbestimmung für den relativen Abstand des Registrierungskommandos zum gewünschten Wechselzeitpunkt, bewertet mit »keine Artefakte« (blau), »leichte Artefakte« (dunkelgrau) und »störende Artefakte« (hellgrau)

Es ist jedoch zu beachten, dass dieser informelle Hörversuch mit einer begrenzten Anzahl von Testfällen bzgl. verwendeter Registrierungen und musikalischer Materialien durchgeführt wurde, und der ermittelte Toleranzbereich deshalb nur eingeschränkt gültig ist. Dies wird auch durch die Versuchsergebnisse unterstrichen, die zeigen, dass der optimale

Bereich für den Kommandozeitpunkt stark von der verwendeten Registrierungsfolge sowie vom Notentext an der betreffenden Stelle abhängig ist. Vergleicht man die Ergebnisse der Testreihen 1 und 2 bzw. 3 und 4, ist zu sehen, dass die als gut bewerteten Regionen im Fall von Registrierungsfolge B – A deutlich größer sind. Dass in diesen Fällen ein weicherer, leiserer Klang auf einen schärferen, lauteren folgt, legt die Vermutung nahe, dass der größere Toleranzbereich auf einen Maskierungseffekt zurückzuführen ist.

Der auffallend große Toleranzbereich in der zweiten Testreihe (Sequenz A, Registrierungsfolge B – A) kommt jedoch zusätzlich aufgrund des Notentextes bzw. der Interpretation durch den Organisten bei der Aufnahme zustande. Er vergrößerte nämlich den Zeitraum des gehaltenen Tones zwischen den Akkorden ganz leicht. Deshalb fanden alle frühen, getesteten Schaltzeitpunkte während dieses gehaltenen Tons statt, und eine zu frühe Umschaltung wurde aufgrund der Registrierungsfolge B – A und des damit verbundenen Maskierungseffektes einfach als Absetzen des Tons wahrgenommen. Dies entsprach zwar nicht mehr exakt dem Notentext, war jedoch als plausibler Klang interpretierbar und damit nicht als störend empfunden worden.

4.3 Zeitliche Systemgrößen

Um einen Überblick über das Systemverhalten zu bekommen, wurden außerdem die zeitlichen Größen des Systems bestimmt.

4.3.1 Reaktionszeit der Orgel

Als erste Systemgröße wurde die Reaktionszeit der Orgel auf vom Laborcomputer aus gesendete MIDI-Signale untersucht. Zu diesem Zweck wurde in einem ersten Versuch eine MIDI-Testsequenz mit Hilfe der DAW generiert, welche zuerst das MIDI-Kommando zur Registrierungsdeaktivierung sendet und das Niederdrücken der Tasten eines Akkords mittels zugehöriger MIDI-Befehle simuliert. Anschließend wird, während die Tasten noch virtuell gedrückt sind, eine Registerkombination per Kommando vom Laborcomputer aus aktiviert.

Wird diese Sequenz an die Orgel gesendet, so ist zu Beginn des Akkordes noch kein Klang zu hören, weil keine Registrierung aktiv ist. Das zweite Registrierungskommando aktiviert jedoch eine Registrierung, wodurch die Klangerzeugung der Orgel einsetzt und damit ein Audiosignal ausgegeben wird.

Zeichnet man nun dieses Audiosignal auf, kann die Zeitdifferenz zwischen dem Senden des zweiten Registrierungskommandos und dem Beginn des resultierenden Klanges aus dem Abstand der entsprechenden Sample-Positionen bestimmt werden.

Da es sich beim Registrierungskommando um einen (dreiteiligen) MIDI-Befehl handelt, sind dessen Sample-Positionen exakt definiert und direkt in der Timeline der DAW ablesbar. Für die Berechnung wurde die Position des dritten Teilbefehls herangezogen, da die Orgel erst auf diesen reagiert. Um den Beginn des ausgelösten Klanges zu bestimmen, wurde ein MATLAB- bzw. Octave-Script[1] programmiert, das jene Sample-Position in den aufgezeichneten Audiodateien sucht, bei welcher der berechnete Spannungspegel erstmals einen Wert von -67 dBFS überschreitet. Dieser gerundete Wert wurde manuell ermittelt und entspricht jenem Pegel, der in allen verwendeten Audioaufnahmen knapp über dem Grundrauschen liegt. Außerdem gibt das Script den Sample-Abstand zwischen MIDI-Befehl und Klangbeginn sowie die daraus berechnete Reaktionszeit der Orgel aus (eine Berücksichtigung der Aufnahmelatenz des Audio-Interfaces ist dabei aufgrund des automatischen Latenzausgleichs der DAW nicht notwendig [Ste15, S. 27]).

Tabelle 4.1 zeigt eine Zusammenfassung der in fünf Wiederholungen ermittelten Werte. Daraus lässt sich ablesen, dass die Reaktionszeit der Orgel auf eine Registrierungsumschaltung etwa 78 ms beträgt.

Reaktionszeit in ms					Mittelwert in ms	Std.abw. in ms
1	2	3	4	5		
92	67	82	70	78	77.8	10.0

Tabelle 4.1: Messergebnisse der Orgelreaktionszeit auf Registrierungsbefehle (fünf Wiederholungen, Werte gerundet)

[1] MATLAB bzw. die Open-Source-Alternative Octave sind Programmiersprachen und -umgebungen, die speziell auf die effiziente Durchführung komplexer mathematischer Berechnungen ausgelegt sind.

In einem zweiten Versuch wurde ebenfalls ein liegender Akkord an die Orgel gesendet, diesmal jedoch bei bereits aktivierter Registrierung, um die Reaktionszeit auf Tastenanschlagsbefehle zu bestimmen (Durchführung analog zum zuvor beschriebenen ersten Versuch). Wie in Tabelle 4.2 ersichtlich ist, fällt die Reaktionszeit in diesem Fall mit etwa 10 ms im Vergleich recht niedrig aus. Dieser Wert legt auch nahe, dass sich die Reaktionszeiten auf per MIDI gesendete sowie manuell an der Orgel ausgeführte Tastenanschläge in einer ähnlichen Größenordnung befinden.

Reaktionszeit in ms					Mittelwert in ms	Std.abw. in ms
1	2	3	4	5		
10	9	10	10	9	9.6	0.5

Tabelle 4.2: Messergebnisse der Orgelreaktionszeit auf Tastenanschlagsbefehle (fünf Wiederholungen, Werte gerundet)

Die Orgel reagiert also recht schnell auf Tastenanschlagsbefehle, die Verarbeitung von Registrierungsbefehlen dauert dagegen etwas länger. Betrachtet man die beiden Reaktionszeiten in Relation zueinander, so lässt sich daraus auch jene Zeit errechnen, die ein Registrierungskommando theoretisch vor dem gewünschten Zeitpunkt der Klangveränderung, nämlich dem Erklingen der ersten Note im neuen Registrierungsabschnitt, gesendet werden müsste. In Bezug auf die beiden eben beschriebenen Versuche ist dies gleichbedeutend mit dem Fall, dass die Effekte der beiden MIDI-Befehle, also die Klangauslösung und die Registrierungsumschaltung, gleichzeitig eintreten sollen. Der berechnete Abstand des Registrierungskommandos vom Klangbeginn ergibt sich folglich aus der Differenz der beiden Reaktionszeiten und liegt bei etwa 68 ms (vom dritten Teilbefehl aus gerechnet). Für den Vergleich mit dem in Abschnitt 4.2 empirisch ermittelten Wert, wo jedoch der erste Teilbefehl für die Positionierung der Registrierungskommandos verwendet wurde, muss noch der Abstand zwischen dem ersten und dritten Teilbefehl (ca. 2 ms) addiert werden. Der Vergleichswert beträgt somit etwa 70 ms, liegt aber dennoch nahe am empirischen Ergebnis von ca. 65 ms.

4.3.2 Detektionslatenz von Antescofo

Die zweite untersuchte zeitliche Systemgröße ist die Detektionslatenz von Antescofo, also jene Zeit, die zwischen dem Eingehen eines Klanges und seiner Erkennung bzw. Zuordnung zu einem Partitur-Event durch den Score-Follower verstreicht.

Dazu wurde Testsequenz C (siehe Abbildung A.3) verwendet, in der die ersten fünf Töne einer C-Dur-Tonleiter einmal auf- und absteigend gespielt werden. Jedem Ton bzw. jedem entsprechenden Event in der Antescofo-Partitur wurde eine Action zugeordnet, um bei Erkennung des Events ausgelöst zu werden und in Max/MSP einen entsprechenden Log-Eintrag zu erzeugen. Im Testablauf wurden dann sowohl die Eingangszeitpunkte der durch die DAW als virtuellen Input wiedergegebenen Töne als auch die Zeitstempel der ausgelösten Actions im Latenz-Log festgehalten, aus deren Differenzen sich die Latenzen in Tabelle 4.3 ergaben.

Event-Nr.	Latenz in ms					Mittelwert in ms	Std.abw. in ms
	1	2	3	4	5		
1	10	7	1	9	5	6.4	3.6
2	21	18	12	19	16	17.2	3.4
3	20	17	23	19	16	19.0	2.7
4	77	75	80	76	73	76.2	2.6
5	18	16	21	17	14	17.2	2.6
6	18	15	20	16	13	16.4	2.7
7	63	61	66	62	59	62.2	2.6
8	16	13	19	15	12	15.0	2.7
9	15	13	18	14	11	14.2	2.6
Global						27.1	23.4

Tabelle 4.3: Messergebnisse der Detektionslatenz (fünf Wiederholungen, Werte gerundet) in der Antescofo-Standardeinstellung (Varianzwert 0.3)

An den Daten ist einerseits zu erkennen, dass die Latenz bei Detektion des ersten Events durchgehend am geringsten war. Es liegt nahe, dass dies der Fall ist, da der Algorithmus mit erhöhter Wahrscheinlichkeit davon ausgehen kann, dass die Sequenz beim ersten Event beginnen wird. Analoges gilt auch für das letzte Event der Sequenz.

Andererseits fällt bei der Auswertung der Daten auf, dass die Latenzen bei der Detektion von Events 4 und 7 deutlich höher sind, als bei den anderen. Bei näherer Betrachtung

stellt sich heraus, dass diese Events jeweils dem zweiten Ton eines Halbtonschrittes entsprechen, was die Vermutung nahe legt, dass dieses Verhalten etwas mit der Nähe der beiden aufeinander folgenden Töne zu tun haben könnte.

Deshalb entstand die Idee, dass dieses Verhalten eventuell über den Varianzparameter (siehe Abschnitt 2.4.2) beeinflusst werden könnte. Dementsprechend wurde der Versuch unter Variierung des Varianzparameters wiederholt. Die Ergebnisse sind in Abbildung 4.2 grafisch dargestellt.

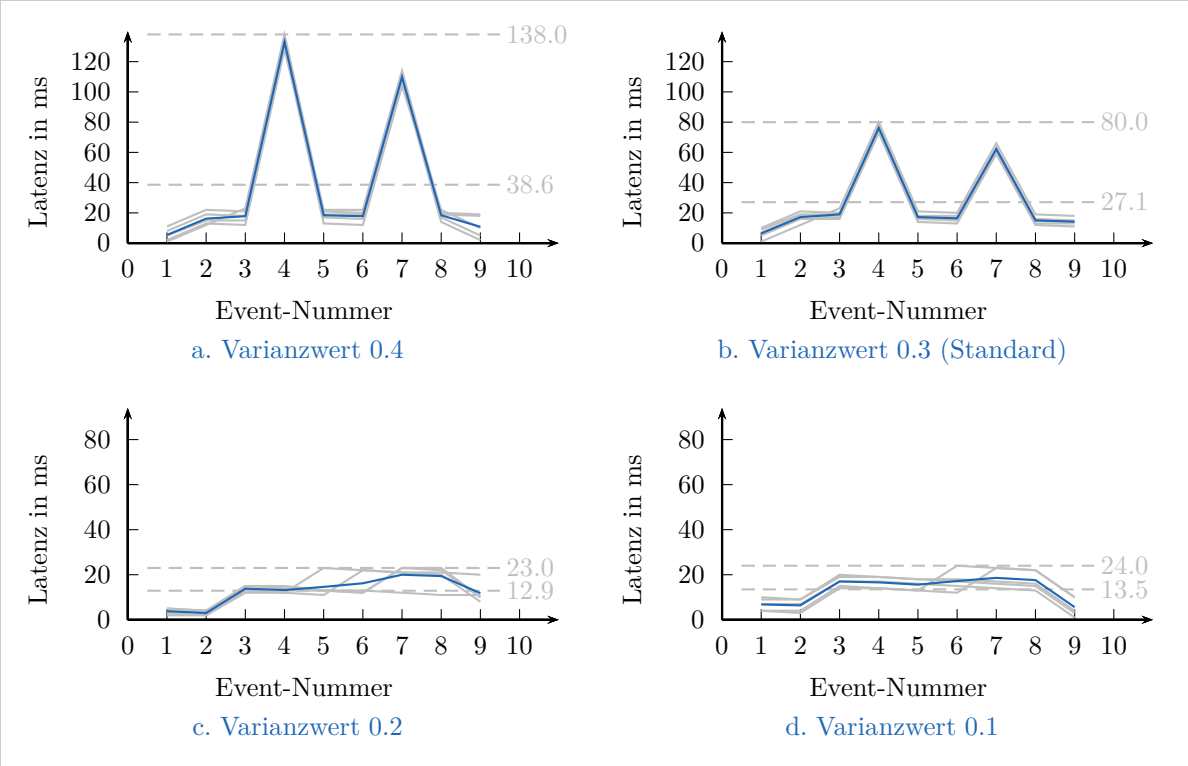


Abbildung 4.2: Verläufe (grau) und Mittelwerte (blau) der gemessenen Detektionslatenzen für jedes Event der Testsequenz C (fünf Wiederholungen), sowie deren globale Mittelwerte und Maxima (grau strichliert)

Wie die Versuchsergebnisse zeigen, hat der Varianz-Parameter einen deutlichen Einfluss auf die Detektionslatenz, welche durch die Wahl kleinerer Varianzwerte vor allem im Fall von Halbtonschritten stark verringert werden kann.

Zu erklären ist dieser Effekt mit der Tatsache, dass mit kleinerem Varianzwert auch die Breite der Normalverteilungen abnimmt, die zur Modellierung der Frequenz-Templates in Antescofo eingesetzt werden (vgl. Abschnitt 2.2.2). Durch die geringere Breite überlappen sich nun die Verteilungen benachbarter Töne weniger, was dem Algorithmus die Entscheidung zwischen den beiden Tönen erleichtert. Da die Standardeinstellung mit einem Varianzwert von 0.3 ca. der Breite eines Dritteltons entspricht, ist diese im Fall von Ganztonschritten zwar ausreichend klein, bei Halbtonschritten jedoch so groß, dass deshalb höhere Detektionslatenzen entstehen.

Andererseits sinkt jedoch mit geringerer Breite auch die zulässige Abweichung der gespielten Tonhöhe von der notierten, was bei der Anwendung mit Audiosignalen zu Problemen führen könnte. Denn stellen akustische Instrumente in realen Räumen die Klangquelle dar, ist die Tonhöhe Schwankungen durch Einflüsse wie Raumtemperatur oder intonatorisches Können der Musikerin bzw. des Musikers unterworfen. Dies muss jedoch nur berücksichtigt werden, falls dieses System zur Registrierungsumschaltung in Zukunft auf Audio-Input umgestellt werden sollte. Solange MIDI verwendet wird, treten solche Schwankungen nicht auf, und die Varianz kann zur Verringerung der Detektionslatenz ohne Weiteres niedriger als der Standardwert gewählt werden.

4.3.3 Jitter bei interner MIDI-Kommunikation

Um sicherzustellen, dass die gemessenen Werte jener Versuche, die mit virtuellem Input unter Einsatz der DAW durchgeführt wurden, nicht durch Jitter bei der internen Übertragung verfälscht werden, wurden auch die Ankunftszeiten der MIDI-Signale in Max/MSP und ihre Genauigkeit untersucht.

Dazu wurde Testsequenz D (siehe Abbildung A.4), ein langsamer Satz einer Orgelsonate, verwendet, in der die bei der Aufnahme vom Organisten selbst durchgeführten Registrierungsumschaltungen enthalten waren. Diese Sequenz wurde über die DAW wiedergegeben (gestartet immer an der selben Stelle), und die Eingangszeitstempel von fünf Wiederholungen der Registrierungskommandos in Max/MSP direkt vor dem Antescofo-Eingang miteinander verglichen. Die ermittelten Werte sind in Tabelle 4.4 dargestellt

(Registrierung 10/1 ist nicht enthalten, da diese schon vor Beginn des Stücks aktiviert und damit nicht wiedergeben wurde).

Registrierung	Zeitstempel in ms					Spannweite in ms
	1	2	3	4	5	
10/2	10207	10207	10208	10207	10207	1
10/3	20166	20166	20167	20165	20166	2
10/4	29665	29666	29666	29665	29665	1
10/5	39809	39809	39809	39809	39808	1

Tabelle 4.4: Messergebnisse des Jitters bei interner MIDI-Kommunikation (fünf Wiederholungen, Werte gerundet)

Wie die berechneten Spannweiten[1] zeigen, ergibt sich aus den Messdaten eine Schwankungsbreite von maximal 2 ms. Dies entspricht einem Jitter von etwa ±1 ms, der jedoch im Vergleich mit Ergebnissen der anderen Versuche zu zeitlichen Faktoren des Systems vernachlässigt werden kann.

4.4 Erster Testlauf

Nachdem nun die Randbedingungen des Systems bestimmt waren, konnte ein erster Testlauf des bis zu diesem Zeitpunkt bestehenden Systems vorgenommen werden. Als Teststück wurde dafür erneut Testsequenz D (siehe Abbildung A.4) herangezogen.

In einem ersten Schritt musste für diese Sequenz eine Antescofo-Partitur erstellt werden. Dazu wurden Kopien der Notenblätter, die der Organist bei der Aufnahme verwendet hatte, per Einlesefunktion eines Notensatzprogramms (in diesem Fall Finale) in dieses übertragen, die Einlesefehler korrigiert und als MusicXML-Datei abgespeichert. Diese konnte wiederum mit Hilfe von AscoGraph in das Antescofo-Format konvertiert werden.

Anschließend sollten die Registrierungswechsel in Form der zu aktivierenden Speicherstellen des Orgelsetzers in die Partitur eingetragen werden, wofür die im Orgelkommunikationsmodul des Laborsystems (siehe Abschnitt 3.1.3) festgelegten Kommandos zur

[1] Die Spannweite bezeichnet die Differenz zwischen Maximum und Minimum einer Gruppe von Daten.

Verfügung standen. Aufgrund der Funktionsweise von Antescofo-Actions mussten diese jedoch an die jeweiligen Events vor den gewünschten Zeitpunkten gebunden und deren Auslösungen mit entsprechenden Delay-Angaben verzögert werden.

Deshalb waren zuerst die dafür notwendigen Verzögerungszeiten festzustellen. Als erste Anhaltspunkte wurden die Zeitabstände zwischen den vom Organisten während der Aufnahme manuell ausgelösten Registrierungswechseln (erster MIDI-Teilbefehl) und den Beginnzeiten der jeweils vorangehenden Akkorde (erster MIDI-Notenbefehl) in der MIDI-Ansicht der DAW gemessen. Diese absoluten Zeitwerte wurden anschließend unter Verwendung der in einem vorherigen Durchlauf des Stücks ermittelten Temposchätzungen für die entsprechenden Stellen in relative Werte der Einheit Schläge (Viertelnoten) umgerechnet, um im Hinblick auf die leichte menschliche Ungenauigkeit von leicht unterschiedlichen Tempi unabhängig sein zu können.

Nach einer anschließenden empirischen Verfeinerung dieser Werte war also alles vorbereitet, und der erste Testlauf konnte beginnen. Dazu wurde Testsequenz D fünf Mal über die DAW wiedergegeben und die Zeitstempel der resultierenden, von Antescofo ausgelösten Registrierungswechsel mit den aufgezeichneten verglichen. Die festgehaltenen Werte sind in Tabelle 4.5 dargestellt (Registrierung 10/1 ist nicht enthalten, da diese schon vor Beginn des Stücks aktiviert und damit nicht wiedergeben wurde).

Registrierung	Zeitstempel in ms						Mittelwert in ms	Std.abw. in ms
	Ref.[1]	1	2	3	4	5		
10/2	10125	10092	10114	10091	10082	10073	10090.4	15.3
10/3	20084	20110	20070	20107	20110	20105	20100.4	17.1
10/4	29584	29636	29626	29634	29615	29602	29622.6	14.2
10/5	39727	39771	39762	39769	39753	39710	39753.0	25.0

Tabelle 4.5: Ergebnisse des ersten Testlaufs (fünf Wiederholungen, Werte gerundet)

Die berechneten Standardabweichungen zeichnen jedoch kein gutes Bild der Systemperformance: Durch die Zugrundelegung einer Normalverteilung bedeutet die maximale Standardabweichung von 25 ms, dass jener Bereich, der 99.6 % aller auftretenden Fälle abdeckt, eine Breite der sechsfachen Standardabweichung, also 150 ms, aufweist. Diese

[1] Da die Spannweite der Referenzzeitstempel pro Registrierung maximal 1 ms betrug, wird hier aus Platzgründen nur der Medianwert angeführt.

Bereichsbreite ist jedoch im Vergleich mit dem in Abschnitt 4.2 ermittelten Toleranzbereich für den Umschaltzeitpunkt von ca. 30 ms absolut inakzeptabel. Bei fünf neuerlichen Durchläufen unter zusätzlicher Betrachtung der Temposchätzungen an den betroffenen Positionen stellte sich heraus, dass diese Abweichungen direkt auf die Unterschiede der bei den einzelnen Durchläufen detektierten Tempi zurückzuführen sind.

4.5 Einflüsse auf die Tempoerkennung

Nach dem unbefriedigenden Ergebnis des ersten Testlaufs in Bezug auf die Systemperformance galt es also festzustellen, ob diese bzw. die zugrunde liegende Reproduzierbarkeit der Temposchätzungen durch geeignete Antescofo-Parameter verbessert werden kann (Abschnitt 4.5.1). Außerdem sollte in diesem Zusammenhang untersucht werden, ob die Interpretation durch Musikerinnen und Musiker (Abschnitt 4.5.2) oder das musikalische Material selbst (Abschnitt 4.5.3) Auswirkungen auf die Leistung des Systems haben.

4.5.1 Antescofo-Parameter

Grundsätzlich könnte bei zwei einstellbaren Parametern von Antescofo angenommen werden, dass diese eventuell Auswirkungen auf die Schätzung der Tempowerte haben.

Varianz

Der erste Parameter, bei dem Auswirkungen denkbar sind, ist die Varianz der zur Klangmodellierung im Frequenzbereich verwendeten Normalverteilungen, da deren Einfluss auf die Detektionslatenz bereits nachgewiesen wurde (siehe Abschnitt 4.3.2).

In einem entsprechenden Versuch wurde Testsequenz D (siehe Abbildung A.4), ein langsamer Satz einer Orgelsonate, mit zwei verschiedenen Varianzeinstellungen jeweils fünf Mal durch den Score-Follower geschickt: zuerst mit der Standard-Varianz von 0.3, und anschließend mit einem Varianzwert von 0.1, da dieser im Versuch zur Detektionslatenz (siehe Abschnitt 4.3.2) sowohl kleine Maximallatenzen als auch die kleinsten Latenzunterschiede zwischen den Events (siehe Abbildung 4.2d) zur Folge hatte.

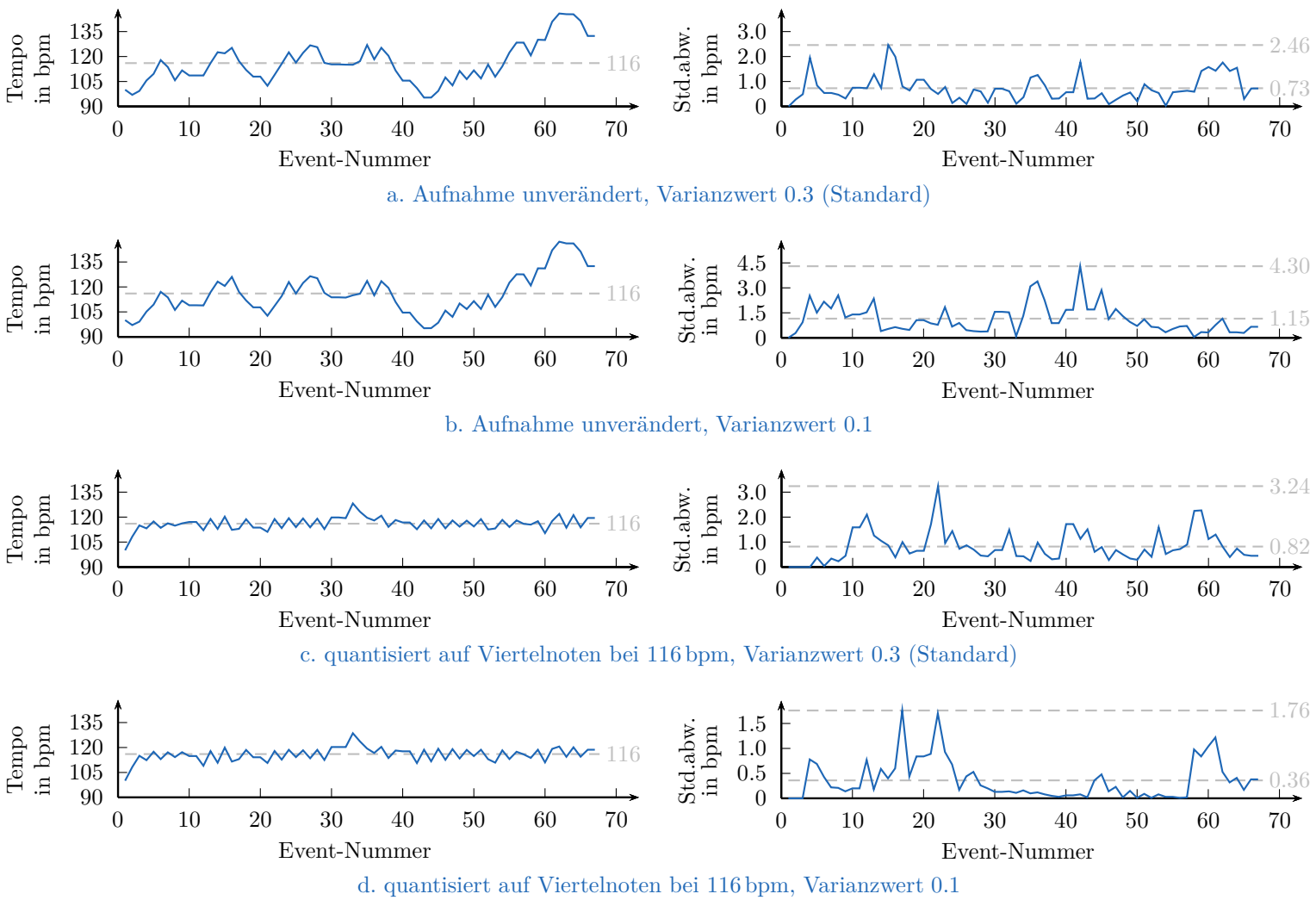
Zudem wurde der Versuch ein zweites Mal mit einer auf Viertelnoten bei 116 bpm quantisierten[1] Version der Testsequenz durchgeführt. Dieses Tempo entsprach dem gerundeten Mittelwert aller Tempowerte und führte damit zur annähernd selben Gesamtdauer von quantisierter und originaler Testsequenz.

Wie die Ergebnisse in Abbildung 4.3 zeigen, kann eine Verbesserung oder Verschlechterung der Systemperformance nicht eindeutig festgestellt werden: Ein Varianzwert von 0.1 führt beim Versuch mit der Originalvariante der Testsequenz offenbar zu höheren globalen Standardabweichungen (Abbildung 4.3b) und damit zu einer schlechteren Reproduzierbarkeit der Temposchätzungen, in Verbindung mit der quantisierten Version jedoch zu einer Verbesserung (Abbildung 4.3d).

Geht man im Sinne einer Worst-Case-Abschätzung von einer Verschlechterung aus, so müsste eine Herabsetzung der Varianz allein aufgrund dieser Daten eigentlich abgelehnt werden, was jedoch im Konflikt mit der Tatsache steht, dass kleinere Varianzwerte die Detektionslatenz deutlich verbessern (vgl. Abschnitt 4.3.2). Wägt man die Größenordnungen der Auswirkungen ab, erscheint es aber trotzdem sinnvoll, einen Varianzwert von 0.1 dem Standardwert vorzuziehen, da damit zwar die Reproduzierbarkeit der Temposchätzungen etwa um den Faktor 1.6 bis 1.7 abnimmt, die Detektionslatenz jedoch um einen etwas höheren Faktor von 2 bis 3.3 verbessert wird.

Auffallend ist außerdem das Aussehen der Tempoverläufe mit der quantisierten Testsequenz als Input, welches an eine Zick-Zack-Linie erinnert, obwohl man eher eine Gerade erwarten würde. Warum dieses Verhalten auftritt, konnte nicht restlos geklärt werden. In einem Gespräch mit Antescofo-Projektleiter Arshia Cont führte dieser den Effekt auf das in Antescofo verwendete Tempodetektionsmodell zurück, welches seiner Aussage nach auf menschliche Darbietungen mit ihren inhärenten Temposchwankungen ausgelegt ist und deshalb auch bei deren Abwesenheit um den wahren Wert oszilliert.

[1] Als Quantisierung wird in der Audiobearbeitung jener Vorgang bezeichnet, bei dem alle musikalischen Ereignisse (wie Beginnzeiten von Noten, Notendauern etc.) eines MIDI-Signals aufgrund einer Vorgabe von Tempo und kleinstem erlaubtem Notenwert soweit verändert werden, dass diese Ereignisse exakt mit den dadurch eingestellten äquidistanten Zeitpunkten bzw. ganzzahligen Vielfachen des entsprechenden Abstandes zusammenfallen.



a. Aufnahme unverändert, Varianzwert 0.3 (Standard)

b. Aufnahme unverändert, Varianzwert 0.1

c. quantisiert auf Viertelnoten bei 116 bpm, Varianzwert 0.3 (Standard)

d. quantisiert auf Viertelnoten bei 116 bpm, Varianzwert 0.1

Abbildung 4.3: Mittelwerte (links) und Standardabweichungen (rechts) sowie globale Mittelwerte und Maxima (grau strichliert) der detektierten Tempi für Testsequenz D mit verschiedenen Varianzwerten (jeweils fünf Wiederholungen, Werte gerundet)

Temposmoothness

Als zweiter beeinflussender Parameter kommt die Temposmoothness in Frage, mit der eingestellt werden kann, wie viele der vergangenen Tempowerte in die aktuelle Schätzung mit einbezogen werden.

Auch hier wurden die originale Testsequenz D sowie die quantisierte Variante als Testmaterial herangezogen, und der Temposmoothness-Parameter einmal auf den Standardwert von 0.1 und einmal auf einen deutlich höheren Wert von 0.6 gesetzt.

Im Gegensatz zum Varianzparameter fällt das in Abbildung 4.4 dargestellte Ergebnis für die Temposmoothness recht deutlich aus. Denn da sich eine Erhöhung der Temposmoothness bei beiden Varianten der Testsequenz negativ auf die Reproduzierbarkeit der Temposchätzungen auswirkt, lässt sich daraus ableiten, dass die Standardeinstellung in diesem Fall die bessere Alternative ist.

4.5.2 Musikalische Interpretation

Wie die Versuchsergebnisse der Varianzwertvariation bereits durchblicken ließen, könnten auch musikalische Gegebenheiten wie etwa Temposchwankungen Auswirkungen auf die Reproduzierbarkeit der Temposchätzungen haben. Deshalb sollte in einem nächsten Schritt untersucht werden, inwieweit die musikalische Interpretation die Leistung des Systems beeinflusst.

Dazu wurden die bei der Partiturverfolgung durch Antescofo ermittelten Tempoverläufe von vier Varianten der Testsequenz D (siehe Abbildung A.4) miteinander verglichen. Das aufgezeichnete MIDI-Original des Stücks bildete das Ausgangsmaterial, welches die Simulation eines langsameren Tempos recht einfach durch verändern der Wiedergabegeschwindigkeit in der DAW ermöglichte. Mit Hilfe des Logical Editors wurde eine Variante generiert, in der die Beginnzeiten der Noten um ±15 Ticks (eine 128stel Note) in Anlehnung an die menschliche Ungenauigkeit einer realen Interpretation zufällig verändert wurden, und als Gegenbeispiel dazu wurde die quantisierte Version aus dem vorherigen Versuch herangezogen.

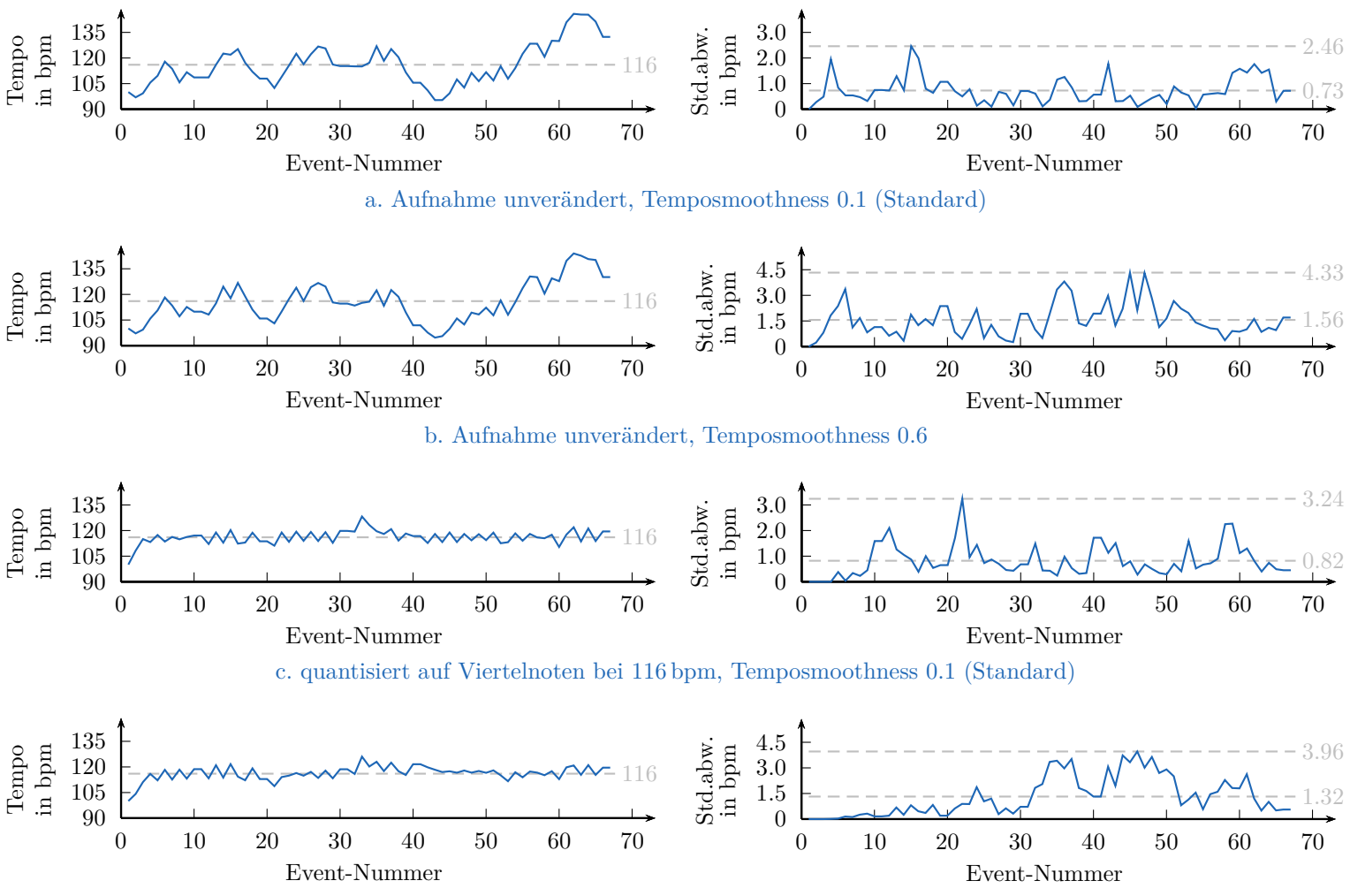


Abbildung 4.4: Mittelwerte (links) und Standardabweichungen (rechts) sowie globale Mittelwerte und Maxima (grau strichliert) der detektierten Tempi für Testsequenz D mit verschiedenen Temposmoothness-Werten (jeweils fünf Wiederholungen, Werte gerundet)

Diese vier Varianten wurden anschließend jeweils fünf Mal wiedergeben, um die dabei von Antescofo geschätzten Tempowerte aufzuzeichnen. Die berechneten Mittelwerte und Standardabweichungen sind in Abbildung 4.5 dargestellt.

Vergleicht man die mittleren Tempoverläufe, so ist erkennbar, dass sich diese in allen Varianten (natürlich mit Ausnahme der quantisierten) qualitativ stark ähneln. Bei Betrachtung der Standardabweichungen zeigt sich, dass eine leichte Abweichung vom Originaltempo kaum Auswirkungen auf die Reproduzierbarkeit der Tempodetektion hat. Eine Vielzahl von Unregelmäßigkeiten bei der musikalischen Interpretation, wie durch die dritte Variante simuliert, hat aber offenbar einen negativen Einfluss, was jedoch anzunehmen war, da die Tempobestimmung an sich dadurch erschwert wird.

Generell ist jedoch festzuhalten, dass die Maxima der Standardabweichungen mit Werten zwischen 2.46 und 3.79 bpm recht hoch ausfallen, wenn man bedenkt, dass dies umgerechnet ca. 2.1 bzw. 3.3 % (bezogen auf das durchschnittliche Tempo der Sequenz von 116 bpm) entspricht.

4.5.3 Musikalisches Material

Um festzustellen, ob die Temposchätzungen eventuell auch vom Notenmaterial selbst beeinflusst werden, wurde der Versuch mit drei weiteren Testsequenzen E bis G (siehe Abbildung A.5 bis Abbildung A.7) statt den Varianten von Sequenz D wiederholt und die Ergebnisse in Abbildung 4.6 gegenübergestellt.

Anhand der globalen Standardabweichungswerte ist zu sehen, dass die Abweichungen zwischen den fünf Wiederholungen bei den Testsequenzen E bis G deutlich geringer ausfallen, als bei Sequenz D. Das musikalische Material selbst hat also offenbar signifikanten Einfluss auf die Reproduzierbarkeit der Detektionswerte. Betrachtet man nun Testsequenz D im Vergleich mit den drei anderen Sequenzen, fallen drei Unterschiede auf.

Testsequenzen D und G zeigen starke Tempoveränderungen über den jeweiligen Verlauf hinweg, während dies bei Sequenzen E und F nicht der Fall ist. Da die Standardabweichungen der Sequenzen F und G jedoch in einem sehr ähnlichen Bereich liegen, ist

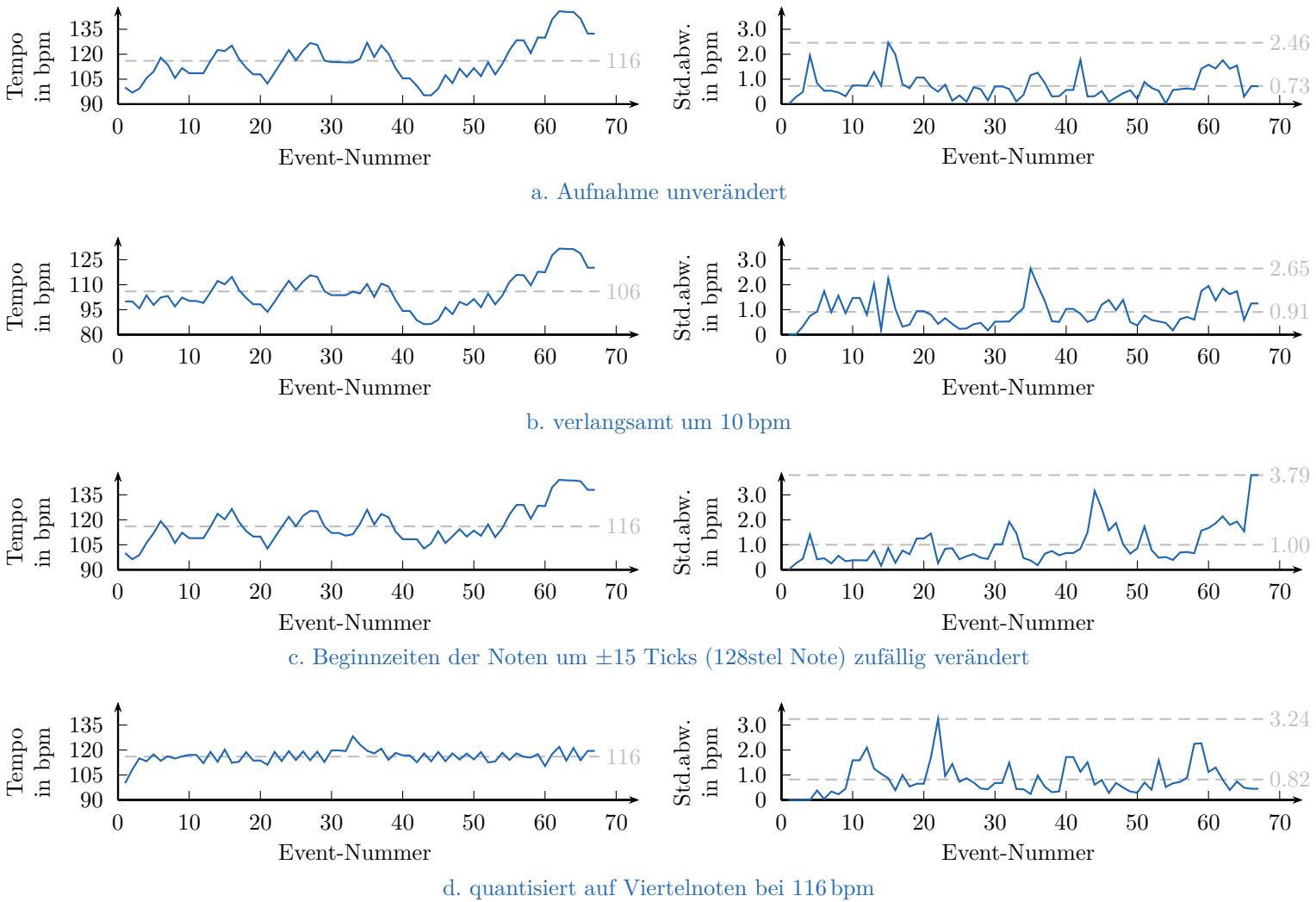
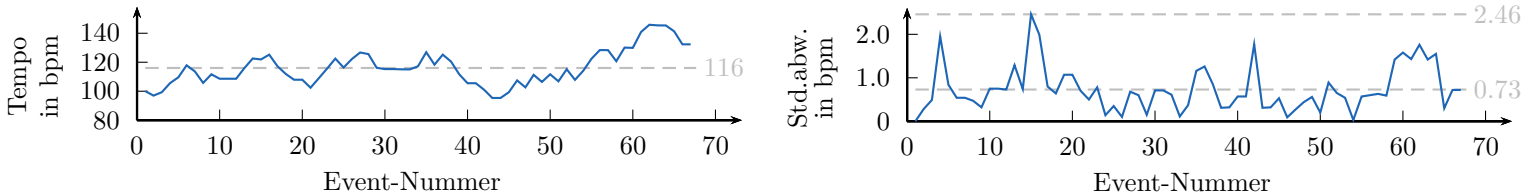
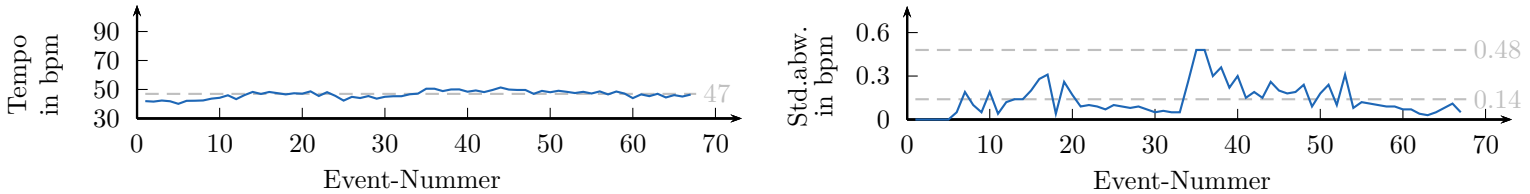


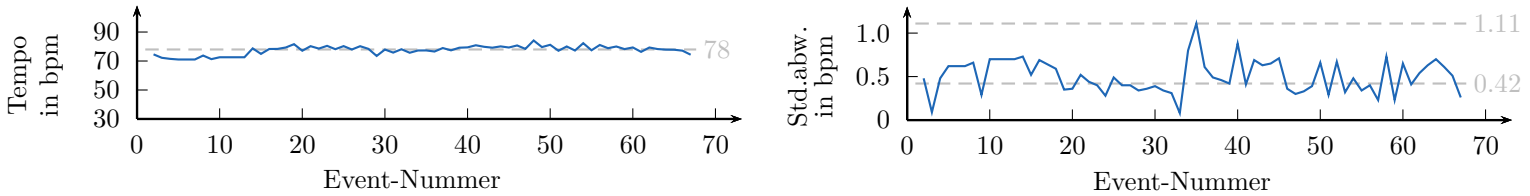
Abbildung 4.5: Mittelwerte (links) und Standardabweichungen (rechts) sowie globale Mittelwerte und Maxima (grau strichliert) der detektierten Tempi für verschiedene Varianten der Testsequenz D (jeweils fünf Wiederholungen, Werte gerundet)



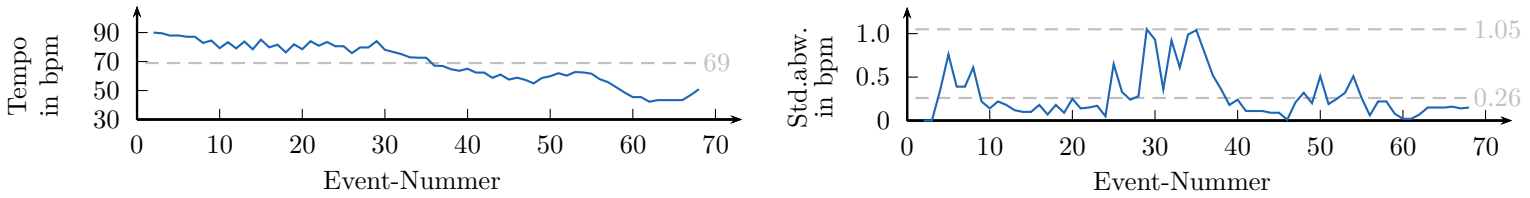
a. Testsequenz D (durchschnittliches Tempo ♩ = 116 bpm, Halbe als häufiger Notenwert)



b. Testsequenz E (durchschnittliches Tempo ♩ = 47 bpm, Achtel als häufiger Notenwert)



c. Testsequenz F (durchschnittliches Tempo ♩ = 78 bpm, Sechzehntel als häufiger Notenwert)



d. Testsequenz G (durchschnittliches Tempo ♩ = 69 bpm, Achtel als häufiger Notenwert)

Abbildung 4.6: Mittelwerte (links) und Standardabweichungen (rechts) sowie globale Mittelwerte und Maxima (grau strichliert) der detektierten Tempi für verschiedene Testsequenzen (jeweils fünf Wiederholungen, Werte gerundet)

anzunehmen, dass diese Eigenschaft keine Auswirkungen auf die Systemperformance hat. Als zweiter Unterschied ist zu erkennen, dass Sequenzen E bis G ein deutlich niedrigeres (durchschnittliches) Tempo aufweisen, als Sequenz D. Betrachtet man weiters die Notentexte der Testsequenzen (siehe Abbildung A.4 bis Abbildung A.7), lässt sich im Notenmaterial selbst der dritte Unterschied feststellen: In Testsequenz D tritt die Halbe Note als überwiegend verwendeter Notenwert auf, in den anderen Sequenzen dagegen die Achtel- oder gar die Sechzehntelnote. Es ist durchaus denkbar, dass diese kleineren Notenwerte die Temposchätzung begünstigen, weil damit auch die Abstände zwischen den Schätzungen kleiner werden.

Da aus den bisherigen Ergebnissen nicht ablesbar war, welche Unterschiede tatsächlich für die festgestellte Verbesserung der Systemperformance verantwortlich sind, wurden diese in der Folge näher untersucht.

Als erste Eigenschaft wurde das durchschnittliche Tempo betrachtet, welches direkt mit dem notierten Tempo zusammenhängt, da sich eine Musikerin bzw. ein Musiker im Normalfall an diesem orientiert. Um deren Auswirkungen zu testen, wurde Testsequenz D als jenes Beispiel, das im Vergleich der Testsequenzen sowohl das höchste durchschnittliche Tempo als auch die schlechtesten Werte aufwies, mit Hilfe der Tempoeinstellung der DAW einfach mit einem Viertel der originalen Geschwindigkeit wiedergegeben. Dies entspricht effektiv der Reduktion des notierten Tempos von $\text{♩} = 100\,\text{bpm}$ auf $\text{♩} = 25\,\text{bpm}$, das Notenmaterial bleibt jedoch unberührt.

Bei der Erstellung eines Testbeispiels für die zweite Eigenschaft, den überwiegend vorkommenden Notenwert, war genau das Gegenteil gewünscht: Alle Notenwerte sollten auf ein Viertel ihrer Dauer skaliert werden (um, ausgehend von den Halben Noten im Original, Achtelnoten als überwiegenden Notenwert zu erreichen), das notierte Tempo und der damit verbundene Detektionsbereich des Score-Followers sollten sich jedoch nicht verändern. Um dies zu erreichen, wurde eine neue Version der Antescofo-Partitur von Testsequenz D erstellt, in der die Notenwerte entsprechend angepasst wurden: Halbe Noten im Original wurden nun als Achtel geschrieben und Viertelnoten als Sechzehntel. Da Notenwerte in der Antescofo-Partitur durch Zahlenwertparameter der jeweiligen Event-Befehle repräsentiert werden (vgl. Abschnitt 2.3.2), konnte die Anpassung recht einfach durch Division aller dieser Werte durch die Zahl 4 erreicht werden. Real wäre

das Teststück damit bei unveränderter Tempoangabe im Vergleich zur Originalnotation vier Mal so schnell zu interpretieren. Deshalb musste die Wiedergabegeschwindigkeit in der DAW auf das Vierfache des Originaltempos angehoben werden, um den gewünschten Testfall zu simulieren und das notierte Tempo bei $\text{♩} = 100\,\text{bpm}$ zu belassen.

Nachdem nun beide Testbeispiele bereit waren, konnte der Versuch gestartet werden. Dieser lief ebenfalls analog zu den vorherigen Versuchen ab: Das Testmaterial wurde entsprechend den obigen Beschreibungen und unter Verwendung der entsprechenden Partitur-Version wiedergegeben, um die in fünf Wiederholungen von Antescofo detektierten Tempi festzuhalten und zu vergleichen.

Die in Abbildung 4.7a bis Abbildung 4.7c dargestellten Ergebnisse zeigten jedoch im Wesentlichen nur, dass die Skalierung der Notenwerte zu einer Kompression des Tempoverlaufs führt (ein weiterer, informeller Test mit quantisiertem Material bestätigte, dass sich diese Kompression auch auf die Amplitude der in Abschnitt 4.5.1 diskutierten Oszillationen um einen gleichbleibenden Tempowert auswirkte und diese damit verringerte). Laut den berechneten Standardabweichungen verbessert jedoch offenbar keine der beiden Eingenschaften die Reproduzierbarkeit der Temposchätzung.

Da aber zuvor im Vergleich der Testsequenzen die Möglichkeit der Verbesserung nachgewiesen wurde, konnte das nur eines bedeuten: Beide Eigenschaften gemeinsam mussten für den aufgetretenen, positiven Effekt verantwortlich sein.

Um dies zu verifizieren, wurde ein drittes Testbeispiel erstellt, für das sowohl Tempo als auch Notenwerte auf ein Viertel des Originals skaliert wurden. Dies entspricht jedoch bloß einer Recodierung des musikalischen Materials, das heißt, die Darbietung der recodierten Variante würde trotz der beschriebenen Notentextveränderungen genau gleich klingen, wie jene des Originals. Deshalb konnte dieser Test auch mit einfacher Geschwindigkeit der Wiedergabe durchgeführt werden.

Bei Betrachtung der Ergebnisgrafiken in Abbildung 4.7d bestätigen diese, dass eine solche Recodierung tatsächlich in der Lage ist, die Systemperformance deutlich zu verbessern. Aus der Gegenüberstellung der globalen Standardabweichungen ergibt sich nämlich ein ungefährer Faktor von 13 bis 20.

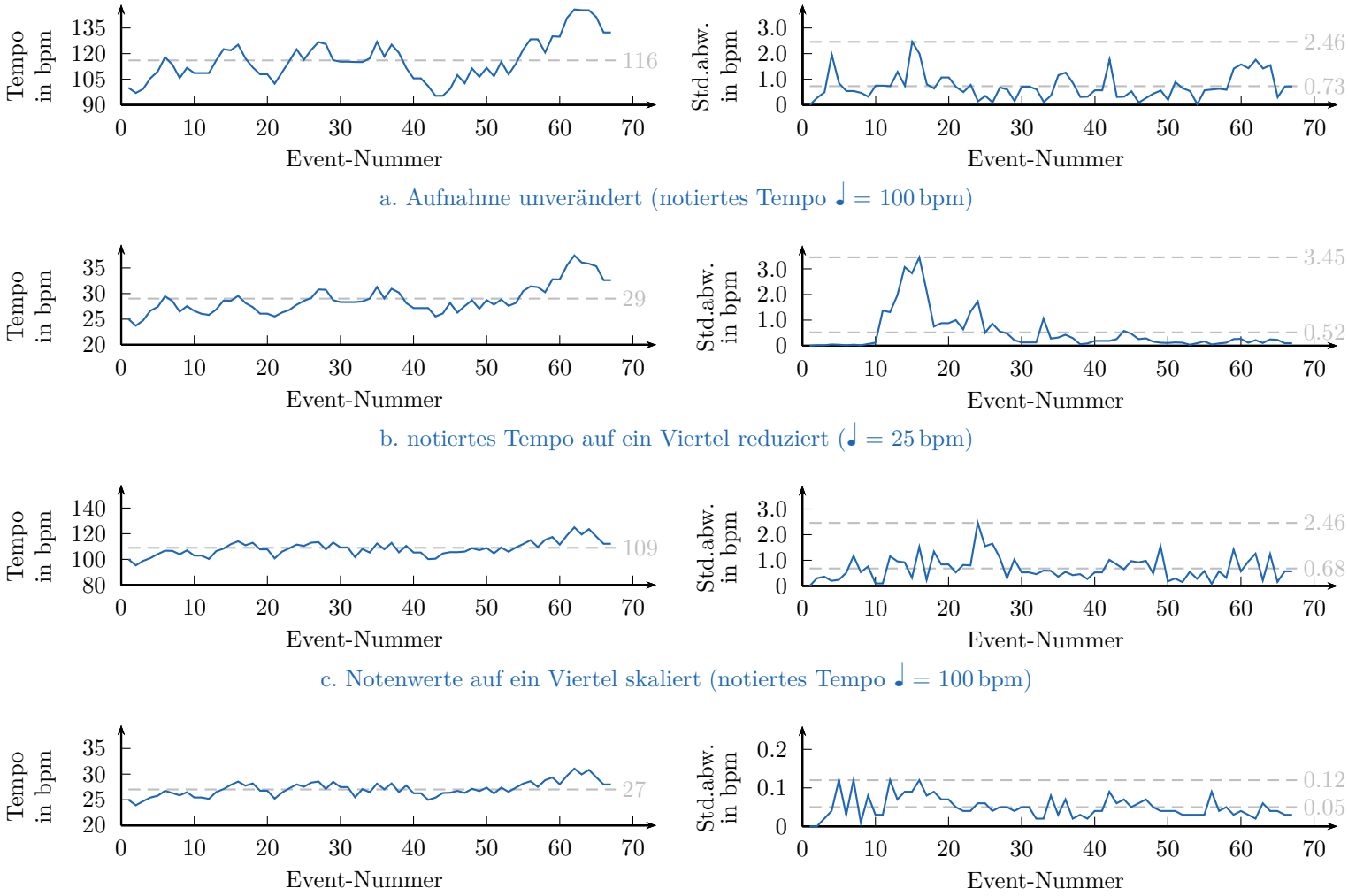


Abbildung 4.7: Mittelwerte (links) und Standardabweichungen (rechts) sowie globale Mittelwerte und Maxima (grau strichliert) der detektierten Tempi für Skalierungsvarianten von Testsequenz D (Varianzwert 0.1, jeweils fünf Wiederholungen, Werte gerundet)

4.5.4 Synchronisationsstrategien

Im Gespräch mit Antescofo-Projektleiter Arshia Cont erwähnte dieser noch eine weitere Möglichkeit, die Reproduzierbarkeit der Temposchätzungen, und damit auch jene der Auslösezeitpunkte, zu erhöhen: die Wahl alternativer Synchronisationsstrategien (vgl. Abschnitt 2.3.2) für die Registrierungskommando-Actions in der Antescofo-Partitur.

Die *Loose*-Strategie wurde ohnehin bereits implizit mituntersucht, da diese der Standardeinstellung entspricht (siehe z.B. Abschnitt 4.4). Und da bereits erste Versuche mit der *Tight*-Strategie zeigten, dass sich die Auslösezeitpunkte der Actions (vermutlich aufgrund der mit dieser Strategie verbundenen Ausrichtung an detektierten Events) nicht genau genug einstellen lassen, wurde diese Variante nicht weiterverfolgt. Damit blieb nur die *Target*-Strategie ausgiebiger zu untersuchen.

Dafür wurde wieder Testsequenz D herangezogen und eine neue *Target*-Version der Antescofo-Partitur erstellt. Dazu wurden alle Registrierungskommandos in einer `GROUP`-Action (siehe Abschnitt 2.3.2) zusammengefasst und diese an das erste Event der Sequenz gebunden. Anschließend waren natürlich auch alle Delay-Werte anzupassen, da sie in diesem Fall nun vom ersten Event bzw. dem jeweils vorherigen Registrierungskommando aus gerechnet wurden. In einem ersten Schritt wurden Basiswerte entsprechend der laut Notentext gegebenen Abstände in Schlägen verwendet, die dann empirisch verfeinert wurden, um möglichst nahe an die aufgezeichneten Referenzzeitpunkte für die Registrierungswechsel heranzukommen.

Außerdem musste der `GROUP`-Action das Attribut `@target` zugeordnet und ein Zeithorizont für die Interpolation zwischen dem internen `GROUP`-Tempo und dem jeweils aktuell detektierten Tempo festgelegt werden. Um herauszufinden, wie groß der Zeithorizont eingestellt werden sollte, wurde Testsequenz D mit verschiedenen Werten jeweils fünf Mal wiedergegeben, die resultierenden Zeitpunkte der Registrierungswechsel aufgezeichnet, und jener Zeithorizont ausgewählt, der die besten Ergebnisse lieferte (um die Unabhängigkeit von leicht unterschiedlichen Tempi bei der Interpretation durch Musikerinnen und Musiker zu erhalten, wurden nur relative Zeithorizonte getestet, obwohl auch absolute Zeitangaben einstellbar wären).

Wie aus Tabelle 4.6 hervorgeht, führt ein Zeithorizont von zwei Schlägen zu den kleinsten maximalen Standardabweichungen, und wurde deshalb für die Verwendung in den folgenden Versuchen ausgewählt.

Zeithorizont in Schlägen	1.0	2.0	5.0	10.0	15.0
mittlere Std.abw. in ms	20.8	18.9	21.0	28.3	67.4

Tabelle 4.6: Zusammenfassung der Testergebnisse zur Auswahl eines *Target*-Zeithorizonts (Varianzwert 0.1, fünf Wiederholungen, Werte gerundet)

Zusätzlich zu den Synchronisationsstrategien, die Antescofo selbst zur Verfügung stellt, entstand aufgrund der gängigen Registrierungspraxis die Idee für eine weitere, sozusagen manuelle Strategie zur Action-Synchronisation. Bei näherer Betrachtung der manuell vom Organisten selbst durchgeführten Registrierungswechsel im aufgezeichneten MIDI-Material ist nämlich zu erkennen, dass die Wechsel zum Großteil mit jenen Stellen zusammenfallen, an denen Tasten losgelassen werden, was sich in den MIDI-Daten als *Note-off*-Befehl niederschlägt.

Nun kann Antescofo solche *Note-offs* zwar nicht direkt erkennen, ist aber in der Lage, Akkordveränderungen und Pausen zu detektieren. Damit ist es möglich, eine solche *Note-off*-Strategie durch geschicktes Umschreiben der betroffenen Stellen manuell umzusetzen. Bei abgesetzten Akkordübergängen wäre dafür das Registrierungskommando lediglich ohne Delay an eine dazwischen eingefügte Pause der Länge Null (Befehl `NOTE 0 0`) zu binden. Im Beispielfall eines Akkordüberganges, bei dem ein Ton liegen bleibt, könnte der erste Akkord um einen Bruchteil gekürzt und der Liegeton als eigenes Event mit der übrigen Länge notiert werden, an welches dann die Action gebunden werden könnte, um beim Loslassen der restlichen Akkordtöne auszulösen.

In einer Versuchsreihe sollte nun getestet werden, welche der drei Synchronisationsstrategien (*Loose*, *Target* und *Note-off*) zu den besten Ergebnisse führt. Dazu wurden die beiden Testsequenzen D und E unter Einsatz der verschiedenen Strategien wiedergegeben und die Zeitstempel der ausgelösten Registrierungswechsel festgehalten. Tabelle 4.7 und Tabelle 4.8 zeigen die Auswertungen der resultierenden Zeitpunkte für die jeweiligen Versuche.

Auswertungsgröße	Strategie		
	Loose	*Target*	*Note-off*
Maximale Standardabweichung in ms	25.0	24.8	9.0
Schwankungsbreite in ms (68.2 % Fallabdeckung)	50.1	49.5	7.7
Schwankungsbreite in ms (95.4 % Fallabdeckung)	100.2	99.1	15.3
Schwankungsbreite in ms (99.6 % Fallabdeckung)	150.3	148.6	23.0

Tabelle 4.7: Zusammenfassung der Strategietests für Testsequenz D (Varianzwert 0.1, jeweils fünf Wiederholungen, Werte gerundet)

Auswertungsgröße	Strategie		
	Loose	*Target*	*Note-off*
Maximale Standardabweichung in ms	5.2	4.9	10.0
Schwankungsbreite in ms (68.2 % Fallabdeckung)	10.4	9.7	7.7
Schwankungsbreite in ms (95.4 % Fallabdeckung)	20.7	19.5	15.4
Schwankungsbreite in ms (99.6 % Fallabdeckung)	31.1	29.2	23.1

Tabelle 4.8: Zusammenfassung der Strategietests für Testsequenz E (Varianzwert 0.1, jeweils fünf Wiederholungen, Werte gerundet)

Anhand der Ergebnisse lässt sich unabhängig von der verwendeten Testsequenz feststellen, dass die *Target*-Strategie keine signifikante Verbesserung in Bezug auf die Reproduzierbarkeit der Auslösezeitpunkte bringt. Obwohl es den Werten nach egal wäre, welche der beiden Varianten verwendet wird, ist dennoch der Einsatz der *Loose*-Strategie zu empfehlen, da die notwendige Feinabstimmung der Delay-Werte ohne die *Target*-Interpolation im Hintergrund einfacher durchführbar ist. Die *Loose*-Strategie ist damit die einzig verbliebene, die Delays verwendet, und wird deshalb in den folgenden Abschnitten zum besseren Verständnis als *Delay*-Strategie bezeichnet.

Im Gegensatz dazu zeigt die Auswertung für die *Note-off*-Strategie erfreuliche Ergebnisse: Diese liegen für beide Testsequenzen bereits ohne Recodierung des musikalischen Materials als zusätzliche Verbesserungsmaßnahme (siehe Abschnitt 4.5.3) in einem akzeptablen Bereich. Die Tatsache, dass sich die Werte der beiden Testsequenzen sehr stark ähneln, lässt sogar darauf schließen, dass eine solche Maßnahme keinen Effekt hätte, was bei etwas Überlegung auch logisch erscheint: Bei Einsatz der *Note-off*-Strategie sind keine Delays mehr nötig, was die Auslösung der Registrierungsumschaltungen unabhängig von der Temposchätzung und deren Schwankungen macht.

Die *Note-off*-Strategie sollte der *Loose*-Strategie also wo immer möglich vorgezogen werden. Da sie jedoch nicht in allen Fällen eingesetzt werden kann, ist die Anwendung der Recodierung dennoch sinnvoll, da die Ergebnisse der *Loose*-Strategie zwar im Fall von Testsequenz E gerade noch innerhalb des in Abschnitt 4.2 ermittelten Toleranzbereichs von ca. 30 ms liegen, jene für Testsequenz D ohne diese Verbesserungsmaßnahme aber inakzeptabel sind.

4.6 Abschließender Testlauf

Zum Abschluss wurde deshalb ein letzter Testlauf durchgeführt, mit dem festgestellt werden sollte, ob nun unter Umsetzung der Empfehlungen und Zuhilfenahme der ermittelten Verbesserungsmaßnahmen auch im Fall der bisher problematischen Testsequenz D (siehe Abbildung A.4) eine akzeptable Systemperformance möglich ist.

Für diesen Versuch wurde also der Varianzwert auf 0.1 gesetzt, die Temposmoothness beim Standardwert von 0.1 belassen und die auf ein Viertel des Tempos und der Notenwerte recodierte Antescofo-Partitur verwendet. Der Versuchsablauf gleicht dem im vorherigen Abschnitt beschriebenen, außer dass nur die beiden bevorzugten Synchronisationsstrategien *Delay* und *Note-off* getestet wurden.

Da es sich um den abschließenden Testlauf handelt, werden nun neben dem Endergebnis in Tabelle 4.11 auch die den Auswertungen zugrunde liegenden Einzeldaten in Tabelle 4.9 und Tabelle 4.10 dargestellt.

Wie die Auswertungen erfreulicherweise zeigen, liegen die Ergebnisse in nahezu 100 % der *Note-off*-Fälle innerhalb des in Abschnitt 4.2 ermittelten Toleranzbereichs von ca. 30 ms, und für jene Stellen, bei der die *Delay*-Strategie zum Einsatz kommt, wird eine Fallabdeckung von etwa 95 % erreicht.

Registrierung	Zeitstempel in ms						Mittelwert in ms	Std.abw. in ms
	Ref.[1]	1	2	3	4	5		
10/2	10125	10124	10126	10120	10126	10127	10124.6	2.8
10/3	20084	20069	20078	20081	20078	20073	20075.8	4.8
10/4	29584	29597	29600	29600	29599	29587	29596.6	5.5
10/5	39727	39735	39722	39736	39722	39721	39727.2	7.6

Tabelle 4.9: Ergebnisse des abschließenden Testlaufs mit *Delay*-Strategie (Testsequenz D, recodiert auf ♩ = 25 bpm, Varianzwert 0.1, fünf Wiederholungen, Werte gerundet)

Registrierung	Zeitstempel in ms						Mittelwert in ms	Std.abw. in ms
	Ref.[1]	1	2	3	4	5		
10/2	10125	10263	10256	10264	10256	10263	10260.4	4.0
10/3	20084	20143	20135	20144	20136	20143	20140.2	4.3
10/4	29584	29570	29574	29571	29563	29571	29569.8	4.1
10/5	39727	39729	39722	39718	39723	39728	39724.0	4.5

Tabelle 4.10: Ergebnisse des abschließenden Testlaufs mit *Note-off*-Strategie (Testsequenz D, recodiert auf ♩ = 25 bpm, Varianzwert 0.1, fünf Wiederholungen, Werte gerundet)

Auswertungsgröße	Strategie	
	Delay	*Note-off*
Maximale Standardabweichung in ms	7.6	11.0
Schwankungsbreite in ms (68.2 % Fallabdeckung)	15.2	9.1
Schwankungsbreite in ms (95.4 % Fallabdeckung)	30.4	18.1
Schwankungsbreite in ms (99.6 % Fallabdeckung)	45.6	27.2

Tabelle 4.11: Zusammenfassung der Ergebnisse des abschließenden Testlaufs (Testsequenz D, recodiert auf ♩ = 25 bpm, Varianzwert 0.1, jeweils fünf Wiederholungen, Werte gerundet)

[1] Da die Spannweite der Referenzzeitstempel pro Registrierung maximal 1 ms betrug, wird hier aus Platzgründen nur der Medianwert angeführt.

5 Prototypischer Workflow

Das folgende Kapitel soll nun abschließend anhand eines prototypischen Workflows die notwendigen und empfohlenen Arbeitsschritte im Laufe der Systembenutzung am Beispiel der Verwendung mit der am Institut für Kirchenmusik und Orgel zur Verfügung stehenden Rodgers-Orgel darstellen.

5.1 Erstellen der Partitur

Vor der Anwendung des vorliegenden Systems zur automatischen Setzersteuerung muss der Notentext des gewünschten Stückes zunächst in eine Form übertragen werden, die vom Score-Follower ausgewertet werden kann. Dazu muss dieser unter Verwendung der den Score-Follower begleitenden Partitursprache in eine Kette sogenannter Event-Befehle übersetzt und als Textdatei abgespeichert werden, in der jede Note, jede Pause, jeder Akkord als separates Partitur-Event dargestellt wird.

Dieser Prozess kann entweder von Hand erledigt werden, was jedoch je nach Länge des Notentextes sehr aufwändig sein kann, oder man bedient sich der automatischen Importfunktion des Zusatzprogramms AscoGraph, das als Editor und grafische Anzeige der Partitur für Antescofo gedacht ist. Es ist in der Lage, MIDI- oder MusicXML-Dateien einzulesen und automatisch in eine Partitur im Antescofo-Format umzuwandeln, wobei die Verwendung von MusicXML vorzuziehen ist, da komplexere Inhalte (wie etwa Triller o.ä.) im MIDI-Format nicht abgebildet und damit auch nicht übersetzt werden können. Um eine vorliegende Datei zu importieren, genügt es, sie über das Dateimenü in AscoGraph zu öffnen oder einfach mit der Maus in das Fenster der Partituranzeige zu ziehen.

Für sehr bekannte Stücke werden Partituren im MusicXML-Format im Internet angeboten, andernfalls müssen sie mit Hilfe eines Notensatzprogramms erstellt werden[1]. Prinzipiell könnte auch hier der Notentext manuell eingegeben werden, was zwar etwas weniger aufwändig und fehleranfällig ist, als direkt im Antescofo-Format, aber dennoch je nach Länge des Stücks sehr mühsam. Doch auch Notenprogramme bieten meist eine Funktion zum Import von MIDI-Dateien an, die eventuell genutzt werden kann, falls eine entsprechende MIDI-Datei vorliegt oder im Internet zu finden ist (was je nach Literatur mehr oder weniger wahrscheinlich, aber jedenfalls deutlich häufiger der Fall ist, als bei MusicXML-Dateien). Außerdem stellen einige Notenprogramme auch die Möglichkeit bereit, gedruckten Notentext einzuscannen und auf diesem Weg zu importieren (z.B. bei Finale im Menü `File > Import > SmartScore Lite Scan` zu finden), was sich in den meisten Fällen anbietet, da die Musikerin bzw. der Musiker den Notentext üblicherweise zumindest in gedruckter Form bereitstellen kann.

Auf welchem Weg der Notentext auch ins Notensatzprogramm gelangte, er sollte danach in jedem Fall noch einmal manuell überprüft werden. Dabei gilt es, sowohl eventuelle Tipp- oder Importfehler zu korrigieren, sowie sicherzustellen, dass an allen Stellen die üblichen, dafür vorgesehenen Zeichen verwendet werden (z.B. dass Tempoangaben wirklich als solche notiert sind, und nicht als bloßer Text, oder dass nur herkömmliche ovale Notenköpfe, etwa im Gegensatz zu rautenförmigen für Flageoletts o.ä., benutzt werden). Ist der Notentext fertiggestellt, kann dieser schließlich abgespeichert, als MusicXML-Datei exportiert (bei Finale im Menü `File > Export > MusicXML`) und anschließend in AscoGraph importiert werden.

5.2 Eventuelle Recodierung

Um eine zufriedenstellende Systemleistung zu erreichen, sollte außerdem darauf geachtet werden, dass der am häufigsten vorkommende Notenwert des Stücks etwa einer Achtelnote entspricht (vgl. Abschnitt 4.5.3). Erfüllt das gewünschte Stück diese Eigenschaft

[1] Im Antescofo-Handbuch wird Finale als Notensatzprogramm empfohlen, da die Importfunktion von AscoGraph auf mit diesem erstellte MusicXML-Dateien optimiert ist.

nicht, kann dies dennoch mit Hilfe einer entsprechenden Recodierung des musikalischen Materials erreicht werden.

Dazu müssen alle Notenwerte des Stücks soweit reduziert werden, dass der häufigste zu einer Achtel wird, während das Tempo ebenfalls im gleichen Maß zu skalieren ist, damit die recodierte Version dennoch exakt gleich klingt, wie das Original. Als Beispiel sollen hier die ersten Takte der Testsequenz D herangezogen werden. Der häufigste Notenwert ist dort die halbe Note, soll aber zu einer Achtelnote werden, d.h. alle Notenwerte müssen auf ein Viertel ihrer Länge reduziert und das Tempo ebenfalls durch die Zahl 4 dividiert werden. In Abbildung 5.1 und 5.2 bzw. Listing 5.1 und 5.2 sind die Notentexte bzw. Antescofo-Partituren vor und nach der Recodierung gegenübergestellt.

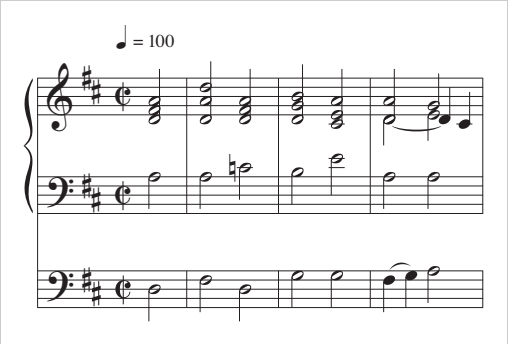


Abbildung 5.1: originaler Notentext für Testsequenz D, Takte 1-3 mit Auftakt

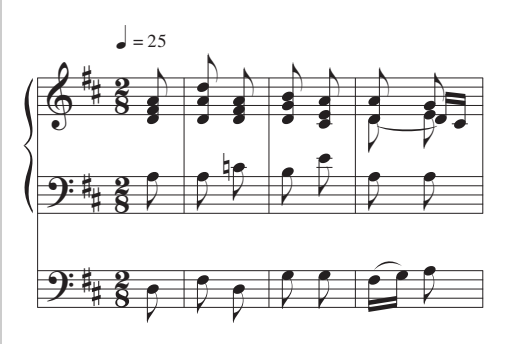


Abbildung 5.2: recodierter Notentext für Testsequenz D, Takte 1-3 mit Auftakt

```
1  BPM 100
2
3  CHORD (D3 A3 D4 F#4 A4) 2 pickupmeasr
4
5  CHORD (F#3 A3 D4 A4 D5) 2 measure1
6  CHORD (D3 C4 D4 F#4 A4) 2
7
8  CHORD (G3 B3 D4 G4 B4) 2 measure2
9  CHORD (G3 C#4 E4 A4) 2
10
11 CHORD (F#3 A3 D4 A4) 1 measure3
12 CHORD (G3 -A3 -D4 -A4) 1
13 CHORD (A3 D4 E4 G4) 1
14 CHORD (-A3 C#4 -E4 -G4) 1
```

Listing 5.1: originale Antescofo-Partitur für Testsequenz D, Takte 1-3 mit Auftakt

```
1  BPM 25
2
3  CHORD (D3 A3 D4 F#4 A4) 1/2 pickupmeasr
4
5  CHORD (F#3 A3 D4 A4 D5) 1/2 measure1
6  CHORD (D3 C4 D4 F#4 A4) 1/2
7
8  CHORD (G3 B3 D4 G4 B4) 1/2 measure2
9  CHORD (G3 C#4 E4 A4) 1/2
10
11 CHORD (F#3 A3 D4 A4) 1/4 measure3
12 CHORD (G3 -A3 -D4 -A4) 1/4
13 CHORD (A3 D4 E4 G4) 1/4
14 CHORD (-A3 C#4 -E4 -G4) 1/4
```

Listing 5.2: recodierte Antescofo-Partitur für Testsequenz D, Takte 1-3 mit Auftakt

Um die Recodierung vorzunehmen, gibt es zwei Möglichkeiten: Entweder man führt diese in der Antescofo-Partitur händisch mit einem Texteditor (oder dessen Suchen/Ersetzen-Funktion) durch, was je nach Länge des Stückes wiederum aufwändig und fehleranfällig sein kann, oder man recodiert das Notenmaterial noch im Notensatzprogramm vor dem MusicXML-Export.

In Finale sind dazu die folgenden Schritte nötig: Zunächst wird das gesamte Notenmaterial mit dem Selection-Tool markiert (bzw. mit der Tastenkombination `Strg+A`) und die Funktion `Utilities > Change > Note Durations` aufgerufen. Dort wird `Change all note duration by` aktiviert, ein Skalierungsfaktor ausgewählt (im Fall dieses Beispiels 25 %, da die Notenlänge einer Achtelnote dem Viertel einer halben Note entspricht), das Häkchen bei `Rebar music` entfernt und die Skalierung mit Klick auf `OK` durchgeführt. Beginnt das Stück mit einem Auftakt, so ist dieser im nächsten Schritt über das Menü `Document > Pickup measure` separat anzugleichen (im Beispiel ist also eine Achtelnote auszuwählen). Anschließend muss mit dem Time-Signature-Tool auf den ersten Takt doppelgeklickt und die Taktart geändert werden, im Beispiel von 2/2 auf 2/8. In einem letzten Schritt ist noch die Tempoangabe anzupassen (im Beispielfall von ♩ = 100 bpm auf ♩ = 25 bpm), indem mit dem Expression-Tool auf dessen Griff doppelgeklickt, im sich öffnenden Fenster die Schaltfläche `Edit` ausgewählt und im Reiter `Main` des Unterfensters der Text entsprechend geändert wird.

5.3 Eintragen der Registrierungswechsel

Sobald die Antescofo-Partitur erstellt und (falls notwendig) recodiert wurde, können die Registrierungswechsel eingetragen werden. Dazu wurde im Rahmen dieser Arbeit das Registrierungskommando `orgReg` geschaffen, mit dem in der Setzereinrichtung der Orgel gespeicherte Registrierungen direkt oder durch ein Delay verzögert aktiviert werden können.

Für das Eintragen der Kommandos wurden zwei empfohlene Varianten herausgearbeitet: die *Delay*- und die *Note-off*-Strategie (vgl. Abschnitt 4.5.4). Unabhängig von der verwendeten Strategie müssen die Registrierungswechseleinträge jeweils direkt an jenes Event in der Partitur gebunden werden, das der gewünschten Stelle für den Wechsel

vorausgeht. Nachfolgend soll die Verwendung des Kommandos anhand einiger Beispiele demonstriert werden.

Startregistrierung eines Stücks

Um eine bestimmte gespeicherte Registrierung zu aktivieren, bietet sich die Verwendung des Registrierungskommandos mit einer absoluten Setzerposition an. Im folgenden Beispiel (Abbildung 5.3 bzw. Listing 5.3) soll die Speicherposition 1 im Setzerlevel[1] 11 aktiviert werden. Dies ist mit dem Kommando `orgReg memory 11 1` möglich.

Damit diese Registrierung bereits vor Beginn des Stücks aktiv wird, muss das Kommando an das davor liegende Event gebunden werden. Im Beispiel wäre dies das `BPM`-Event der Tempoangabe[2]. Da die Registrierung sofort bei Verarbeitung aktiviert werden soll, beträgt die Dauer der Verzögerung 0 Sekunden und muss damit nicht eingetragen werden. Die Einrückung des Kommandos wurde aus Gründen der besseren Sichtbarkeit vorgenommen, ist jedoch optional.

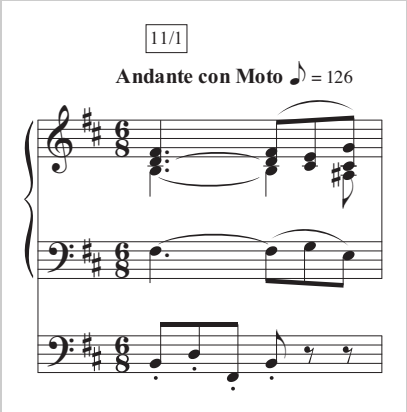


Abbildung 5.3: Notentext für Testsequenz E, Takt 1

```
1  BPM 42
2    orgReg memory 11 1
3
4  CHORD (B2 F#3 B3 D4 F#4) 1/3 measure1
5  CHORD (D3 -F#3 -B3 -D4 -F#4) 1/3
6  CHORD (F#2 -F#3 -B3 -D4 -F#4) 1/3
7  CHORD (B2 -F#3 -B3 -D4 -F#4) 1/3
8  CHORD (G3 -B3 C#4 E4) 1/3
9  CHORD (E3 A#3 -C#4 G4) 1/3
```

Listing 5.3: Antescofo-Partitur für Testsequenz E, Takt 1

[1] Ein Level einer Setzereinrichtung wird auch als Bank bezeichnet.

[2] Da bei der Notentexterstellung eine Gruppierung der Schläge in zweimal drei Achtel eingestellt wurde, hat AscoGraph bei der Konversion der MusicXML-Datei die Zeitbasis auf eine punktierte Viertelnote festgelegt, weshalb die Tempoangabe `BPM 42` lautet.

Registrierungswechsel zwischen abgesetzten Klängen

Im folgenden Beispiel (Abbildung 5.4 bzw. Listing 5.4) findet ein Registrierungswechsel von der vorherigen zur direkt darauffolgenden Setzerposition beim Übergang zwischen zwei abgesetzten Akkorden, also ohne Legatobindung, statt. Dafür kann das relative Kommando `orgReg next` verwendet werden. Dieses könnte theoretisch mit beiden der erwähnten Strategien in die Partitur eingetragen werden, weshalb hier der Unterschied demonstriert werden soll.

Wird die *Delay*-Strategie verwendet, muss das Kommando mit einer Verzögerung an das vierte Event des Taktes gebunden werden (siehe Listing 5.4a), um erst zum Beginn des fünften Events den Wechsel auszulösen (eine Bindung an das fünfte Event selbst ist nicht möglich, da der Wechsel sonst aufgrund der Detektionslatenz von Antescofo erst kurz nach Beginn des Events wirksam würde).

Die Dauer der Verzögerung muss jedoch erst bestimmt werden, was z.B. mit Hilfe einer MIDI-Aufnahme möglich ist, indem der Zeitabstand zwischen dem aufgezeichneten, manuell ausgelösten Wechsel und dem vorhergehenden Akkordbeginn gemessen wird (in diesem Beispiel 0.327 s). Obwohl bereits dieser Zeitwert eingetragen werden könnte, sollte er anschließend in einen relativen Wert entsprechend dem Tempo umgerechnet werden, damit sich die Verzögerung leicht unterschiedlichen Tempi anpassen kann. Aufgrund der Detektionsschwankungen von Antescofo ist dafür ein über mehrere Durchläufe des Stücks gemittelter Tempowert an der Wechselstelle zu errechnen (hier 49.738 bpm), der durch Multiplikation mit dem Tempowert und Division durch die Zahl 60 in die relative Verzögerung umgerechnet werden kann (0.271 Schläge). Diese stellt einen guten Ausgangswert dar, bedarf jedoch in den meisten Fällen noch manueller Anpassungen in mehreren Testläufen.

An dieser Beschreibung sind jedoch bereits die beiden Nachteile der *Delay*-Strategie zu erkennen: Erstens ist die Prozedur der Verzögerungsermittlung recht aufwändig, und zweitens ist der Auslösezeitpunkt des Registrierungswechsels direkt abhängig von den Schwankungen bei der Temposchätzung des Score-Followers, was zu einer nicht ganz zufriedenstellenden Position des Wechsels führen könnte.

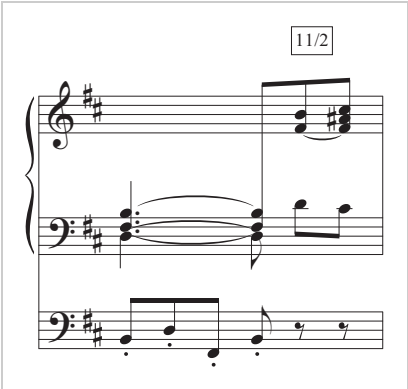


Abbildung 5.4: Notentext für Testsequenz E, Takt 8

```
1  CHORD (B2 D3 F#3 B3) 1/3 measure8
2  CHORD (D3 -D3 -F#3 -B3) 1/3
3  CHORD (F#2 -D3 -F#3 -B3) 1/3
4  CHORD (B2 -D3 -F#3 -B3) 1/3
5    0.271 orgReg next
6  CHORD (D4 F#4 B4) 1/3
7  CHORD (C#4 -F#4 A#4 C#5) 1/3
```

a. *Delay*-Strategie

```
1  CHORD (B2 D3 F#3 B3) 1/3 measure8
2  CHORD (D3 -D3 -F#3 -B3) 1/3
3  CHORD (F#2 -D3 -F#3 -B3) 1/3
4  CHORD (B2 -D3 -F#3 -B3) 1/3
5  NOTE 0 0  // dummy silence
6    orgReg next
7  CHORD (D4 F#4 B4) 1/3
8  CHORD (C#4 -F#4 A#4 C#5) 1/3
```

b. *Note-off*-Strategie

Listing 5.4: Antescofo-Partitur für Testsequenz E, Takt 8

Deshalb ist die im Rahmen dieser Arbeit vorgeschlagene *Note-off*-Strategie wo immer möglich zu bevorzugen, da die oben genannten Nachteile auf sie nicht zutreffen. Die *Note-off*-Strategie ist von der gängigen Praxis inspiriert, in der ein Registrierungswechsel häufig am Ende eines Klanges vorgenommen wird. Da Antescofo selbst jedoch nicht in der Lage ist, das Ende eines Events zu detektieren, muss es durch geschicktes Umschreiben der Partitur dabei unterstützt werden.

Im oben gezeigten Beispiel (Abbildung 5.4 bzw. Listing 5.4b) ist dies durch Einschieben einer sogenannten Dummy-Silence recht einfach möglich, da das Absetzen zwischen den beiden Akkorden damit von Antescofo als Beginn eines (wenn auch sehr kurzen) Pausen-Events interpretiert und erkannt wird. Das Registrierungskommando kann dann ohne Verzögerung an dieses gebunden werden.

Damit werden auch die Vorteile dieser Strategie klar: Da keine Verzögerung nötig ist, wird der Registrierungswechsel wieder unabhängig von der Schwankung der Tempodetektion, und ist damit während der Darbietung von der Musikerin bzw. dem Musiker über den Zeitpunkt, an dem die Finger von den Tasten genommen werden, direkt beeinflussbar. Die *Note-off*-Strategie hat jedoch auch einen Nachteil: Sie kann nicht für

Übergänge verwendet werden, an denen alle Töne liegen bleiben oder legato gespielt werden. In diesem Fall muss trotz ihrer Nachteile die *Delay*-Strategie benutzt werden.

Registrierungswechsel mit teilweisen Liegetönen

Werden jedoch ein oder mehrere Töne abgesetzt, während andere liegen bleiben, stellt dies kein Hindernis für die *Note-off*-Strategie dar, wie das folgende Beispiel demonstrieren soll (Abbildung 5.5 bzw. Listing 5.5). Dort findet zwischen der vierten und fünften Achtelnote des Taktes ein Registrierungswechsel statt, wobei das h über den Wechsel hinweg liegen bleibt, während die anderen Töne abgesetzt werden.

Um die Verwendung der *Note-off*-Strategie auch hier zu ermöglichen, kann das Event der vierten Achtelnote entsprechend der musikalischen Ausführung in zwei separate Events aufgespalten werden: eines, das z.B. drei Viertel der ursprünglichen Eventdauer lang ist und alle Akkordbestandteile enthält, sowie ein zweites, das die restliche Dauer einnimmt, und den Übergang repräsentiert, bei dem für kurze Zeit nur der liegen gebliebene Ton klingt.

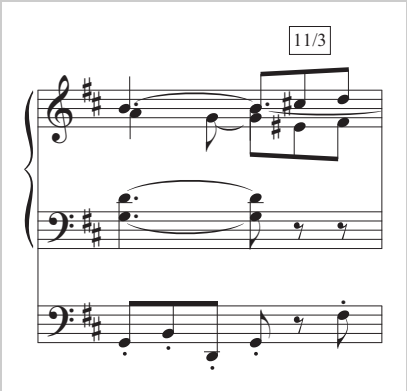


Abbildung 5.5: Notentext für Testsequenz E, Takt 12

```
1  CHORD (G2 G3 -D4 A4 B4) 1/3 measure12
2  CHORD (B2 -G3 -D4 -A4 -B4) 1/3
3  CHORD (D2 -G3 -D4 G4 -B4) 1/3
4  CHORD (G2 -G3 -D4 -G4 -B4) 3/12
5  NOTE -B4 1/12  // tied note
6    orgReg next
7  CHORD (E#4 -B4 C#5) 1/3
8  CHORD (F#3 F#4 -B4 D5) 1/3
```

Listing 5.5: Antescofo-Partitur für Testsequenz E, Takt 12

Wie in Listing 5.5 zu sehen ist, werden Liegetöne mit einem Minuszeichen vor der Tonhöhenangabe gekennzeichnet. Durch diese Aufteilung des ursprünglichen Events wird es Antescofo ermöglicht, den einzelnen, liegen bleibenden Ton zu detektieren, und das Registrierungskommando kann somit an dieses Zwischen-Event gebunden werden.

5.4 Einrichten des Laborsystems

Bevor die automatische Registrierungsumschaltung eingesetzt werden kann, ist in einem letzten Schritt das Laborsystem einzurichten. Da sich nach anfänglicher Untersuchung herausstellte, dass das Audiosignal der Orgel leider nicht als Input für den Score-Follower verwendet werden kann (vgl. Abschnitt 4.1), muss zunächst die MIDI-Kommunikation zwischen Orgel und Laborcomputer über ein MIDI-Interface hergestellt werden. Falls (wie beim im Rahmen dieser Arbeit verwendeten Aufbau) eine Orgel mit elektronischer Klangerzeugung in Zusammenarbeit mit einem kombinierten Audio-/MIDI-Interface benutzt wird, ist darauf zu achten, dass das Wiedergabesystem trotzdem mit dem Audiosignal der Orgel versorgt wird. Dies kann z.B. erreicht werden, indem der Ausgang des Interfaces mit dem Wiedergabesystem verbunden und das Audiosignal durchgeschleift wird.

Sofern die in Max/MSP programmierte Steuerungsoberfläche des Laborsystems nicht ohnehin standardmäßig so eingestellt ist, muss anschließend im *Input*-Modul MIDI als Eingangssignal für den Score-Follower sowie das verwendete MIDI-Interface als Input für alle weiteren, zu benutzenden Module der Oberfläche gewählt werden. Dies gilt insbesondere für das Modul der Orgelregistrierung, da sonst keine Kommunikation mit der Orgel und damit keine Setzersteuerung möglich ist.

Außerdem sollte die positive Auswirkung des Varianzparameters auf die Detektionslatenz des Score-Followers ausgenutzt werden, wobei sich ein Wert von 0.1 als empfehlenswert erwies. Da dieser Parameter jedoch in der aktuellen Antescofo-Version nicht mehr von Max/MSP aus gesetzt werden kann, muss dies mit dem Befehl `VARIANCE 0.1` am Beginn der Antescofo-Partitur erledigt werden. Abbildung 5.6 bzw. Listing 5.6 zeigen den Beginn von Testsequenz E mit eingetragenem Varianzkommando.

5.5 Verwendung des Systems

Damit die automatische Registrierungsumschaltung während einer Darbietung ihre Arbeit tun kann, muss zunächst die vorbereitete Partitur in Antescofo geladen werden.

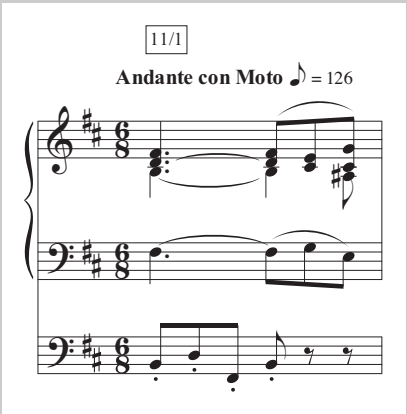


Abbildung 5.6: Notentext für Testsequenz E, Takt 1

```
1  VARIANCE 0.1
2
3  BPM 42
4    orgReg memory 11 1
5
6  CHORD (B2 F#3 B3 D4 F#4) 1/3 measure1
7  CHORD (D3 -F#3 -B3 -D4 -F#4) 1/3
8  CHORD (F#2 -F#3 -B3 -D4 -F#4) 1/3
9  CHORD (B2 -F#3 -B3 -D4 -F#4) 1/3
10 CHORD (G3 -B3 C#4 E4) 1/3
11 CHORD (E3 A#3 -C#4 G4) 1/3
```

Listing 5.6: Antescofo-Partitur für Testsequenz E, Takt 1 inkl. Varianz-Kommando

Anschließend wird der Score-Follower vor Beginn des Stücks aktiviert und gestartet bzw. muss in Übungsphasen bei jeder Wiederholung zurückgesetzt werden. Dafür sind über das Kontrollmodul der entwickelten Laborsystemoberfläche (siehe Abbildung 3.2) die wichtigsten Navigationsmöglichkeiten, wie etwa das Starten und Stoppen des Score-Followers oder Sprünge zu in der Partitur markierten Positionen, direkt zugänglich.

Solche Sprünge sind praktisch, um z.B. beim Üben an einer bestimmten Stelle beginnen zu können. Wird das Programm AscoGraph verwendet, um eine MusicXML-Datei zu importieren und daraus eine Antescofo-Partitur zu erstellen, so fügt es standardmäßig an jedem Taktbeginn Sprungmarken mit einer aus dem Wort »measure« und der Taktnummer zusammengesetzten Bezeichnung hinzu, indem es an das jeweils erste Event im Takt diese Bezeichnung anhängt (siehe z.B. Listing A.3). Mit derselben Vorgangsweise können auch manuell beliebige Events wie etwa Auftakte zu Sprungmarken gemacht werden, die dann nach dem Laden der Partitur über das Dropdown-Feld des Kontrollmoduls in der Laborsystemoberfläche als Startpunkt der Darbietung ausgewählt werden können.

Das Ausgabemodul zeigt alle nötigen Ausgabewerte von Antescofo, um die Funktion des Systems mitverfolgen und überwachen zu können. Handelt es sich beim verwendeten Laborcomputer um ein Gerät mit Macintosh-Betriebssystem, so kann auch das Zusatzprogramm AscoGraph (siehe Abbildung 5.7) zur grafischen Darstellung der Partitur (ähnlich jener eines MIDI-Sequencers) und der aktuell detektierten Position verwendet

werden. Diese wird in der Sequencer-Ansicht (links) als schwarze, vertikale Linie bzw. in der Code-Ansicht (rechts) als grüne Markierung des entsprechenden Event-Befehls angezeigt.

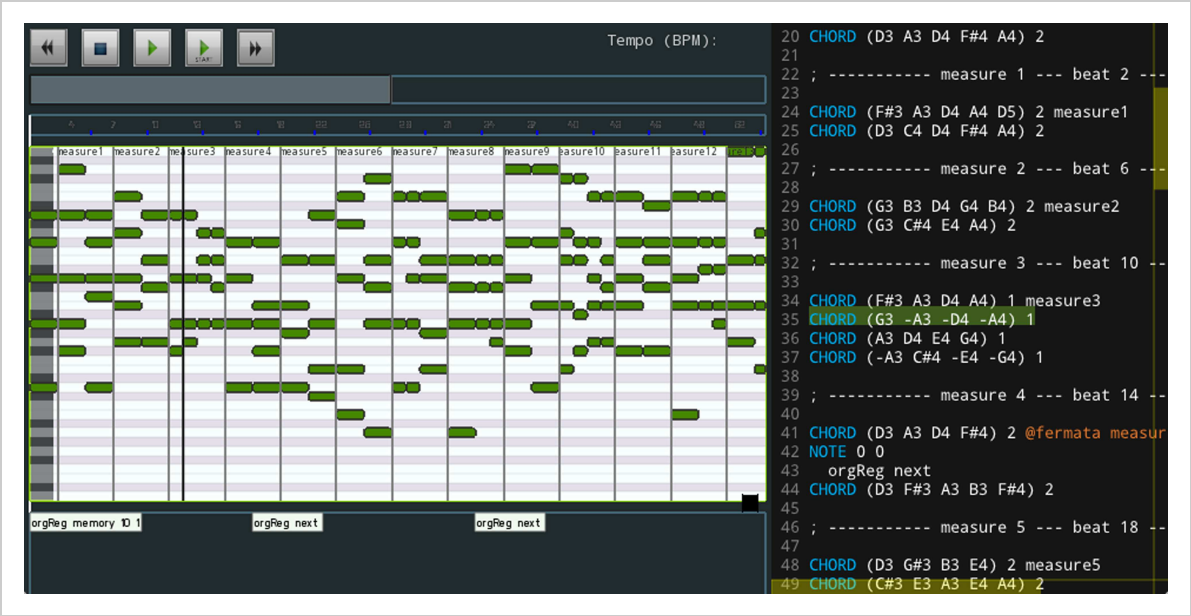


Abbildung 5.7: Oberfläche des Zusatzprogramms AscoGraph

6 Zusammenfassung und Ausblick

Ziel der vorliegenden Arbeit war es, ein Laborsystem zur automatischen Steuerung von Orgelsetzern zu erstellen, das in der Lage ist, die Partitur eines Stückes während dessen Darbietung mitzuverfolgen und an zuvor eingetragenen Stellen automatisch Registrierungswechsel auszulösen.

Zu diesem Zweck wurde das Feld des Score-Following unter Betrachtung von dessen Geschichte sowie aktueller Entwicklungen auf diesem Gebiet erschlossen (Kapitel 1). Dabei wurde das seit dem Jahr 2007 ständig weiterentwickelte Softwarepaket Antescofo aufgrund seines umfangreichen und direkt einsetzbaren Entwicklungsstandes als Score-Follower-Komponente des erarbeiteten Laborsystems ausgewählt. Da das Softwarepaket in Verbindung mit dessen grafischer Anzeige AscoGraph primär auf die Verwendung mit Max/MSP ausgelegt ist, war damit zugleich auch die Wahl der Plattform für das Laborsystem entschieden.

Um zu einem besseren Verständnis der grundsätzlichen Möglichkeiten von Antescofo im Rahmen der zu entwickelnden Setzersteuerung zu gelangen, wurden sowohl dessen Systemarchitektur und die begleitende Partitursprache untersucht als auch der Umgang mit dem Softwarepaket bezüglich Partiturerstellung, Konfigurations- und Steuerungsmöglichkeiten diskutiert (Kapitel 2). Dabei stellte sich heraus, dass für die Zusammenarbeit des Score-Followers mit dem umgebenden Laborsystem bereits die einfachste angebotene Interaktionsmöglichkeit über Max/MSP-Nachrichten ausreichend ist.

Darauf aufbauend wurde das Laborsystem aus aufeinander abgestimmten, bestehenden Hardware- und Software-Komponenten zusammengestellt, sowie zusätzliche Programmteile, wie etwa ein Modul zur Steuerung der Orgel, entwickelt, um deren Kommunikation untereinander herzustellen (Kapitel 3). Dabei wurde dieses System so gestaltet, dass es die notwendigen Funktionen für alle drei Szenarien bot, in denen es eingesetzt wurde: bei der Aufnahme von Audio- und MIDI-Material für die durchzuführenden Versuche,

als Testumgebung in der Versuchsphase selbst, sowie schlussendlich bei der Anwendung im Rahmen von Darbietungen.

Am Beginn des Entwicklungsprozesses stellte sich entgegen der ursprünglichen Konzeption heraus, dass das Audiosignal von Orgeln zumindest derzeit als Input für Antescofo aufgrund der zusätzlichen Klangkomponenten durch die Orgelregister nicht einsetzbar ist (Abschnitt 4.1). Deshalb wurde das System auf MIDI-Signale ausgelegt, da diese nur die angeschlagenen Tasten und keine Registerklänge enthalten. Weiters wurde ein informeller Hörtest durchgeführt, der den Toleranzbereich für den Auslösezeitpunkt der Registrierungsumschaltung in Bezug auf wahrgenommene Artefakte mit einer Größe von ca. 30 ms ermittelte (Abschnitt 4.2). Außerdem wurden die zeitlichen Größen des Systems bestimmt (Abschnitt 4.3), wobei gezeigt werden konnte, dass die Detektionslatenz von Antescofo mit Hilfe des Varianzparameters deutlich positiv beeinflusst werden kann.

Ein erster Testlauf offenbarte jedoch, dass die Schwankungsbreite der durch den Score-Follower ausgelösten Wechselzeitpunkte den ermittelten Toleranzbereich deutlich überstieg (Abschnitt 4.4). Dies war auf die für die geplante Anwendung unzureichende Reproduzierbarkeit der Temposchätzungen zurückzuführen, die durch das Funktionsprinzip von Antescofo bedingt ist. Also wurden mögliche Einflüsse auf die Systemperformance untersucht (Abschnitt 4.5): Die Variation von Parametern des Score-Followers brachte in diesem Zusammenhang keine Verbesserung, die Betrachtung von Auswirkungen des musikalischen Materials selbst zeigten jedoch, dass Stücke mit niedrigerem Tempo und zugleich zeitlich dichterem Notenmaterial bessere Ergebnisse lieferten. Deshalb wurde für Stücke, auf die diese Eigenschaften nicht zutreffen, ein Recodierungsverfahren vorgestellt, das zur Verbesserung der Performance angewendet werden konnte (Abschnitt 4.5.3).

Weiters wurden auch die von Antescofo zur Verfügung gestellten sogenannten Synchronisationsstrategien hinsichtlich ihres Einflusses auf die Reproduzierbarkeit der Temposchätzungen untersucht (Abschnitt 4.5.4). Entsprechende Tests ergaben jedoch, dass die alternativen Strategien ebenfalls keine Verbesserung gegenüber der standardmäßig eingesetzten *Delay*-Strategie mit sich brachten. Allerdings wurde eine zusätzliche, manuell anzuwendende *Note-off*-Strategie vorgeschlagen, die es trotz Antescofos ausschließlicher Detektion von Klanganfängen ermöglicht, Registrierungswechsel – entsprechend

der gängigen Registrierungspraxis – auch beim Loslassen der Tasten und dem damit einhergehenden Ende eines Klanges vorzunehmen.

In einem vollständigen Testlauf, der auch Pate für die Verwendungsdokumentation des Systems in einem prototypischen Workflow (Kapitel 5) stand, konnte abschließend gezeigt werden, dass der ermittelte Toleranzbereich mit den vorgeschlagenen Verbesserungsmaßnahmen laut statistischen Auswertungen der Versuche in rund 95 % (*Delay*-Strategie) bzw. nahezu 100 % der Fälle (*Note-off*-Strategie) eingehalten werden kann (Abschnitt 4.6).

Ausblick

Obwohl mit Hilfe der untersuchten Einflussmöglichkeiten eine akzeptable Systemperformance des Prototyps erreicht werden konnte, lässt sich nicht ausschließen, dass einzelne Registrierungswechsel außerhalb des ermittelten Toleranzbereichs liegen. Deshalb wäre für eine noch zufriedenstellendere Leistung bei zukünftigen Weiterentwicklungen in jedem Fall nach weiteren Möglichkeiten zur Verbesserung der Performance zu suchen.

Eine mögliche Richtung, in der diese Thematik weiterverfolgt werden könnte, zeigte sich, als für informelle Zwischentests kurze Zeit ein Gerät benutzt werden musste, das zwei Jahre älter als die beiden sonst verwendeten Laborcomputer war. Es fiel dabei auf, dass die darauf ausgeführten Versuche sichtbar schlechtere Ergebnisse lieferten. Im Umkehrschluss wäre also zu untersuchen, ob ein neueres bzw. besseres Modell auch die Leistungswerte signifikant begünstigen könnte.

Eine andere Idee für eine mögliche Steigerung der Systemleistung kam in einem Gespräch mit Antescofo-Projektleiter Arshia Cont auf. Er erwähnte, dass das Scheduling in Max/MSP, vor allem bei Einsatz vieler grafischer Elemente, eventuell für Leistungseinbußen verantwortlich sein könnte. Er vermutete weiter, dass mit der Verwendung von Pure Data statt Max/MSP als Plattform eventuell eine Steigerung erzielbar wäre. Um auf Pure Data umzusatteln, müsste allerdings auf AscoGraph, das Programm zur grafischen Anzeige der Partitur und aktuellen Partiturposition, verzichtet werden, da dieses nur auf MacOS läuft, eine Antescofo-Variante für Pure Data jedoch nur für Linux existiert.

Da die Versuche im Rahmen dieser Arbeit nur mit einer begrenzten Anzahl von musikalischen Beispielfällen durchgeführt werden konnten, wäre es interessant, das Systemverhalten auch eingehender im Praxiseinsatz mit weiterer Orgelliteratur unterschiedlicher Epochen und Stile sowie mit verschiedenen Registrierungen und Registrierungsweisen zu studieren, um daraus einerseits weitere potentielle Optimierungsstrategien ableiten sowie andererseits die Ergebnisse dieser Arbeit vor einem größeren Hintergrund von untersuchten Anwendungsbereichen betrachten und einordnen zu können.

Ein zukünftiges, für die Praxis entwickeltes Produkt sollte außerdem hinsichtlich der Usability verbessert werden, denn die teilweise empirische Bestimmung der im Verwendungsfall der *Delay*-Strategie notwendigen Verzögerungen, und damit die Einstellung der Auslösezeitpunkte für Registrierungswechsel, ist in der derzeitigen Version sehr mühsam. Deutlich angenehmer wäre es, wenn die Basiswerte für die Delays automatisch durch Analyse einer zuvor erstellten Trainingsversion des Stücks mit händisch ausgelösten Registrierungswechseln ermittelt und nachfolgende empirische Anpassungen mit einer Art grafischem Interface unterstützt werden könnten. In einem fortgeschrittenen Stadium des Systems wäre auch die Integration von Interface-Möglichkeiten in den Orgelspieltisch denkbar, um den Umgang mit dem System für Organistinnen und Organisten zu vereinfachen. In Frage kämen beispielsweise Tasten am Spieltisch, um die Startposition für den Score-Follower zu setzen, oder sie könnte direkt an eine manuelle Auswahl der Setzerposition vor dem Beginn der Darbietung gekoppelt werden. Außerdem könnten spontane Eingriffsmöglichkeiten während einer Darbietung hilfreich sein, etwa um zu früh ausgelöste Registrierungswechsel rückgängig zu machen oder fehlende manuell zu aktivieren, wobei das System im Optimalfall sogar auf solche Eingriffe reagieren und den nachfolgenden Ablauf entsprechend anpassen würde.

In einer zukünftigen Version sind auch durchaus Erweiterungen des Systems denkbar, um zusätzliche Möglichkeiten zu eröffnen. Es könnten beispielsweise die Schweller angesteuert werden, um die Lautstärke auch an komplizierten Stellen verändern zu können, oder es sind im Fall von Orgeln mit elektronischer Klangerzeugung sogar vom Stückverlauf abhängige Spatialisierungseffekte vorstellbar.

A Anhang

A.1 Verwendete Testsequenzen

Nachfolgend sind die Notentexte der verwendeten Testsequenzen abgebildet. Positionen für Registrierungswechsel sind darin mit Markierungen in kleinen Rechtecken über dem jeweils ersten Akkord eines neuen Registrierungsabschnitts gekennzeichnet.

Bei jenen Sequenzen, für die auch die entsprechenden Antescofo-Partituren generiert wurden, sind diese ebenfalls abgedruckt. Zur besseren Lesbarkeit sind in diesen Listings die einzelnen Takte durch Leerzeilen getrennt. Das jeweils erste Event eines Taktes ist außerdem mit eine Label markiert, das sich aus dem Wort »measure« (engl. Takt, aus Platzgründen an vereinzelten Stellen abgekürzt) und der Taktnummer zusammensetzt.

Testsequenz A

F. Mendelssohn-Bartholdy, op.65, No.5, 2. Satz, Takte 11 bis 14 mit Auftakt

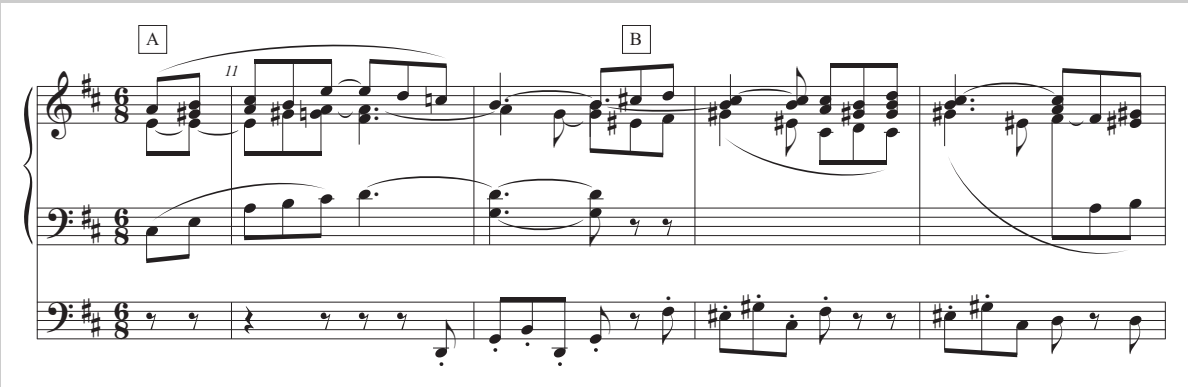


Abbildung A.1: Notentext der Testsequenz A

Testsequenz B

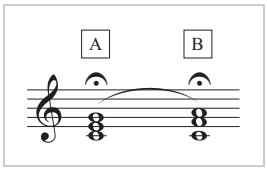


Abbildung A.2: Notentext der Testsequenz B

Testsequenz C

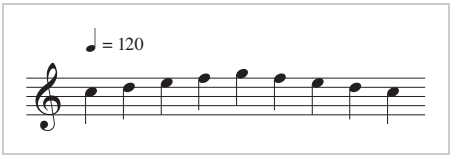


Abbildung A.3: Notentext der Testsequenz C

```
1  VARIANCE 0.1
2
3  BPM 120
4
5  NOTE C5 1
6  NOTE D5 1
7  NOTE E5 1
8  NOTE F5 1
9  NOTE G5 1
10 NOTE F5 1
11 NOTE E5 1
12 NOTE D5 1
13 NOTE C5 1
```

Listing A.1: Antescofo-Partitur der Testsequenz C

Testsequenz D

F. Mendelssohn-Bartholdy, op.65, No.5, 1. Satz

(Aufnahme mit Organist Stjepan Molnar)

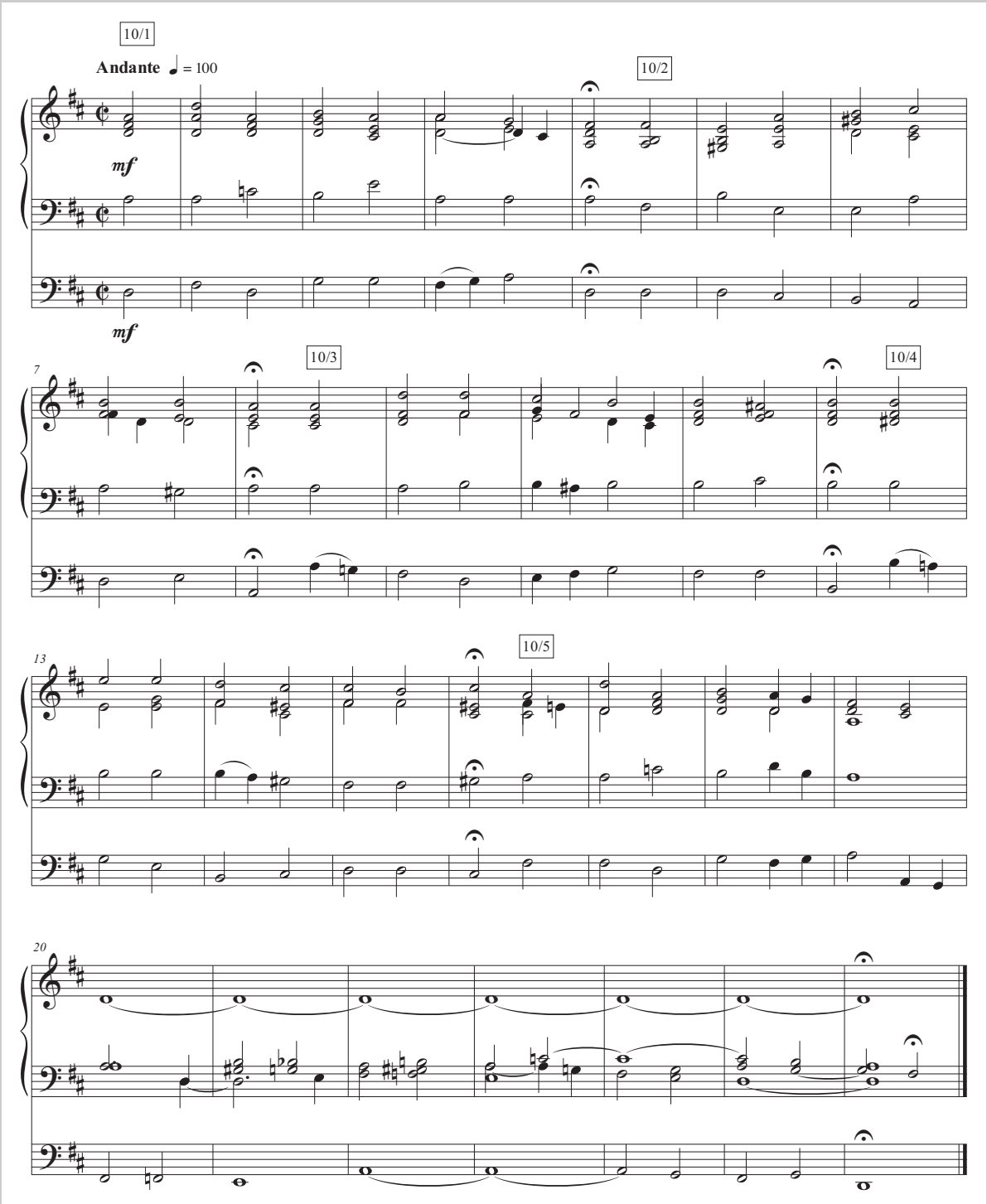


Abbildung A.4: Notentext der Testsequenz D

```
VARIANCE 0.1

BPM 100
  orgReg memory 10 1

CHORD (D3 A3 D4 F#4 A4) 2 pickupmeasure

CHORD (F#3 A3 D4 A4 D5) 2 measure1
CHORD (D3 C4 D4 F#4 A4) 2

CHORD (G3 B3 D4 G4 B4) 2 measure2
CHORD (G3 C#4 E4 A4) 2

CHORD (F#3 A3 D4 A4) 1 measure3
CHORD (G3 -A3 -D4 -A4) 1
CHORD (A3 D4 E4 G4) 1
CHORD (-A3 C#4 -E4 -G4) 1

CHORD (D3 A3 D4 F#4) 2 @fermata measr4
NOTE 0 0
  orgReg next
CHORD (D3 F#3 A3 B3 F#4) 2

CHORD (D3 G#3 B3 E4) 2 measure5
CHORD (C#3 E3 A3 E4 A4) 2

CHORD (B2 E3 D4 G#4 B4) 2 measure6
CHORD (A2 A3 C#4 E4 C#5) 2

CHORD (D3 A3 F#4 B4) 1 measure7
CHORD (-D3 -A3 D4 -F#4 -B4) 1
CHORD (E3 G#3 D4 E4 B4) 2

CHORD (A2 A3 C#4 E4 A4) 2 @fermata m8
NOTE 0 0
  orgReg next
CHORD (A3 C#4 E4 A4) 1
CHORD (G3 -A3 -C#4 -E4 -A4) 1

CHORD (F#3 A3 D4 F#4 D5) 2 measure9
CHORD (D3 B3 F#4 D5) 2

CHORD (E3 B3 E4 G4 C#5) 1 measure10
CHORD (F#3 A#3 -E4 F#4 -C#5) 1
CHORD (G3 B3 D4 -F#4 B4) 1
CHORD (-G3 -B3 C#4 E4 -B4) 1

CHORD (F#3 B3 D4 F#4 B4) 2 measure11
CHORD (F#3 C#4 E4 F#4 A#4) 2

CHORD (B2 B3 D4 F#4 B4) 2 @fermata m12
NOTE 0 0
  orgReg next
CHORD (B3 D#4 F#4 B4) 1
CHORD (A3 -B3 -D#4 -F#4 -B4) 1

CHORD (G3 B3 E4 E5) 2 measure13
CHORD (E3 B3 E4 G4 E5) 2

CHORD (B2 B3 F#4 D5) 1 measure14
CHORD (-B2 A3 -F#4 -D5) 1
CHORD (C#3 G#3 C#4 E#4 C#5) 2

CHORD (D3 F#3 F#4 C#5) 2 measure15
CHORD (D3 F#3 F#4 B4) 2

CHORD (C#3 G#3 C#4 E#4 C#5) 2 @fermata m16
NOTE 0 0
  orgReg next
CHORD (F#3 A3 C#4 F#4 A4) 1
CHORD (-F#3 -A3 -C#4 E4 -A4) 1

CHORD (F#3 A3 D4 D5) 2 measure17
CHORD (D3 C4 D4 F#4 A4) 2

CHORD (G3 B3 D4 G4 B4) 2 measure18
CHORD (F#3 D4 A4) 1
CHORD (G3 B3 -D4 G4) 1

CHORD (A3 D4 F#4) 2 measure19
CHORD (A2 -A3 C#4 E4) 1
CHORD (G2 -A3 -C#4 -E4) 1

CHORD (F#2 A3 D4) 2 measure20
CHORD (F2 -A3 -D4) 1
CHORD (-F2 D3 -A3 -D4) 1

CHORD (E2 -D3 G#3 B3 -D4) 2 measure21
CHORD (-E2 -D3 G3 Bb3 -D4) 1
CHORD (-E2 E3 -G3 -Bb3 -D4) 1

CHORD (A2 F#3 A3 -D4) 2 measure22
CHORD (-A2 F3 G#3 B3 -D4) 2

CHORD (-A2 E3 A3 -D4) 2 measure23
CHORD (-A2 -E3 -A3 C4 -D4) 1
CHORD (-A2 -E3 G3 -C4 -D4) 1

CHORD (-A2 F#3 -C4 -D4) 2 measure24
CHORD (G2 E3 G3 -C4 -D4) 2

CHORD (F#2 D3 A3 -C4 -D4) 2 measure25
CHORD (G2 -D3 G3 B3 -D4) 2

CHORD (D2 -D3 -G3 A3 -D4) 2 @fermata m26
CHORD (-D2 -D3 F#3 -A3 -D4) 2 @fermata
```

Listing A.2: Antescofo-Partitur der Testsequenz D

Testsequenz E

F. Mendelssohn-Bartholdy, op.65, No.5, 2. Satz, Takte 1 bis 12

(Aufnahme mit Organist Stjepan Molnar)

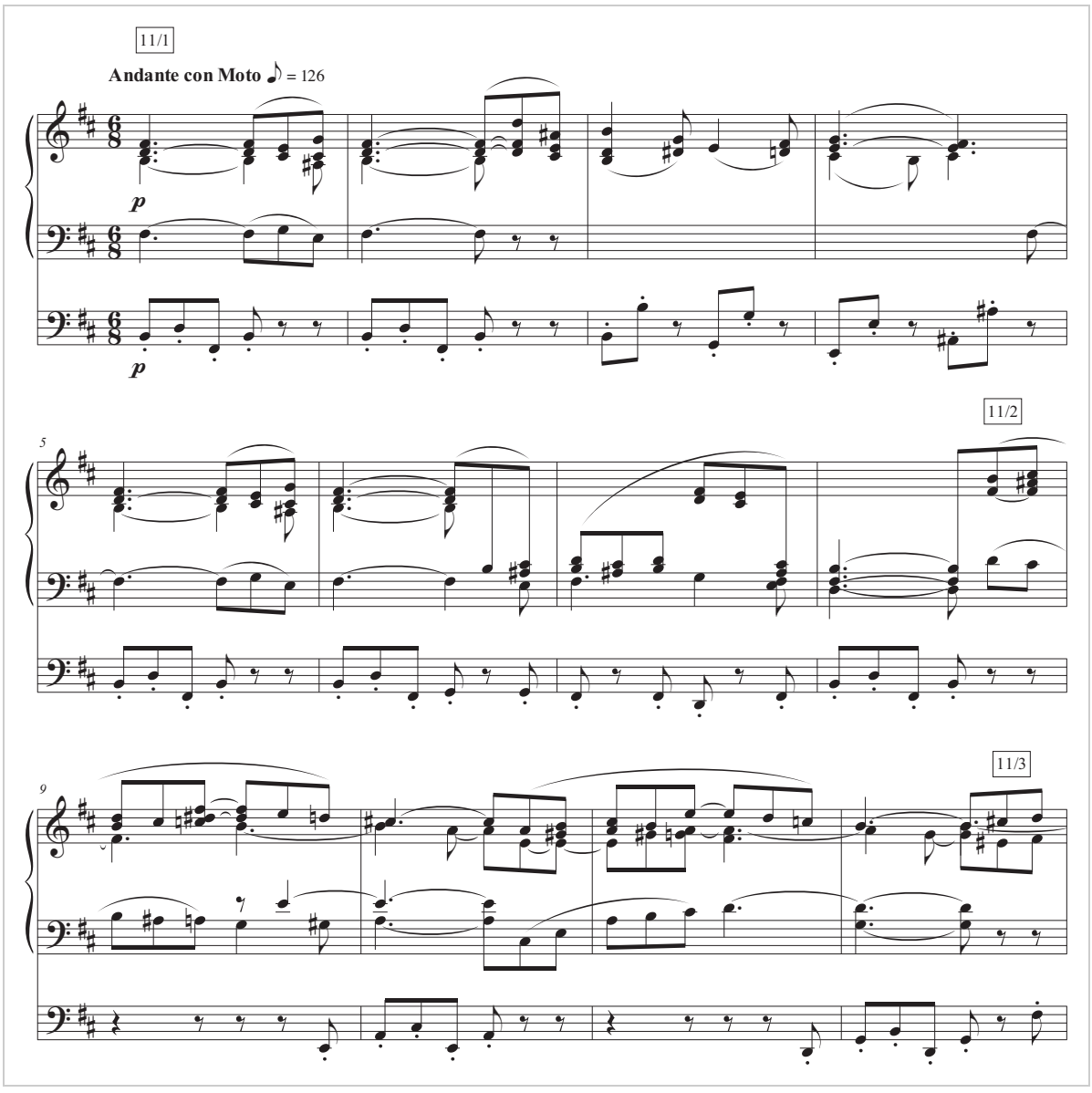


Abbildung A.5: Notentext der Testsequenz E

```
VARIANCE 0.1

BPM 42 ;(126/3)
  orgReg memory 11 1

CHORD (B2 F#3 B3 D4 F#4) 1/3 measure1
CHORD (D3 -F#3 -B3 -D4 -F#4) 1/3
CHORD (F#2 -F#3 -B3 -D4 -F#4) 1/3
CHORD (B2 -F#3 -B3 -D4 -F#4) 1/3
CHORD (G3 -B3 C#4 E4) 1/3
CHORD (E3 A#3 -C#4 G4) 1/3

CHORD (B2 F#3 B3 D4 F#4) 1/3 measure2
CHORD (D3 -F#3 -B3 -D4 -F#4) 1/3
CHORD (F#2 -F#3 -B3 -D4 -F#4) 1/3
CHORD (B2 -F#3 -B3 -D4 -F#4) 1/3
CHORD (-D4 -F#4 D5) 1/3
CHORD (C#4 E4 A#4) 1/3

CHORD (B2 B3 D4 B4) 1/3 measure3
CHORD (-B3 -D4 -B4) 1/3
CHORD (D#4 G4) 1/3
CHORD (G2 E4) 1/3
CHORD (G3 -E4) 1/3
CHORD (D4 F#4) 1/3

CHORD (E2 C#4 E4 G4) 1/3 measure4
CHORD (E3 -C#4 -E4 -G4) 1/3
CHORD (B3 -E4 -G4) 1/3
CHORD (A#2 C#4 E4 F#4) 1/3
CHORD (A#3 -C#4 -E4 -F#4) 1/3
CHORD (F#3 -C#4 -E4 -F#4) 1/3

CHORD (B2 -F#3 B3 D4 F#4) 1/3 measure5
CHORD (D3 -F#3 -B3 -D4 -F#4) 1/3
CHORD (F#2 -F#3 -B3 -D4 -F#4) 1/3
CHORD (B2 -F#3 -B3 -D4 -F#4) 1/3
CHORD (G3 -B3 C#4 E4) 1/3
CHORD (E3 A#3 -C#4 G4) 1/3

CHORD (B2 F#3 B3 D4 F#4) 1/3 measure6
CHORD (D3 -F#3 -B3 -D4 -F#4) 1/3
CHORD (F#2 -F#3 -B3 -D4 -F#4) 1/3
CHORD (G2 F#3 -B3 -D4 -F#4) 1/3
CHORD (-F#3 B3) 1/3
CHORD (G2 E3 A#3 C#4) 1/3

CHORD (F#2 F#3 B3 D4) 1/3 measure7
CHORD (-F#3 A#3 C#4) 1/3
CHORD (F#2 -F#3 B3 D4) 1/3
CHORD (D2 G3 D4 F#4) 1/3
CHORD (-G3 C#4 E4) 1/3
CHORD (F#2 E3 F#3 A#3 C#4) 1/3

CHORD (B2 D3 F#3 B3) 1/3 measure8
CHORD (D3 -D3 -F#3 -B3) 1/3
CHORD (F#2 -D3 -F#3 -B3) 1/3
CHORD (B2 -D3 -F#3 -B3) 1/3
NOTE 0 0
  orgReg next
CHORD (D4 F#4 B4) 1/3
CHORD (C#4 -F#4 A#4 C#5) 1/3

CHORD (B3 F#4 B4 D5) 1/3 measure9
CHORD (A#3 -F#4 C#5) 1/3
CHORD (A3 -F#4 C5 D#5 F#5) 1/3
CHORD (G3 B4 -D#5 -F#5) 1/3
CHORD (-G3 E4 -B4 E5) 1/3
CHORD (E2 G#3 -E4 -B4 D5) 1/3

CHORD (A2 A3 E4 -B4 C#5) 1/3 measure10
CHORD (C#3 -A3 -E4 -B4 -C#5) 1/3
CHORD (E2 -A3 -E4 A4 -C#5) 1/3
CHORD (A2 -A3 -E4 -A4 C#5) 1/3
CHORD (C#3 E4 A4) 1/3
CHORD (E3 -E4 G#4 B4) 1/3

CHORD (A3 -E4 A4 C#5) 1/3 measure11
CHORD (B3 G#4 B4) 1/3
CHORD (C#4 G4 A4 E5) 1/3
CHORD (D4 F#4 A4 -E5) 1/3
CHORD (-D4 -F#4 -A4 D5) 1/3
CHORD (D2 -D4 -F#4 -A4 C5) 1/3

CHORD (G2 G3 -D4 A4 B4) 1/3 measure12
CHORD (B2 -G3 -D4 -A4 -B4) 1/3
CHORD (D2 -G3 -D4 G4 -B4) 1/3
CHORD (G2 -G3 -D4 -G4 -B4) 3/12
NOTE -B4 1/12
  orgReg next
CHORD (E#4 -B4 C#5) 1/3
CHORD (F#3 F#4 -B4 D5) 1/3
```

Listing A.3: Antescofo-Partitur der Testsequenz E

Testsequenz F

J. S. Bach, BWV 525, No.1, 1. Satz, Takte 1 bis 5

(Aufnahme mit Organist Stjepan Molnar)

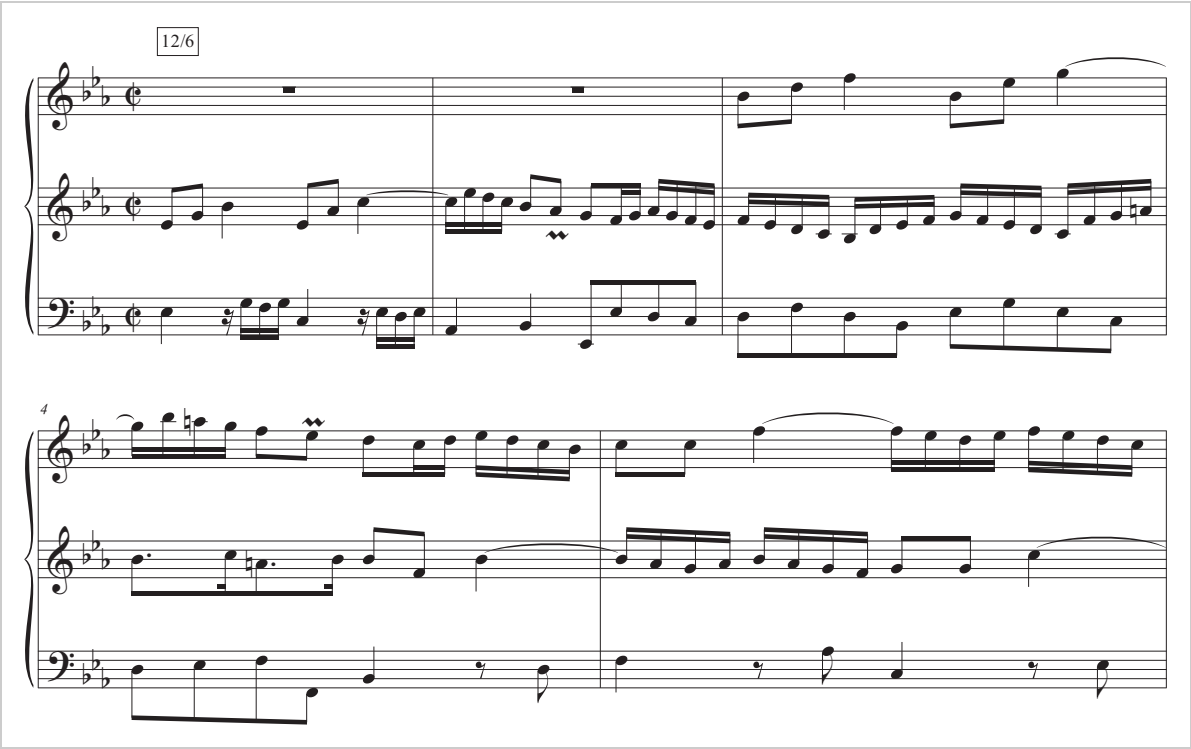


Abbildung A.6: Notentext der Testsequenz F

```
VARIANCE 0.1

BPM 75

CHORD (Eb3 Eb4) 1/2 measure1
CHORD (-Eb3 G4) 1/2
NOTE Bb4 1/4
CHORD (G3 -Bb4) 1/4
CHORD (F3 -Bb4) 1/4
CHORD (G3 -Bb4) 1/4
CHORD (C3 Eb4) 1/2
CHORD (-C3 Ab4) 1/2
NOTE C5 1/4
CHORD (Eb3 -C5) 1/4
CHORD (D3 -C5) 1/4
CHORD (Eb3 -C5) 1/4

CHORD (Ab2 -C5) 1/4 measure2
CHORD (-Ab2 Eb5) 1/4
CHORD (-Ab2 D5) 1/4
CHORD (-Ab2 C5) 1/4
CHORD (Bb2 Bb4) 1/2
TRILL ( (-Bb2 Ab4) (-Bb2 Bb4) ) 1/2
CHORD (Eb2 G4) 1/2
CHORD (Eb3 F4) 1/4
CHORD (-Eb3 G4) 1/4
CHORD (D3 Ab4) 1/4
CHORD (-D3 G4) 1/4
CHORD (C3 F4) 1/4
CHORD (-C3 Eb4) 1/4

CHORD (D3 F4 Bb4) 1/4 measure3
CHORD (-D3 Eb4 -Bb4) 1/4

CHORD (-F3 C4 -D5) 1/4
CHORD (D3 Bb3 F5) 1/4
CHORD (-D3 D4 -F5) 1/4
CHORD (Bb2 Eb4 -F5) 1/4
CHORD (-Bb2 F4 -F5) 1/4
CHORD (Eb3 G4 Bb4) 1/4
CHORD (-Eb3 F4 -Bb4) 1/4
CHORD (G3 Eb4 Eb5) 1/4
CHORD (-G3 D4 -Eb5) 1/4
CHORD (Eb3 C4 G5) 1/4
CHORD (-Eb3 F4 -G5) 1/4
CHORD (C3 G4 -G5) 1/4
CHORD (-C3 A4 -G5) 1/4

CHORD (D3 Bb4 -G5) 1/4 measure4
CHORD (-D3 -Bb4 Bb5) 1/4
CHORD (Eb3 -Bb4 A5) 1/4
CHORD (-Eb3 C5 G5) 1/4
CHORD (F3 A4 F5) 1/2
TRILL ( (F2 -A4 Eb5) (F2 -A4 F5) ) 1/4
TRILL ( (-F2 Bb4 Eb5) (-F2 Bb4 F5) ) 1/4
CHORD (Bb2 Bb4 D5) 1/2
CHORD (-Bb2 F4 C5) 1/4
CHORD (-Bb2 -F4 D5) 1/4
CHORD (Bb4 Eb5) 1/4
CHORD (-Bb4 D5) 1/4
CHORD (D3 -Bb4 C5) 1/4
CHORD (-D3 Bb4) 1/4

CHORD (F3 -Bb4 C5) 1/4 measure5
CHORD (-F3 Ab4 -C5) 1/4
CHORD (-F3 G4 C5) 1/4
CHORD (-F3 Ab4 -C5) 1/4
CHORD (Bb4 F5) 1/4
CHORD (Ab4 -F5) 1/4
CHORD (Ab3 G4 -F5) 1/4
CHORD (-Ab3 F4 -F5) 1/4
CHORD (C3 G4 -F5) 1/4
CHORD (-C3 -G4 Eb5) 1/4
CHORD (-C3 G4 D5) 1/4
CHORD (-C3 -G4 Eb5) 1/4
CHORD (C5 F5) 1/4
CHORD (-C5 Eb5) 1/4
CHORD (Eb3 -C5 D5) 1/4
CHORD (-Eb3 C5) 1/4
```

Listing A.4: Antescofo-Partitur der Testsequenz F

Testsequenz G

C. Franck, op.17, No.2, 1. Satz, Takte 1 bis 11

(Archivaufnahme mit Organistin Milica Debelnogić)

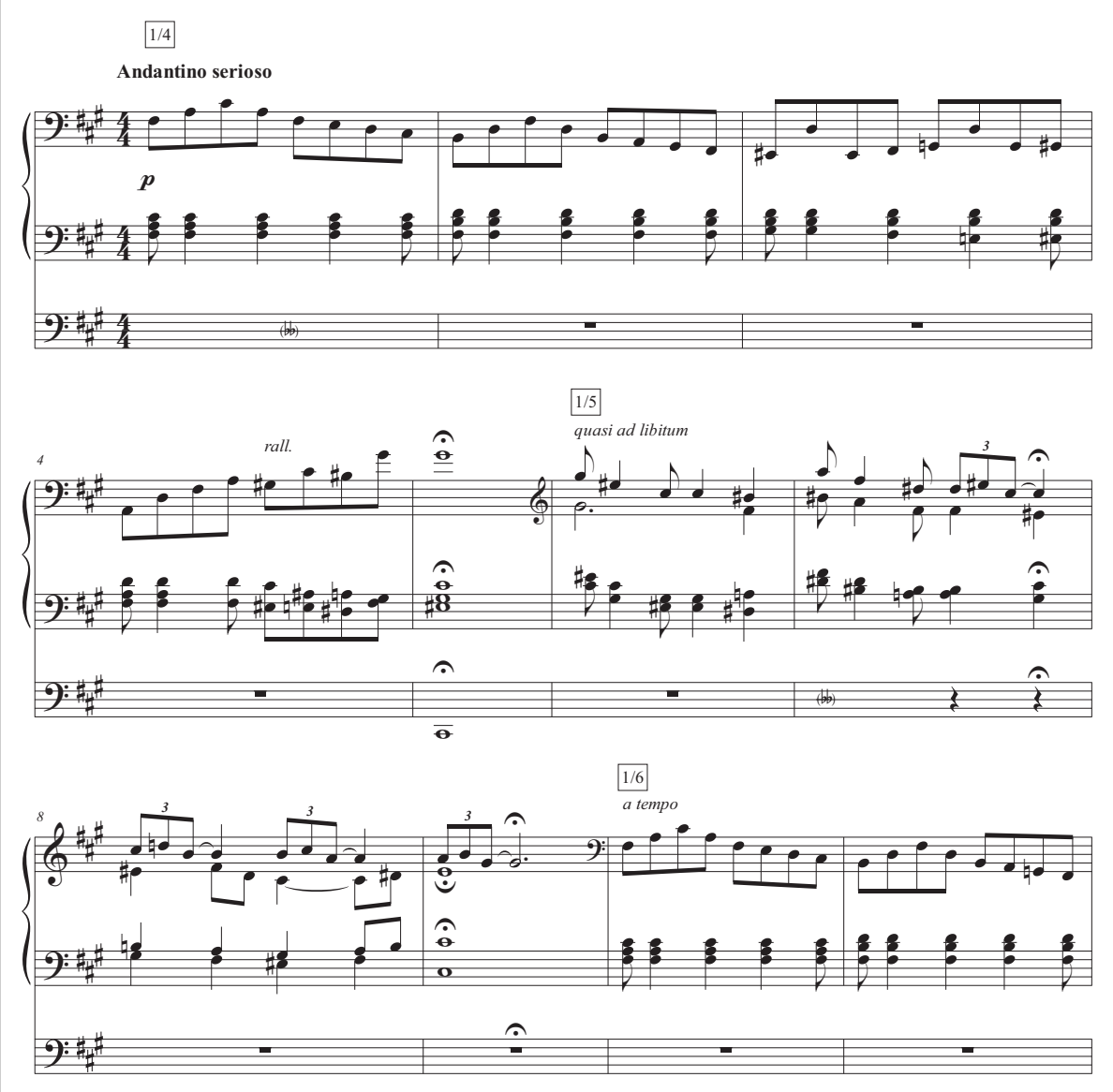


Abbildung A.7: Notentext der Testsequenz G

```
VARIANCE 0.1

BPM 90
  orgReg memory 10 4

CHORD (F#3 A3 C#4) 1/2 measure1
CHORD (F#3 A3 C#4) 1/2
CHORD (-F#3 -A3 C#4) 1/2
CHORD (F#3 A3 C#4) 1/2
CHORD (F#3 -A3 -C#4) 1/2
CHORD (E3 F#3 A3 C#4) 1/2
CHORD (D3 -F#3 -A3 -C#4) 1/2
CHORD (C#3 F#3 A3 C#4) 1/2

CHORD (B2 F#3 B3 D4) 1/2 measure2
CHORD (D3 F#3 B3 D4) 1/2
CHORD (F#3 -B3 -D4) 1/2
CHORD (D3 F#3 B3 D4) 1/2
CHORD (B2 -F#3 -B3 -D4) 1/2
CHORD (A2 F#3 B3 D4) 1/2
CHORD (G#2 -F#3 -B3 -D4) 1/2
CHORD (F#2 F#3 B3 D4) 1/2

CHORD (E#2 G#3 B3 D4) 1/2 measure3
CHORD (D3 G#3 B3 D4) 1/2
CHORD (E#2 -G#3 -B3 -D4) 1/2
CHORD (F#2 F#3 B3 D4) 1/2
CHORD (G2 -F#3 -B3 -D4) 1/2
CHORD (D3 E3 B3 D4) 1/2
CHORD (G2 -E3 -B3 -D4) 1/2
CHORD (G#2 E#3 B3 D4) 1/2

CHORD (A2 F#3 A3 D4) 1/2 measure4
CHORD (D3 F#3 A3 D4) 1/2
CHORD (F#3 -A3 -D4) 1/2
CHORD (F#3 A3 D4) 1/2
CHORD (E#3 G#3 C#4) 1/2
CHORD (E3 A#3 C#4) 1/2
CHORD (D#3 A3 B#3) 1/2
CHORD (F#3 G#3 G#4) 1/2


CHORD (E#3 G#3 C#4 G#4) 4 @fermata msr5
NOTE 0 0 //registration
  orgReg next

CHORD (C#4 E#4 G#4 G#5) 1/2 measure6
CHORD (G#3 C#4 -G#4 E#5) 1
CHORD (E#3 -G#4 C#5) 1/2
CHORD (E#3 G#3 -G#4 C#5) 1
CHORD (D#3 A3 F#4 B#4) 1

CHORD (D#4 F#4 B#4 A5) 1/2 measure7
CHORD (B#3 D#4 A4 F#5) 1
CHORD (A3 B#3 F#4 D#5) 1/2
CHORD (A3 B#3 F#4 D#5) 1/3
CHORD (-A3 -B#3 -F#4 E#5) 1/3
CHORD (-A3 -B#3 -F#4 C#5) 1/3
CHORD (G#3 C#4 E#4 -C#5) 1

CHORD (G#3 B3 E#4 C#5) 1/3 measure8
CHORD (-G#3 -B3 -E#4 D5) 1/3
CHORD (-G#3 -B3 -E#4 B4) 1/3
CHORD (F#3 A3 F#4 -B4) 1/2
CHORD (-F#3 -A3 D4 -B4) 1/2
CHORD (E#3 G#3 C#4 B4) 1/3
CHORD (-E#3 -G#3 -C#4 C#5) 1/3
CHORD (-E#3 -G#3 -C#4 A4) 1/3
CHORD (F#3 A3 -C#4 -A4) 1/2
CHORD (-F#3 B3 D#4 -A4) 1/2

CHORD (C#3 C#4 E#4 A4) 1/3 measure9
CHORD (-C#3 -C#4 -E#4 B4) 1/3
CHORD (-C#3 -C#4 -E#4 G#4) 1/3
CHORD (-C#3 -C#4 -E#4 -G#4) 3
NOTE 0 0
  orgReg next

CHORD (F#3 A3 C#4) 1/2 measure10
CHORD (F#3 A3 C#4) 1/2
CHORD (-F#3 -A3 C#4) 1/2
CHORD (F#3 A3 C#4) 1/2
CHORD (F#3 -A3 -C#4) 1/2
CHORD (E3 F#3 A3 C#4) 1/2
CHORD (D3 -F#3 -A3 -C#4) 1/2
CHORD (C#3 F#3 A3 C#4) 1/2 @fermata

CHORD (B2 F#3 B3 D4) 1/2 measure11
CHORD (D3 F#3 B3 D4) 1/2
CHORD (F#3 -B3 -D4) 1/2
CHORD (D3 F#3 B3 D4) 1/2
CHORD (B2 -F#3 -B3 -D4) 1/2
CHORD (A2 F#3 B3 D4) 1/2
CHORD (G2 -F#3 -B3 -D4) 1/2
CHORD (F#2 F#3 B3 D4) 1/2
```

Listing A.5: Antescofo-Partitur der Testsequenz G

A.2 Geräte- und Software-Verzeichnis

Verwendete Geräte

Orgel
Trillium Masterpiece 968
Hersteller: Rodgers
Seriennummer: Z7A2466
Firmware-Version: V-4003706-N16

MADI-Wandler
Andiamo 2
Hersteller: DirectOut
Seriennummer: 0501000000000000030009090B0A0202
Firmware-Version: 4.1

MIDI-Embedder
Exbox.Midicom
Hersteller: DirectOut
Seriennummer: 23566789

Audio-/MIDI-Interface
MADIface USB
Hersteller: RME
Seriennummer: 23628258
Firmware-Version: 20, Treiber-Version: 2.18

Laborcomputer
Mac mini (Late 2012)
Hersteller: Apple
Seriennummer: C07LG02SDY3J
Ausstattung: Intel Core i7-3720QM, 8 GB 1600 MHz DDR3 RAM
Betriebssystem: OS X 10.11.2

Laborcomputer (Orgelaufnahme/-tests)
Mac mini (Late 2012)
Hersteller: Apple
Seriennummer: C07M80LCDY3H
Ausstattung: Intel Core i7-3615QM, 8 GB 1600 MHz DDR3 RAM
Betriebssystem: OS X 10.9.5

Verwendete Software-Programme

Programmier- und Laufzeitumgebung
Max/MSP
Hersteller: Cycling '74
Version (Mac): 6.1.10 (26bd7fb) 32bit
Version (Win): 6.1.10 (26bd7fb) 64bit

Score-Follower
Antescofo
Hersteller: MuTant Team Project
Version (Mac): 0.92-46

Anzeige Score-Follower
AscoGraph
Hersteller: MuTant Team Project
Version (Mac): 0.2.9

DAW
Cubase Pro
Hersteller: Steinberg
Version (Mac): 8.0.0 (Build 397) 64bit
Version (Win): 8.0.35 (Build 565) 64bit

DAW (Orgel-Aufnahme)
Nuendo
Hersteller: Steinberg
Version (Mac): 6.0.7 (Build 2277) 64bit

Programmier- und Laufzeitumgebung (Mathematik)
Octave
Hersteller: GNU Project
Version (Win): 4.2.0

Notensatzprogramm
Finale
Hersteller: MakeMusic
Version (Win): 2014.5.6359

A.3 Textauszüge der Online-Quellen

Zur Dokumentation der reinen Online-Quellen sollen die verwendeten Passagen der entsprechenden Webseiten hier im Originalwortlaut festgehalten werden.

Antescofo [Ant]

Antescofo~ for Max and PureData is a modular polyphonic Score Following system as well as a Synchronous Programming language for real-time computer music composition and live performance. The module allows for automatic recognition of music score position and tempo from a realtime audio Stream coming from performer(s), making it possible to synchronize an instrumental performance with computer realized elements. The synchronous language within Antescofo allows flexible writing of time and interaction in computer music in conjuction with its dedicated graphical editor AscoGraph.

http://forumnet.ircam.fr/product/antescofo-en (besucht am 27. 10. 2016)

AscoGraph [Asc]

Main features

- automatic conversion from MusicXML or MIDI scores to Antescofo Language
- automatic scrolling of notes view and editor view with Antescofo Score Follower (using OpenSoundControl)
- note selection from editor or visual, in both directions for fast and easy modifications : double click on a line in the text editor to see which note it is in the pianoroll,
- graphical editing of curves objects (break point functions)
- syntax coloring, group folding, etc.
- Template group (group, loop, curve, whenever, oscsend,. . .) creation.
- OSC automatic communication with Antescofo (start, play, stop, nextevent, reloadscore on save, etc. . .)
- simulation and graphical display of performance.

http://forumnet.ircam.fr/product/antescofo-en/ascograph-en (besucht am 27. 10. 2016)

Autoflip Sheet Music Viewer [Aut]

Autoflip is an advanced sheet music viewer that automatically turns the pages in sync with your performance by actually listening to how you are playing. It can account for variations in tempo as well as minor mistakes. Autoflip version 1.0 is optimized for solo performances using pianos and keyboards.

Autoflip works with any sheet music in PDF format. You can enable automatic turning for each new piece by running it through a simple training session. A training session is simply you playing the piece and turning the pages! After that, all you have to do is play and Autoflip will take care of the turning.

https://itunes.apple.com/us/app/autoflip-sheet-music-viewer/id413455877 (besucht am 27. 10. 2016)

Flowkey [Flo]

[Demo-Animation]

[...]

Entdecke Klavierlernen neu

flowkey zeigt dir die genaue Spieltechnik und erkennt automatisch, ob du richtig mitspielst. Dadurch lernst du die Noten und Akkorde einfacher und schneller, als je zuvor.

[...]

Alles, was du zum Klavier lernen brauchst

Songs entdecken Finde Songs aus allen Genres und Schwierigkeitsstufen
Warte-Modus Spiele Songs mit interaktivem Live-Feedback
Video und Noten Auditives und visuelles Lernen perfekt kombiniert
Zeitlupe Langsame und schnelle Übungen für jeden Song
Loop Funktion Wiederhole Songabschnitte, um sie gezielt zu üben
Handauswahl Lerne linke und rechte Hand getrennt, wie im Unterricht

http://www.flowkey.com (besucht am 27. 10. 2016)

IMuSE [Imu]

The Integrated Multimodal Score-following Environment (IMuSE) is a research project funded by the Social Sciences and Humanities Research Council (SSHRC) starting in 2009.

IMuSE is a software environment based on the NoteAbilityPro music editor and the IIMPE system developed by Keith Hamel and Bob Pritchard. With IMuSE, score-following is expanded from audio pitch-tracking to also include a variety of multi-channel gesture data. Live performance gestures can come from any kind of single or multi-channel continuous data from standard controllers, accelerometers, movement trackers, video trackers, live audio analysis etc. This data is matched against pre-recorded gesture data which has been stored in the NoteAbilityPro score as multi-channel break-point functions.

Polyphonic pitch-tracking is performed using the antescofo score-following software developed by Arshia Cont and gesture-tracking is performed using the gf software developed by Frederic Bevilacqua and Bruno Zamborlin. IMuSE negotiates between tracking data received from multiple sources and adjusts score playback based on an assessment of which data is most reliable at a given point in the performance.

[...]

In the performance below, the motion of the pianists hands are used both to synchronize the performance to the score and to generate electroacoustic events. The pianist (Megumi Masaki) plays from the score (which includes information about how the hands are moved above the keyboard) while a NoteAbilityPro score is running on a separate computer. The piano sounds are processed and a variety of other samples are generated during the performance. You can see the camera mounted directly above the piano keyboard.

http://debussy.music.ubc.ca/muset/imuse.html (besucht am 15. 02. 2017)

The One Smart Piano teaches you to play using an iPad [SmaB]

The first, The One Smart Piano, is an digital piano with 88 weighted keys; the other, The One Light Keyboard, is 61-key MIDI keyboard. Both keyboards integrate directly with an iPhone or an iPad [...] to teach you songs and present music lessons through a free app.

[...]

Both of The One's keyboards teach piano using light-up keys, which you can already find on existing MIDI keyboards and piano accessories. [...] But The One takes the concept a bit further. Rather than just running through songs, its keyboards are able to stop at each note and wait for you to play it correctly. They're also able to simultaneously highlight the note that you're supposed to play on the keyboard and on the iPad.

http://www.theverge.com/2015/6/29/8860927/the-one-smart-piano-and-keyboard-indiegogo (besucht am 27. 10. 2016)

Tonara Interactive Piano Sheet Music [Ton]

With Tonara's patented interactive scores you can now experience playing with sheet music in a completely effective and engaging way! Tonara scores will listen to you play, mark your position in the score, and turn the pages automatically for you exactly at the right time. Enjoy hundreds of free classical scores and songs in the Tonara 'Free Zone'. Get access to thousands of classical, rock, pop, soundtracks and other scores in the Tonara store. Use our music synchronization capabilities – your recordings automatically are synchronized to the score so that it's easy and effective to review your practice.

https://itunes.apple.com/us/app/tonara-interactive-piano-sheet/id454753605 (besucht am 27. 10. 2016)

Literaturverzeichnis

[Ace11] Jonathan Aceituno. *Real-time score following techniques*. 2011.

[AD90] Paul E. Allen und Roger B. Dannenberg. »Tracking Musical Beats in Real Time«. In: *Proceedings of the 1990 International Computer Music Conference*. 1990.

[Ant] *Antescofo*. URL: http://forumnet.ircam.fr/product/antescofo-en (besucht am 27. 10. 2016).

[AntHV] *Antescofo Hilfe-Patch, Subpatch ScoreLanguage, Subsubpatch Variance*. Max/MSP-Patch, ausgeliefert mit Antescofo-Version 0.92-46.

[AntOD] *Antescofo online documentation, Section Synchronization Strategies*. URL: http://support.ircam.fr/docs/Antescofo/manuals/Reference/time_synchro/index.html (besucht am 17. 01. 2017).

[AntTS] *The antescofo∼ tutorial series*. Max/MSP-Patches, ausgeliefert mit Antescofo-Version 0.92-46.

[Arz07] Andreas Arzt. »Score Following with Dynamic Time Warping – An Automatic Page-Turner«. Masterarbeit. Technische Universität Wien, 2007.

[Asc] *AscoGraph*. URL: http://forumnet.ircam.fr/product/antescofo-en/ascograph-en (besucht am 27. 10. 2016).

[Aut] *Autoflip Sheet Music Viewer*. URL: https://itunes.apple.com/us/app/autoflip-sheet-music-viewer/id413455877 (besucht am 27. 10. 2016).

[BD85] Joshua J. Bloch und Roger B. Dannenberg. »Real-Time Computer Accompaniment of Keyboard Performances«. In: *Proceedings of the 1985 International Computer Music Conference*. 1985.

[BG92] Gérard Berry und Georges Gonthier. »The Esterel synchronous programming language: design, semantics, implementation«. In: *Science of Computer Programming* 19.2 (1992), S. 87–152.

[BP66] Leonard E. Baum und Ted Petrie. »Statistical Inference for Probabilistic Functions of Finite State Markov Chains«. In: *The Annals of Mathematical Statistics* 37.6 (1966), S. 1554–1563.

[BW05] Mark A. Bartsch und Gregory H. Wakefield. »Audio thumbnailing of popular music using chroma-based representations«. In: *IEEE Transactions on Multimedia* 7.1 (2005), S. 96–104.

[CLB99] Pedro Cano, Alex Loscos und Jordi Bonada. »Score-Performance Matching Using HMMs«. In: *Proceedings of the 1999 International Computer Music Conference*. 1999.

[Con08] Arshia Cont. »Antescofo: Anticipatory Synchronization and control of Interactive parameters in Computer Music«. In: *Proceedings of the 2008 International Computer Music Conference*. 2008.

[Con10] Arshia Cont. »A Coupled Duration-Focused Architecture for Real-Time Music-to-Score Alignment«. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2010), S. 974–987.

[CRV+12] Julio Carabias-Orti, Francisco Rodríguez-Serrano, Pedro Vera-Candeas u. a. »A real-time NMF-based score follower for MIREX 2012«. In: *Music Information Retrieval Evaluation eXchange*. 2012.

[CSSR07] Arshia Cont, Diemo Schwarz, Norbert Schnell und Christopher Raphael. »Evaluation of Real-Time Audio-to-Score Alignment«. In: *International Symposium on Music Information Retrieval (ISMIR)*. 2007.

[Dan84] Roger B. Dannenberg. »An on-line algorithm for real-time accompaniment«. In: *Proceedings of the 1984 International Computer Music Conference*. 1984.

[DH03] Roger B. Dannenberg und Ning Hu. »Polyphonic Audio Matching for Score Following and Intelligent Audio Editors«. In: *Proceedings of the 2003 International Computer Music Conference*. 2003.

[DHH97] Peter Desain, Henkjan Honing und Hank Heijink. »Robust Score-Performance Matching: Taking Advantage of Structural Information«. In: *Proceedings of the 1997 International Computer Music Conference*. 1997.

[Dix05] Simon Dixon. »An On-Line Time Warping Algorithm for Tracking Musical Performances«. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. 2005, S. 1727–1728.

[DM87] Roger B. Dannenberg und Bernard Mont-Reynaud. »Following an Improvisation in Real Time«. In: *Proceedings of the 1987 International Computer Music Conference*. 1987.

[DM88] Roger B. Dannenberg und Hirofumi Mukaino. »New Techniques for Enhanced Quality of Computer Accompaniment«. In: *Proceedings of the 1988 International Computer Music Conference*. 1988.

[DP11] Zhiyao Duan und Bryan Pardo. »Soundprism: An online system for score-informed source separation of music audio«. In: *IEEE Journal of Selected Topics in Signal Processing* 5.6 (2011), S. 1205–1215.

[DW05] Simon Dixon und Gerhard Widmer. »MATCH: A Music Alignment Tool Chest«. In: *Proceedings of the 6th International Conference on Music Information Retrieval*. 2005, S. 492–497.

[Flo] *Flowkey*. URL: http://www.flowkey.com (besucht am 27. 10. 2016).

[For73] George David Forney. »The viterbi algorithm«. In: *Proceedings of the IEEE* 61.3 (1973), S. 268–278.

[FRK10] Ludwig Fahrmeir, Günter Raßer und Thomas Kneib. *Skript zur Vorlesung Stochastische Prozesse*. Ludwig-Maximilians-Universität München, Institut für Statistik, 2010. URL: http://www.statistik.lmu.de/institut/lehrstuhl/semwiso/stochastische-prozesse/vorlesung.html.

[GCEM16] Jean-Louis Giavitto, Arshia Cont, José Echeveste und MuTant Team Members. *Antescofo – a not-so-short introduction to version 0.x*. 2016.

[GD97] Lorin Grubb und Roger B. Dannenberg. »A Stochastic Method of Tracking a Vocal Performer«. In: *Proceedings of the 1997 International Computer Music Conference*. 1997.

[GD98] Lorin Grubb und Roger B. Dannenberg. »Enhanced Vocal Performance Tracking Using Multiple Information Sources«. In: *Proceedings of the 1998 International Computer Music Conference*. 1998.

[HDHW00] Hank Heijink, Peter Desain, Henkjan Honing und Luke Windsor. »Make Me a Match: An Evaluation of Different Approaches to Score Performance Matching«. In: *Computer Music Journal* 24.1 (2000), S. 43–56.

[Imu] *IMuSE*. URL: http://debussy.music.ubc.ca/muset/imuse.html (besucht am 15. 02. 2017).

[Jor07] Anna Jordanous. »Score Following: An Artificially Intelligent Musical Accompanist«. Masterarbeit. University of Edinburgh, 2007.

[KKAW13] Filip Korzeniowski, Florian Krebs, Andreas Arzt und Gerhard Widmer. »Tracking rests and Tempo changes: Improved Score following with Particle filters«. In: *Proceedings of the 39th International Computer Music Conference*. 2013.

[KR06] Hagen Kaprykowsky und Xavier Rodet. »Globally Optimal Short-Time Dynamic Time Warping, Application to Score to Audio Alignment«. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 2006, S. 249–252.

[LD15] Bochen Li und Zhiyao Duan. »Score Following for Piano Performances with Sustain-Pedal Effects«. In: *Proceedings of the 16th International Society for Music Information Retrieval Conference*. 2015, S. 469–475.

[MD10] Robert Macrae und Simon Dixon. »A guitar tablature score follower«. In: *Proceedings of the 2010 IEEE International Conference on Multimedia and Expo*. 2010, S. 725–726.

[Mül07] Meinard Müller. »Dynamic Time Warping«. In: *Information Retrieval for Music and Motion*. Springer Berlin Heidelberg, 2007, S. 69–84.

[OD01] Nicola Orio und François Déchelle. »Score Following Using Spectral Analysis and Hidden Markov Models«. In: *Proceedings of the 2001 International Computer Music Conference*. 2001.

[OS01] Nicola Orio und Diemo Schwarz. »Alignment of Monophonic and Polyphonic Music to a Score«. In: *Proceedings of the 2001 International Computer Music Conference*. 2001.

[PB01] Bryan Pardo und William P. Birmingham. »Following a musical performance from a partially specified score«. In: *Proceedings of the Multimedia Technology Applications Conference*. 2001.

[PB02] Bryan Pardo und William P. Birmingham. »Improved Score Following for Acoustic Performances«. In: *Proceedings of the 2002 International Computer Music Conference*. 2002.

[PB05] Bryan Pardo und William P. Birmingham. »Modeling Form for On-line Following of Musical Performances«. In: *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference (USA)*. 2005, S. 1018–1023.

[Phe14] Phenicx Consortium. *Towards enriched classical concert experiences: first integrated prototypes of the PHENICX project*. 2014.

[PL92] Miller Puckette und Cort Lippe. »Score Following in Practice«. In: *Proceedings of the 1992 International Computer Music Conference*. 1992.

[Puc90] Miller Puckette. »EXPLODE: A User Interface for Sequencing and Score Following«. In: *Proceedings of the 1990 International Computer Music Conference*. 1990.

[Rab89] Lawrence R. Rabiner. »A tutorial on hidden Markov models and selected applications in speech recognition«. In: *Proceedings of the IEEE*. Bd. 77. 2. 1989, S. 257–286.

[Rap04] Christopher Raphael. »A Hybrid Graphical Model for Aligning Polyphonic Audio with Musical Scores«. In: *Proceedings of the 5th International Conference on Music Information Retrieval*. 2004.

[Rap99] Christopher Raphael. »Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models«. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.4 (1999), S. 360–370.

[RHP13] Martin Ritter, Keith Hamel und Bob Pritchard. »Integrated Multimodal Score-following Environment«. In: *Proceedings of the 39th International Computer Music Conference*. 2013.

[RN93] Matthew S. Ryan und Graham R. Nudd. *The Viterbi Algorithm*. Techn. Ber. 1993.

[Ros96] Sheldon M. Ross. *Stochastic Processes*. 2. Aufl. Wiley, 1996.

[RVC15] Francisco Rodríguez-Serrano, Pedro Vera-Candeas und Julio Carabias-Orti. »A real-time score follower for MIREX 2015«. In: *Music Information Retrieval Evaluation eXchange*. 2015.

[She64] Roger N. Shepard. »Circularity in Judgments of Relative Pitch«. In: *The Journal of the Acoustical Society of America* 36.12 (1964), S. 2346–2353.

[SmaA] *SmartPiano*. URL: http://www.smartpiano.com/pages/smart-piano (besucht am 27. 10. 2016).

[SmaB] Jacob Kastrenakes. *The One Smart Piano teaches you to play using an iPad*. URL: http://www.theverge.com/2015/6/29/8860927/the-one-smart-piano-and-keyboard-indiegogo (besucht am 27. 10. 2016).

[SOS04] Diemo Schwarz, Nicola Orio und Norbert Schnell. »Robust Polyphonic Midi Score Following with Hidden Markov Models«. In: *Proceedings of the 2004 International Computer Music Conference*. 2004.

[Ste15] Steinberg Media Technologies GmbH. *Cubase Pro 8 Benutzerhandbuch*. 2015.

[Toi99] Petri H. Toiviainen. »An interactive midi accompanist«. In: *Computer Music Journal* 22.4 (1999), S. 63–75.

[Ton] *Tonara Interactive Piano Sheet Music*. URL: https://itunes.apple.com/us/app/tonara-interactive-piano-sheet/id454753605 (besucht am 27. 10. 2016).

[Van95] Jason D. Vantomme. »Score following by temporal patterns«. In: *Computer Music Journal* 19.3 (1995), S. 50–59.

[Ver84] Barry Vercoe. »The Synthetic Performer in The Context of Live Performance«. In: *Proceedings of the 1984 International Computer Music Conference*. 1984.

[VP85] Barry Vercoe und Miller Puckette. »Synthetic Rehearsal: Training the Synthetic Performer«. In: *Proceedings of the 1985 International Computer Music Conference*. 1985.

[Wer14] Erik Werner. »Realisierung eines webbasierten Score-Followers für Tasteninstrumente mit Chroma-Features«. Masterarbeit. Technische Universität Berlin, 2014.

[WS13] Karl-Heinz Waldmann und Ulrike M. Stocker. *Stochastische Modelle. Eine anwendungsorientierte Einführung*. 2. Aufl. Springer-Verlag Berlin Heidelberg, 2013.

[Yu10] Shun-Zheng Yu. »Hidden semi-Markov models«. In: *Artificial Intelligence* 174.2 (2010), S. 215–243.

Liste der Errata

zur Masterarbeit »Der automatische Registrant. Erschließung von Score-Following zur Setzersteuerung von Orgeln«, verfasst von Markus Maier, eingereicht im Februar 2017

Fehlerhafte Stelle	Original	Korrektur
Seite 90, Tabelle 4.7, Datensatz 1, Spalte 4	9.0	3.8
Seite 90, Tabelle 4.8, Datensatz 1, Spalte 4	10.0	3.8
Seite 92, Tabelle 4.11, Datensatz 1, Spalte 3	11.0	4.5

Stand: 05.03.2017