



Automatic Drum Transcription



Diplomarbeit 2005

Drum2Midi

Studenten

Bürki Hanspeter

Herzig Patrick

Betreuer

Dr. Franz Bachmann

Dr. Werner Bäni

Experte

Dr. Hansjörg Klock

Burgdorf, im Januar 2005

Zusammenfassung

Sie hören an einem Konzert einen für Ihr Ohr besonders gut klingenden Rhythmus. Er gefällt Ihnen so, dass Sie ihn gerne selber nachspielen möchten. Sie können den schnellen Schlägen aber nicht folgen, und im Nu ist Ihnen der Rhythmus aus dem Hinterkopf verschwunden. Wie könnte man das Gespielte in Erinnerung behalten?

Eine Möglichkeit bietet die Notenschrift. Wie es beim Klavier für jeden Ton eine Notenlinie gibt, so existieren beim Schlagzeug Notenlinien für die verschiedenen Instrumente. Wie ist es nun aber möglich, von einem Schlagzeugsolo die Notenschrift zu erhalten? Der Bericht zeigt eine Variante auf, wie es mit technischen Mitteln realisierbar ist.

Die automatische Umwandlung („Transcription“) von Musikstücken in Notenschrift (MIDI-Daten) ist ein bekanntes Problem der musikalischen Akustik. Das oben erwähnte Beispiel der Transcription eines Konzertes ist zu anspruchsvoll, um es im Rahmen einer Diplomarbeit lösen zu können. Deshalb wurde die Problemstellung ausschliesslich auf Schlagzeugmusik eingeschränkt. Das Resultat soll ein System sein, welches einem Schlagzeugspieler ermöglicht, von gespielten Rhythmen die Noten auszudrucken. Ein Schlagzeuglehrer kann dann beispielsweise seinem Schüler die Noten zu Übungszwecken nach Hause mitgeben. Der Schüler wiederum ist in der Lage zu prüfen, ob das Gespielte mit den Noten übereinstimmt.

Basierend auf der vorangegangenen Semesterarbeit führten wir die „Untersuchung von Schlagzeugtönen“ fort. Nach einer gründlichen Internet-Recherche entschieden wir uns für zwei Methoden, welche auf unterschiedliche Weise das Transcription-Problem lösen. Die eine widmet sich dem Template Matching und die andere versucht die Problemstellung mittels Mustererkennung zu lösen. Die Kombination beider Methoden erhöht die Zuverlässigkeit.

Zur Bewerkstelligung der Aufgabe benutzten wir hauptsächlich die Mathematiksoftware Matlab 6.5. Aus Geschwindigkeitsgründen mussten einige Funktionen in die Programmiersprache C / C++ übersetzt werden.

Als Resultat steht ein System (Computer, Mikrophon und von uns erstellte Software) zur Verfügung, welches Schlagzeugmusik mit den Instrumenten Pauke, Snare, Tom1, Tom2, Tom3, Hihat, Crash, Splash, Ride und Kuppel in MIDI-Form umwandeln kann.

Die Musiksignale können entweder direkt vom „Line-In“ Eingang der Soundkarte oder von bereits abgespeicherten Wav-Dateien eingelesen werden. Die Software verfügt zudem über einen Demonstrationsmodus, welcher für didaktische Zwecke gedacht ist. Dieser stellt die Klassifikation einzelner Schläge anschaulich dar.

Dieser Bericht gibt dem technisch versierten Leser Einblick in die Funktionsweise der „Automatic Drum Transcription“. Kenntnisse aus dem Musikbereich sind zum Verständnis nicht zwingend notwendig.

Übersicht

1	EINLEITUNG	1
2	DAS SCHLAGZEUG	2
3	BESCHRIEB DER ALGORITHMEN	4
4	BESCHRIEB DER SOFTWARE.....	35
5	BEDIENUNGSANLEITUNG	41
6	TESTRESULTATE	46
7	SCHLUSSWORT UND DISKUSSION.....	51
8	LITERATURVERZEICHNIS	52
A	HILFSPROGRAMME	
B	ORDNERSTRUKTUR DER CD	
C	AUFGABENSTELLUNG	
D	ERKLÄRUNG DER DIPLOMANDEN	

Inhaltsverzeichnis

1	EINLEITUNG	1
1.1	ALLGEMEIN	1
1.2	ZIEL DER DIPLOMARBEIT	1
2	DAS SCHLAGZEUG	2
2.1	FELLINSTRUMENTE	2
2.2	METALLINSTRUMENTE	3
3	BESCHRIEB DER ALGORITHMEN	4
3.1	DETEKTIEREN DER ONSETS	4
3.1.1	<i>Übersicht</i>	6
3.1.2	<i>Onset Detection</i>	7
3.1.3	<i>Feature Onsets bestimmen</i>	9
3.2	TEMPLATE MATCHING	10
3.2.1	<i>Einführung</i>	10
3.2.2	<i>Übersicht</i>	12
3.2.3	<i>Der Template Matching Algorithmus</i>	14
3.2.4	<i>Grenzen des Template Matchings</i>	16
3.3	FEATURES	17
3.3.1	<i>Übersicht</i>	17
3.3.2	<i>FeatValue – Merkmalswerte</i>	18
3.3.3	<i>InstrClassification – Instrumentenklassifikation</i>	24
3.3.4	<i>Merkmalsräume</i>	25
3.4	VEREINIGUNG VON MATCHING- UND FEATURERESULTATEN	33
4	BESCHRIEB DER SOFTWARE	35
4.1	ÜBERSICHT	35
4.2	BETRIEBSMODI	36
4.2.1	<i>Live2Midi</i>	36
4.2.2	<i>Wav2Midi</i>	39
4.2.3	<i>DemoMode</i>	39
4.3	LEISTUNGSFÄHIGKEIT	40
4.3.1	<i>Instrumente</i>	40
4.3.2	<i>Geschwindigkeit</i>	40
4.3.3	<i>Dynamik</i>	40
5	BEDIENUNGSANLEITUNG	41
5.1	SYSTEMANFORDERUNGEN	41
5.2	INSTALLATION	41
5.3	DIE BENUTZER OBERFLÄCHE	42
5.3.1	<i>Einstellungen</i>	44
6	TESTRESULTATE	46
6.1	EINLEITUNG	46
6.2	TESTS AB NOTENBLATT	46
6.3	TEST MIT EIGENEN RHYTHMEN	48
6.4	DYNAMIK- UND GESCHWINDIGKEITSTESTS	49
6.5	ZUSAMMENFASSUNG DER TESTS	50

7	SCHLUSSWORT UND DISKUSSION.....	51
7.1	DISKUSSION DER ERGEBNISSE	51
7.2	VERBESSERUNGSVORSCHLÄGE	51
7.3	SCHLUSSWORT	51
8	LITERATURVERZEICHNIS	52
A	HILFSPROGRAMME	
B	ORDNERSTRUKTUR DER CD	
C	AUFGABENSTELLUNG	
D	ERKLÄRUNG DER DIPLOMANDEN	

1 Einleitung

1.1 Allgemein

Mit dem Musikgehör von Mozart könnten wir problemlos die einzelnen Instrumente aus einem Orchester erkennen. Den Ton einer Trompete zu bestimmen, würde uns keine Mühe bereiten. Wir wären in der Lage, das Gespielte in Noten umzusetzen. Da nicht jeder so begabt ist wie Mozart, versuchen wir die Aufgabe mit Hilfe der Technik zu lösen. Unser Ziel ist ein „kleiner elektronischer Mozart“, der aus Musikstücken Noten generieren kann. Die Diplomarbeit „Drum2Midi“ zielt darauf ab, ein solches System für Schlagzeugspieler zu realisieren. Dabei dürfen keine anderen Instrumente als das Schlagzeug verwendet werden.

1.2 Ziel der Diplomarbeit

Ziel der Diplomarbeit ist es, von einem Schlagzeugspiel die Noten aufzuzeichnen. Das heisst, ein Schlagzeugspieler spielt ein Drumsolo, das mittels Mikrofon an den Computer übermittelt wird. Auf dem Computer läuft eine Windows Applikation, welche die Audiodaten in die entsprechenden Noten umwandelt. Als Resultat liegt schlussendlich eine MIDI-Datei vor, welche alle wichtigen Informationen des Gespielten enthält.

Ein Schlagzeug wird je nach Musikstilrichtung verschieden angespielt. Es existiert eine riesige Menge an Klängen, die mit einem Schlagzeug erzeugt werden können. Wir lösen die Aufgabe unter folgenden Bedingungen:

- Es darf nie mehr als ein Metallinstrument zur gleichen Zeit angespielt werden
- Metallinstrumente dürfen alleine oder in Kombination mit Pauke oder Snare gespielt werden
- Unter den Fellinstrumenten sind diverse Kombinationen erlaubt
- Die Dynamik muss klein gehalten werden
- Zwischen zwei Schlägen muss mindestens 100ms Zeit verstreichen

Auf dem Markt sind zurzeit Systeme erhältlich, die mittels piezoelektrischen Sensoren die Schläge an jedem Instrument selbst erkennen können. Nachteil dieser Systeme ist jedoch der hohe Preis. Unser System dagegen soll für jeden Schlagzeuger erschwinglich sein. Man benötigt lediglich einen Computer und ein Mikrofon.

Burgdorf, im Januar 2005

Hanspeter Bürki

Patrick Herzig

.....

.....

2 Das Schlagzeug

Unser Schlagzeugset (Abbildung 1) besteht aus den Instrumenten Pauke, Tom1, Tom2, Tom3, Snare, Crash, Splash, Ride, Kuppel und Hihat. Die einzelnen Instrumente werden anschliessend näher erläutert.



Abbildung 1: Das Schlagzeug-Set mit dem wir das System entwickelt haben

Die Schlägel, mit denen man die Instrumente anspielt, werden Sticks genannt.

2.1 Fellinstrumente

Snare Das Fell der Snare ist auf ihrer Unterseite mit Metalldrähten bespannt, welche ihr den typischen Klang verleihen. Die Drähte können gespannt oder gelöst werden. Wir verwenden die Snare nur mit gespannten Drähten.

Der Rimshot ist eine Spielweise, bei der, wie es der Name schon verrät, der Stick gleichzeitig Fell und Rand der Trommel trifft und somit einen besonders lauten und kesselartigen Ton bewirkt. Ein Rimshot wird von unserer Software als normaler Snareschlag erkannt.

Der Rimclick dürfte vielen von der Nutzung in ruhigen Stücken bekannt sein. Hierbei wird der Stick auf das Fell gelegt und mit dessen Schaft auf den Rand der Snare geschlagen. Es entsteht ein „Klick“-Geräusch. Die Software ist für Rimclicks nicht trainiert.

Tom1, Tom2 und Tom3 Die Trommeln unterscheiden sich anhand ihrer Durchmesser und den daraus resultierenden Klängen.

Pauke Sie wird mit dem Fusspedal angespielt und tönt sehr dumpf.

2.2 Metallinstrumente

Hihat Wird häufig als Taktgrundlage verwendet. Die gegeneinander liegenden Becken des Hihats können mit einem Fusspedal mehr oder weniger stark zusammen gepresst werden. Hihat closed steht für satt gegeneinadergepresste Becken, Hihat half wenn das Fusspedal etwas gelockert wird und Hihat open wenn sich die zwei Becken nicht mehr berühren. Die Software generiert nur ein Typ von Hihatschlägen in der MIDI-Datei. Auf Hihat open Schläge ist das System nicht trainiert.

Crash Der Klang ist sehr laut und klingt lange nach. Wird vor allem bei Taktübergängen gespielt.

Splash Tönt wie ein lautes, abklingendes Rauschen.

Ride Kann auch als Taktgrundlage verwendet werden. Im Gegensatz zum Hihat klingt es lange, aber nur sehr leise nach.

Kuppel Das Ride kann auch in der Umgebung der Aufhängung angespielt werden. Diese Region heisst Kuppel und stellt ein zusätzliches Instrument dar. Der Klang gleicht dem Ride, ist aber deutlich härter.

3 Beschrieb der Algorithmen

Das Drum2Midi-System besteht aus folgenden Algorithmen:

- Detektieren der Onsets
- Template Matching
- Features
- Vereinigung von Matching- und Featurerisultaten

In Kapitel 4.1(Beschrieb der Software) Abbildung 31 wird das Zusammenwirken der Algorithmen anschaulich dargestellt.

Mit dem ersten Algorithmus wird erkannt zu welcher Zeit ein Schlag ertönt. Die folgenden zwei Algorithmen widmen sich der Erkennung des Schlages. Es ist zu erwähnen, dass das Template Matching ausschliesslich Fellinstrumente klassifiziert, während die Features für das Erkennen von Fell- und Metallinstrumenten geeignet sind. Der letzte Punkt beschreibt, wie die Ergebnisse der beiden Erkennungsalgorithmen zu einem zuverlässigen Resultat verknüpft werden.

3.1 Detektieren der Onsets

Als Onset wird der Zeitpunkt bezeichnet, bei dem ein Schlag zu klingen beginnt. Also der Zeitpunkt, wo ein Instrument angeschlagen wird. In Abbildung 2 ist ein Audiosignal mit acht Onsets abgebildet. Im Zeitbereich sind diese acht Onsets nicht ersichtlich. Der Grund liegt darin, dass die Schläge nach dem Onset weiterklingen, was zu einer Überlagerung der einzelnen Schläge führt. In der Wellenform eines klingenden Schlages kann so nur sehr schwer ein zusätzlich ertönender Schlag erkannt werden.

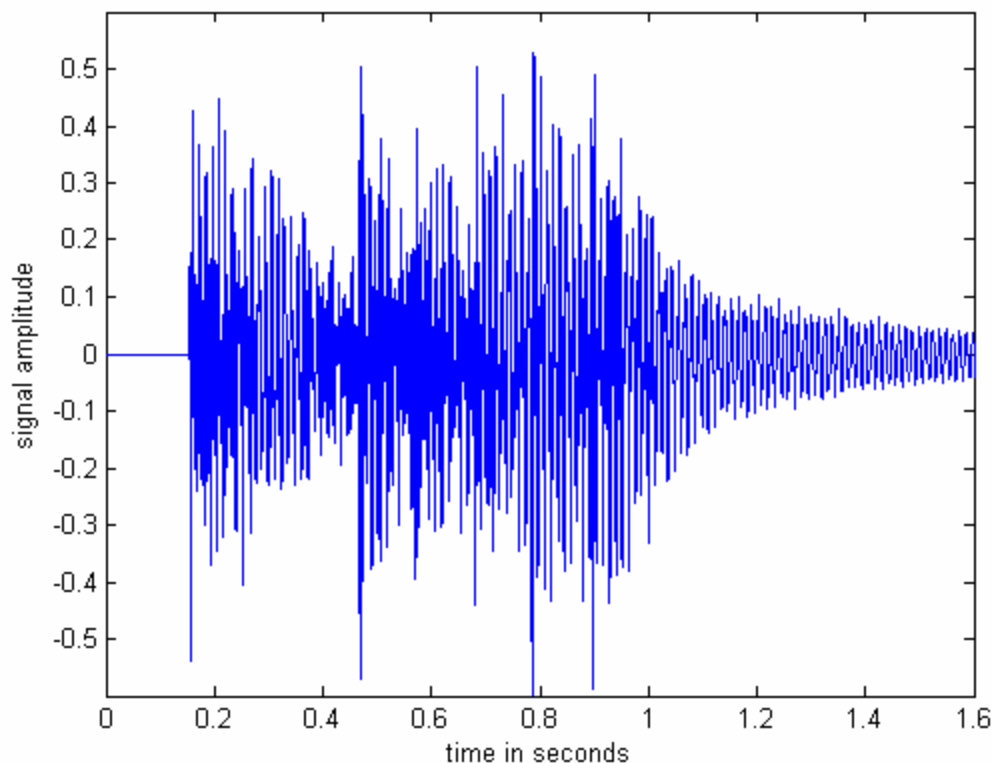


Abbildung 2: Im Zeitbereich sind die acht Schläge nicht erkennbar

Aus diesen Gründen benutzen wir als Grundlage für die Detektierung der Onsets das Spektrum (Betrag der Kurzzeit-Fouriertransformation). Daraus sind auch in bereits klingenden Schlägen weitere Onsets als vertikale Linien problemlos erkennbar. Das Bild in Abbildung 3 ist entsprechend dem Betrag der Kurzzeit-Fouriertransformation eingefärbt. Die Farben sind so gewählt, dass kleine Werte blau, und grosse Werte braun (oder dunkelrot) erscheinen.

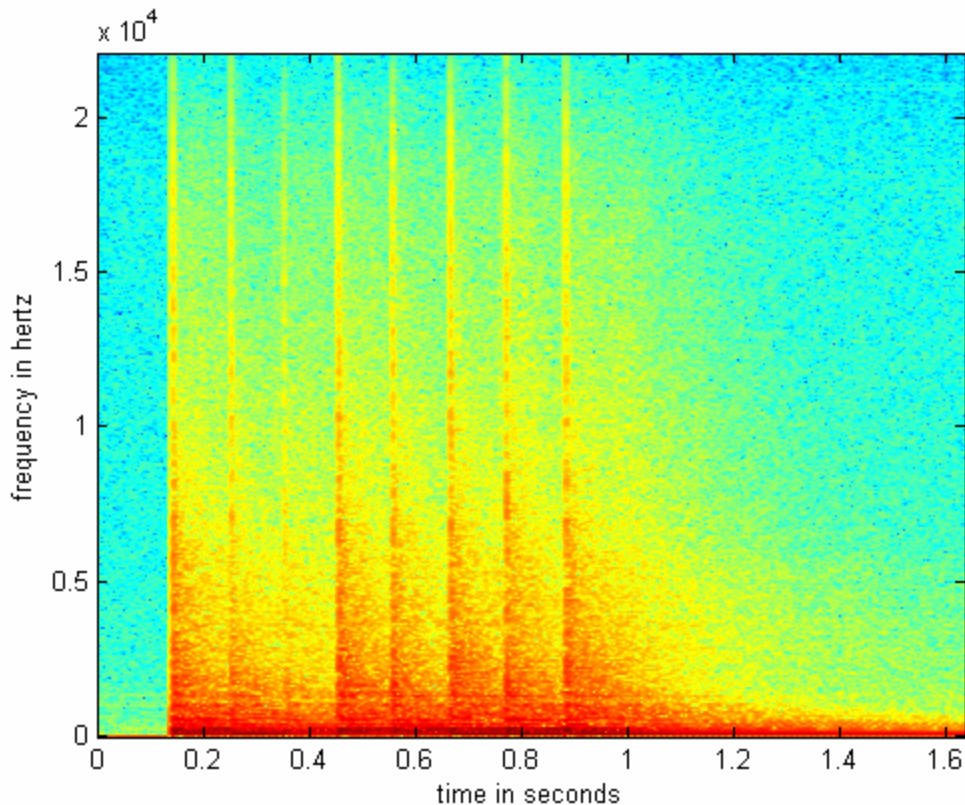


Abbildung 3: Im Spektrum sieht man die acht Schläge gut

Eigenschaften Onset-Spektrum:

- Sampling Frequenz: 44100Hz
- FFT-Länge: 1024, entspricht einer Frequenzauflösung von 43Hz
- Fensterlänge (Hanning Fenster): 1024
- Überlappung: 936
- Frame-Länge: Fensterlänge – Überlappung = 88 Samples, entspricht 2ms

Es wird eine grosse Überlappung benötigt, damit die zeitliche Auflösung (ca. 2ms) genügend hoch ist.

Im weiteren Text wird das Spektrum mit $s(t, f)$ bezeichnet. Dabei ist t die Nummer des Zeitfensters und f die Nummer des Frequenzbins.

3.1.1 Übersicht

Das Herz der „Onset Detektierung“ ist ein Algorithmus, welcher die Onsets aus einem wählbaren Frequenzbereich des Spektrogramms bestimmen kann. Die Software berechnet die Onsets doppelt. Einerseits für ein Band im oberen Frequenzbereich (Bereich Frequenzbin: 300-512) und andererseits für ein Band im unteren Frequenzbereich (Bereich Frequenzbin: 5-128).

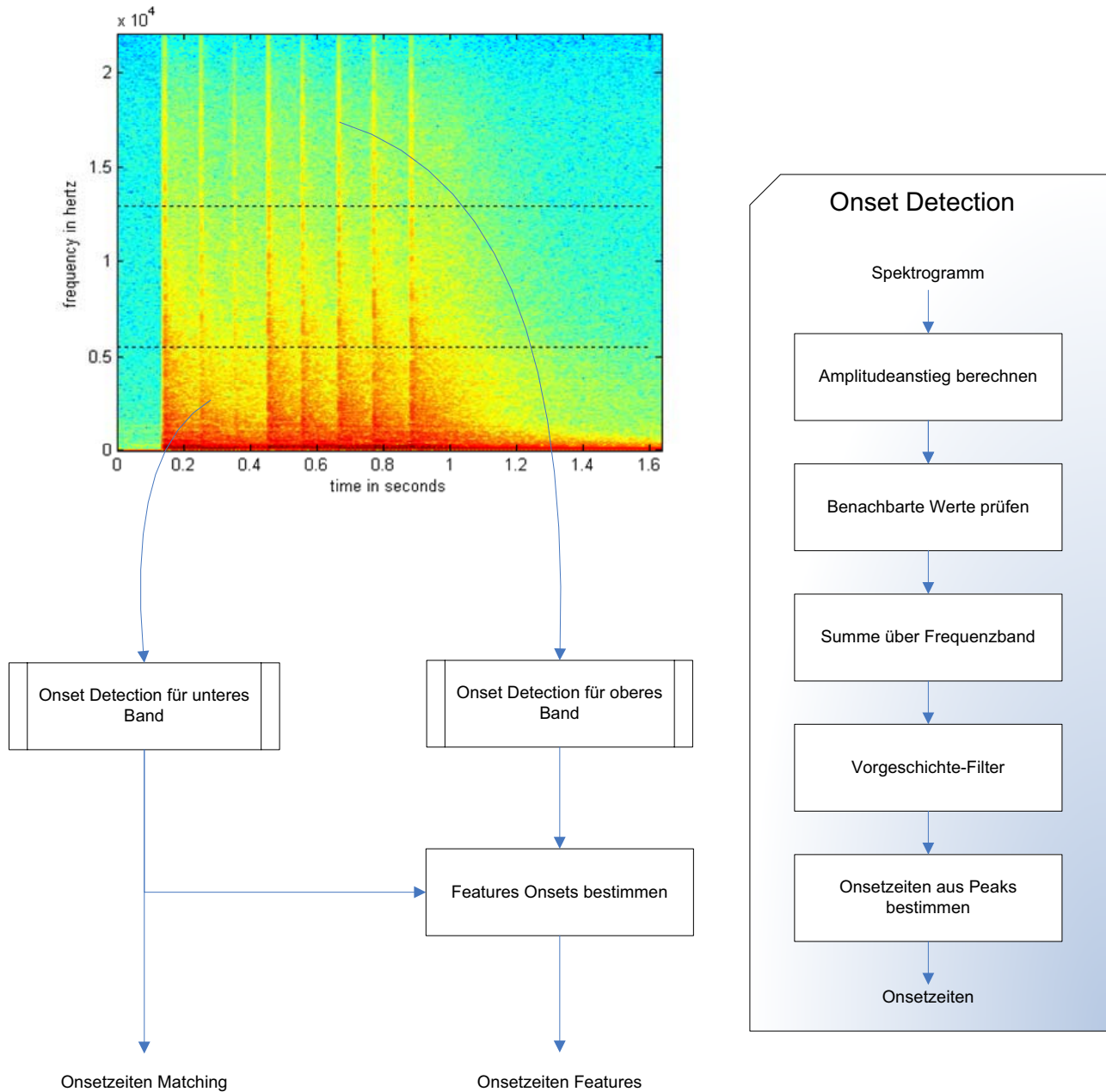


Abbildung 4: Funktionsweise der Onset Detection

Im unteren Frequenzband werden nicht alle Onsets von Metallinstrumenten erkannt. Da mit der Matchingmethode keine Metallinstrumente klassifiziert werden, ist es sinnvoll, die Onsetzeiten nur aus dem unteren Band zu verwenden. Sie sind für Fellinstrumente zudem genauer. Mit den Features werden alle Instrumente klassifiziert. Der Funktionsblock „Feature Onsets bestimmen“ ergänzt deshalb die Onsets des oberen Bandes mit Onsets, welche nur im unteren Band erkannt wurden. So entstehen die Onsetzeiten der Features, die auch am Schluss zur Generierung der MIDI-Datei verwendet werden. Abbildung 4 verdeutlicht den Ablauf und zeigt die verwendeten Funktionsblöcke. Im nächsten Kapitel wird erläutert, wie die zwei Onsetmengen berechnet werden.

3.1.2 Onset Detection

Es geht darum, dem Computer beizubringen, wie er den Zeitpunkt der Onsets (vertikale Linien) im Spektrogramm $s(t, f)$ zuverlässig bestimmen kann. Der Ablauf ist in Abbildung 4 rechts dargestellt.

Amplitudeanstieg berechnen Der Amplitudeanstieg $d(t, f)$ ist wie folgt definiert:

$$d(t, f) = s(t+1, f) - s(t, f)$$

Benachbarte Werte prüfen Da nur der Beginn eines Schlages interessiert (ansteigende Amplitude), lassen wir nur noch diejenigen Werte stehen, welche selbst positiv sind und zeitlich vor und danach ($d(t-1, f)$, $d(t+1, f)$) einen positiven Wert aufweisen. Die Anderen setzen wir auf Null:

$$\tilde{d}(t, f) = (d(t-1, f) > 0) \cdot (d(t, f) > 0) \cdot (d(t+1, f) > 0) \cdot d(t, f)$$

Das Prüfen der Nachbarwerte ($d(t-1, f)$, $d(t+1, f)$) wird in obiger Formel mit einbezogen, damit am Schluss ein aussagekräftigeres Onsetsignal entsteht.

Summe über Frequenzband Mit dem Aufsummieren über ein Frequenzband entsteht das Summensignal $o(t)$.

$$o(t) = \sum_f \tilde{d}(t, f)$$

Für die Onsets des unteren Bandes wird über die Frequenzbins 5-128 aufsummiert. Für die Onsets des oberen Bandes wird über die Frequenzbins 300-512 aufsummiert.

Abbildung 5 zeigt das Summensignal $o(t)$ für die acht Onsets, wobei zwei nur schwer zu erkennen sind.

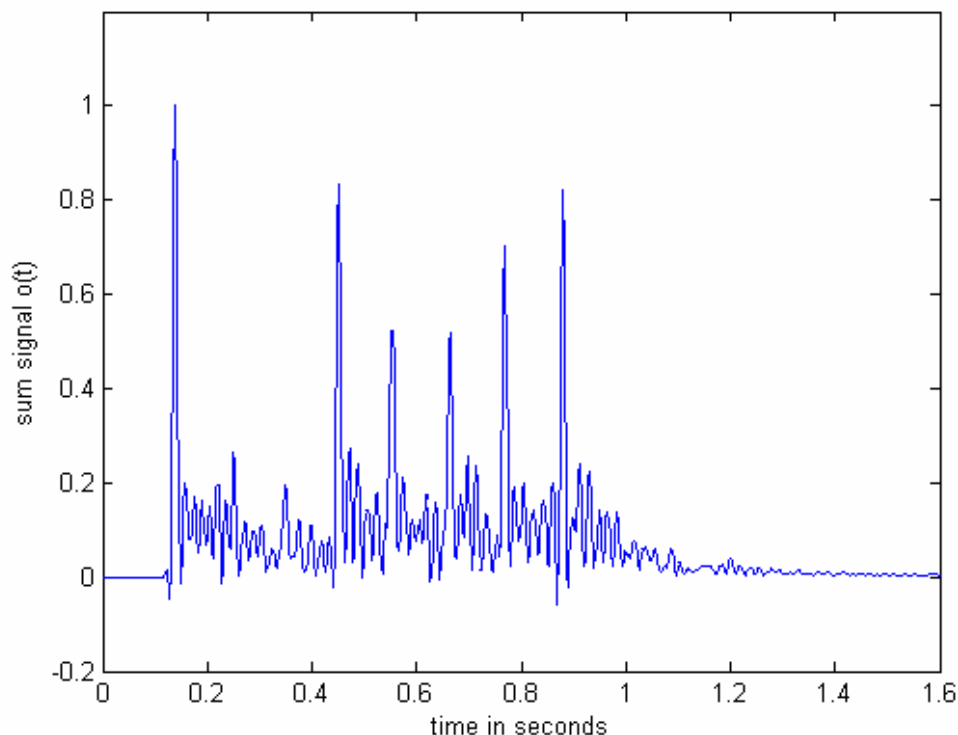


Abbildung 5: Das ungefilterte Summensignal

Vorgeschichte-Filter Das Summensignal $o(t)$ basiert auf der Arbeit von Goto[13]. Es wird mit einem von uns benannten „Vorgeschichte-Filter“ bearbeitet, damit es den hohen Anforderungen einer zuverlässigen Onset detection entspricht. Nachfolgender Text beschreibt die Aufarbeitung des Onset Signals:

Wir lassen einen Wert nur stehen, wenn er höher als der Mittelwert von vierzig Werten unmittelbar vor ihm ist. Damit können wir fehlerhafte Spitzen, welche innerhalb eines Schlages auftreten, wirksam löschen. Ob ein Wert stehengelassen wird oder ob er gelöscht wird, hängt von der Vorgeschichte ab. Deshalb nennen wir es Vorgeschichte-Filter. Abbildung 6 wiedergibt das gefilterte Onset Signal.

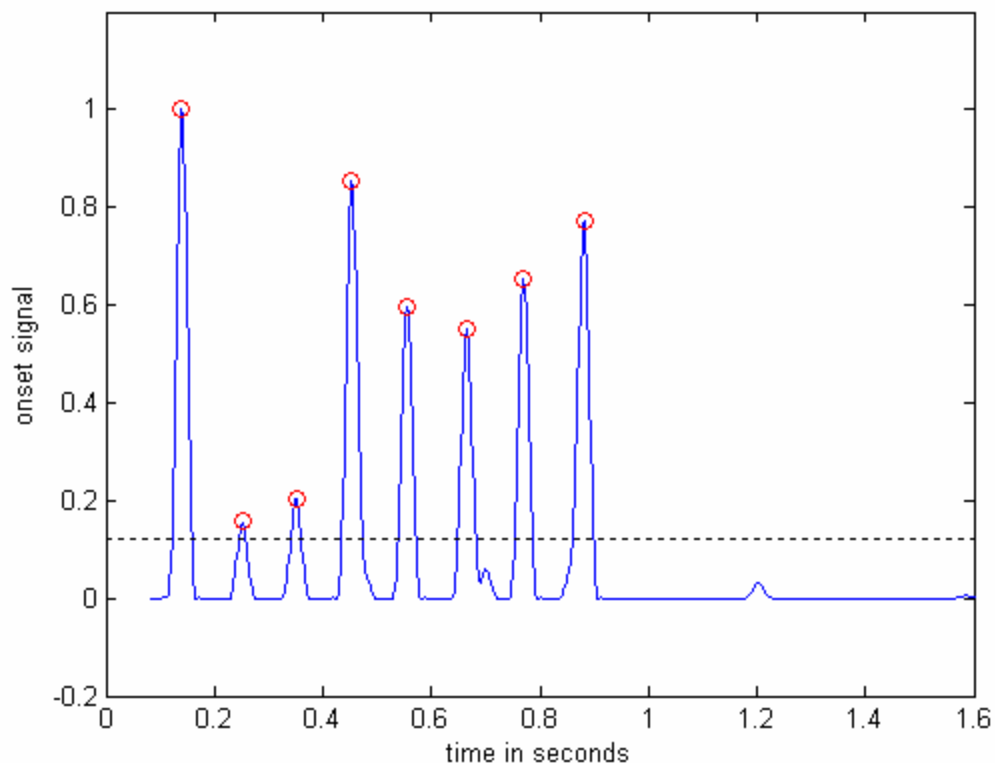


Abbildung 6: Das Onsetsignal nach dem Vorgeschichte-Filter

Onsetzeiten aus Peaks bestimmen Die Peaks (rote Kringel) des Onset Signals lassen auf die Onsetzeiten schließen. Mittels einer Schwelle, welche in Abbildung 6 als gestrichelte Linie angedeutet ist, wird sichergestellt, dass nur deutliche Schläge zu einem Onset führen.

3.1.3 *Feature Onsets bestimmen*

Fellinstrumente, mit einem erheblich kleineren Anteil an hohen Frequenzen als Metallinstrumente, dominieren im unteren Band. Onsets, welche von Paukenschlägen stammen, werden manchmal nur im unteren Band detektiert. Der Funktionsblock „Feature Onsets bestimmen“ fügt die Resultate der beiden Bänder zusammen und sorgt dafür, dass keine doppelten Einträge vorkommen. Die meisten Schläge werden in beiden Bändern detektiert. Von ihnen wird, wenn die Differenz nicht zu gross ist, der Mittelwert genommen. Die Onsetzeiten vom oberen und unteren Band können sich für einem bestimmen Schlag deutlich unterscheiden. Dies trifft vor allem bei polyphonen Schlägen mit Metall- und Fellinstrumenten zu. Das Zustandekommen des Unterschieds ist verständlich, wenn man bedenkt, dass ein Schlagzeugspieler zwei Instrumente nicht exakt zur gleichen Zeit anschlägt. Das Ausmass der Differenz hängt davon ab, wie exakt gespielt wird.

3.2 Template Matching

3.2.1 Einführung

Der Matching Prozess [13] vergleicht den zu untersuchenden Schlag mit Vorlagen. Die Vorlagen sind Spektrogramme (hier Quadrat des Betrages der Kurzzeit-Fouriertransformation) eines jeden Instruments und werden im Text als Templates bezeichnet. Sie werden beim Trainieren des Systems erstellt. Die Vergleiche finden anhand von charakteristischen Punkten der jeweiligen Vorlage statt.

Von einem zu untersuchenden Schlag wird genau gleich wie bei den Templates das Spektrogramm berechnet. Es wird mit allen Templates verglichen und im weiteren Excerpt genannt. Sowohl Excerpts als auch Templates werden auf ihren Maximalwert normiert.

Die folgenden Spektrogramme (Abbildung 7 und Abbildung 8) stellen die Daten zweier Templates anschaulich dar:

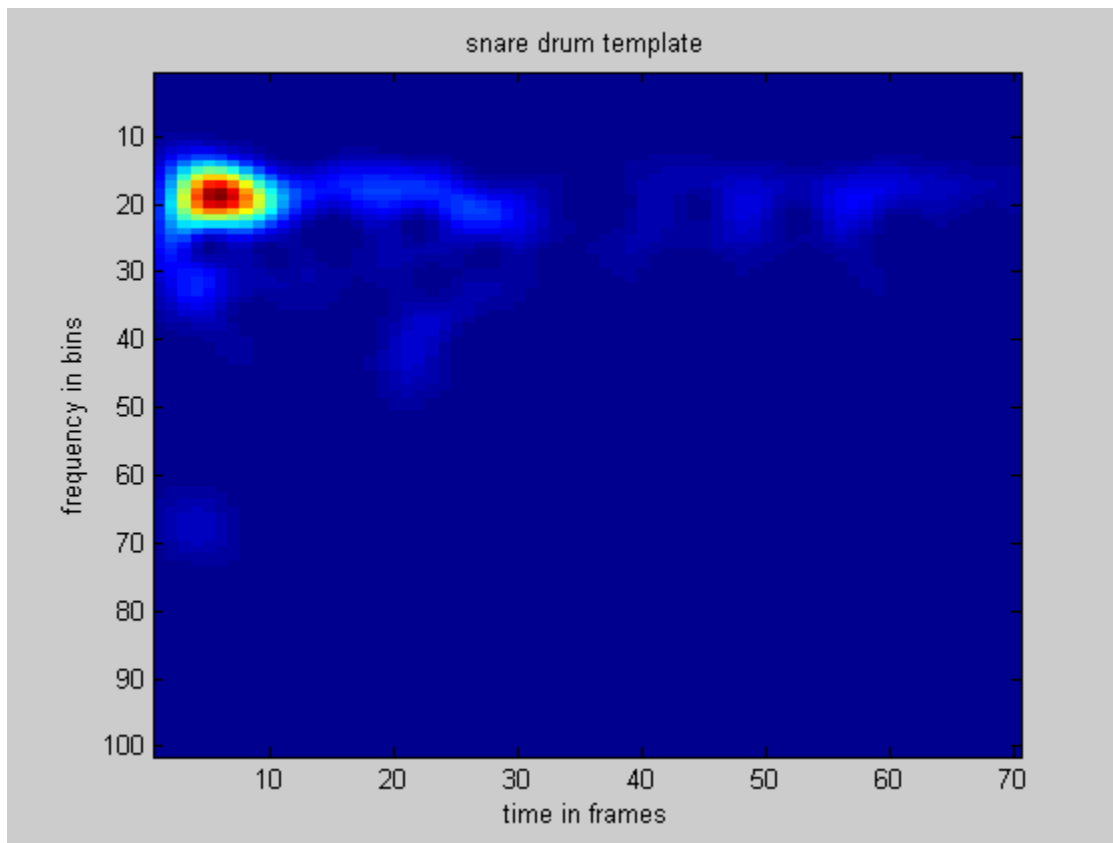


Abbildung 7: Template einer Snare

Um die charakteristischen Punkte eines Templates zu bestimmen, berechnet das Programm die 100 grössten Werte im Spektrogramm.

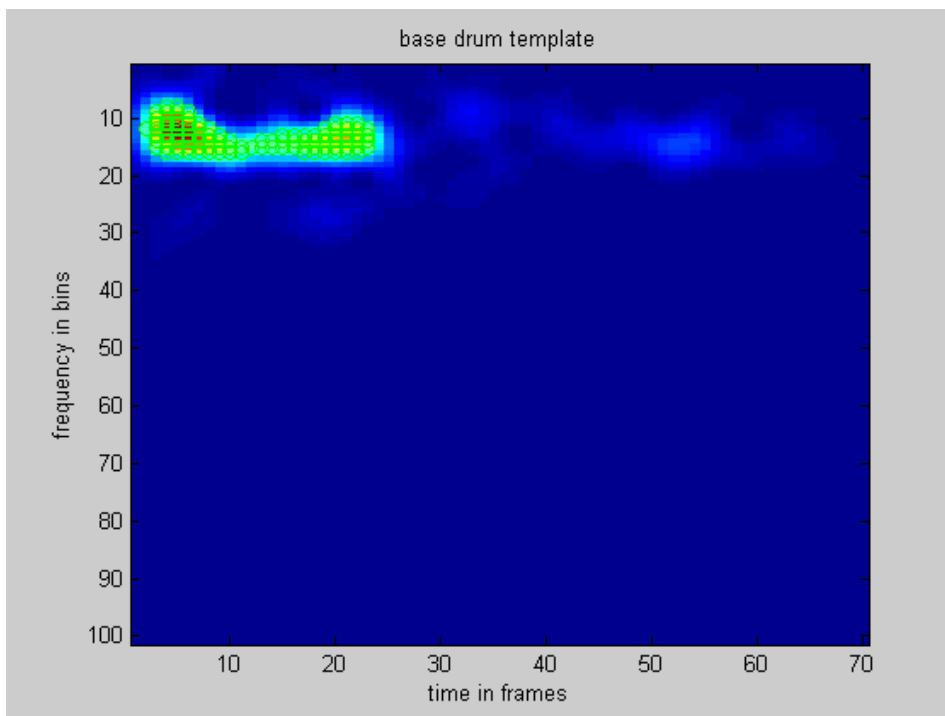


Abbildung 8: Template einer Pauke mit den vom Programm bestimmten charakteristischen Punkten (grün)

Im Weiteren wird das Spektrogramm eines polyphonen Excerpts von Pauke und Snare dargestellt:

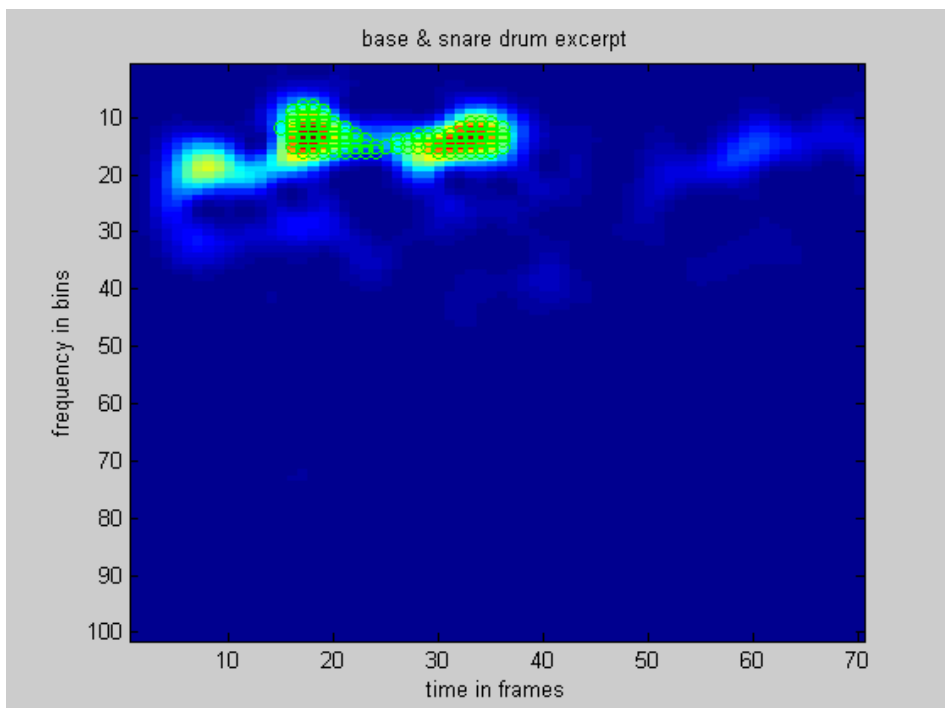


Abbildung 9: Die grünen Kreise sind die charakteristischen Punkte des Paukentemplates. Sie markieren hier den Paukenanteil im polyphonen Schlag.

3.2.2 Übersicht

Ein Vergleich und ein Zuweisen der Excerpts zu den entsprechenden Templates ist von Auge in vielen Fällen möglich. Der Vergleich mit technischen Mitteln ist nicht unproblematisch.

Der Mensch kann die charakteristischen Punkte eines jeden Templates gut bestimmen. Es sind die rot- und gelb-farbigen Bereiche. Dort sind die Amplituden hoch. Unser Hirn merkt sich beim Vergleich die Form des Fleckens. Wir schauen bei einem Excerpt, ob wir diese Form irgendwo wiederfinden.

Abbildung 9 stellt das Spektrogramm eines polyphonen Schlages von Snare und Pauke dar. Obwohl das eine Muster zeitlich später im Spektrogramm erscheint als auf dem Template, ordnen wir es völlig unbewusst dem Pauke Template zu. Wir erkennen von Auge die beiden Muster des Snare- und Pauke Templates im Excerpt. Die Differenz (ca. 20ms) zwischen den beiden Schlägen ist für das Ohr nicht hörbar.

Unser Algorithmus vergleicht nicht die Form, sondern merkt sich die zu einem Template gehörigen charakteristischen Punkte (Energie bei bestimmten Frequenzen und Zeiten). Er prüft bei einem Excerpt, ob er an diesen Stellen in gleicher oder stärkerer Weise markante Energiewerte findet. Abbildung 10 zeigt den Ablauf des Template Matchings.

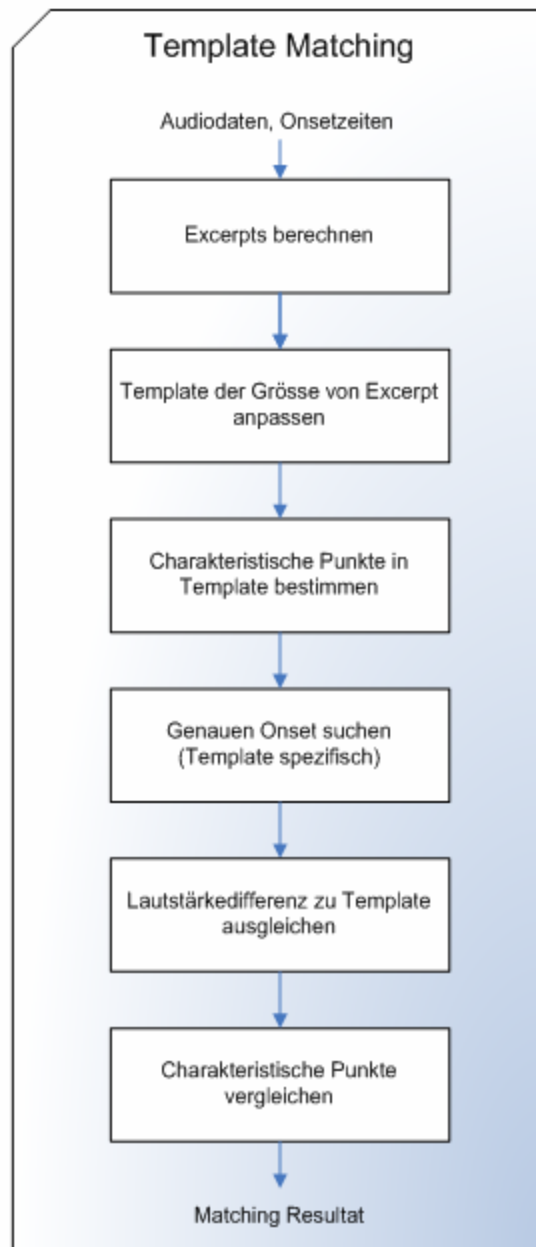


Abbildung 10: Der Weg von den Audiodaten zum Matching Resultat

3.2.3 Der Template Matching Algorithmus

Die einzelnen Funktionsblöcke werden nun näher erläutert.

Excerpts berechnen Bei jedem Onset wird wenn möglich, aus dem Audiosignal ein Stück von 70 Frames (ca. 140ms) herausgeschnitten. Ist die Zeitdauer zwischen zwei Onsets kürzer als 70 Frames, so verkürzt sich die Dauer des ausgeschnittenen Signals entsprechend. Die Funktion „Excerpts berechnen“ erstellt zu jedem von diesen Signalstücken das Spektrogramm mit den hier aufgeführten Eigenschaften:

- Sampling Frequenz: 44100Hz
- Fensterlänge (Hanning Fenster): 1024
- Überlappung: 936
- Frequenzvektor: 0..1000Hz in 10Hz Schritten
- Frame-Länge: Fensterlänge – Überlappung = 88 Samples, entspricht 2ms

Template der Grösse von Excerpt anpassen Ein Template ist 70 Frames lang. Folgen die Onsets näher aufeinander (→ verkürzte Excerpts), so muss, damit verglichen werden kann, die Templatelänge der Länge des Excerpts angepasst werden.

Charakteristische Punkte in Template bestimmen Das Programm berechnet die 100 grössten Werte des Template Spektrogramms und nimmt sie als die charakteristischen Punkte. Der Grösste wird als der stärkste charakteristische Punkt bezeichnet. Sie werden in den nachfolgenden Formeln *tmp_spec_points* genannt.

Genauen Onset suchen (Template spezifisch) Bei polyphonen Schlägen unter Fellinstrumenten führt die Ungenauigkeit des Spielers zu zeitlich unterschiedlichem Erscheinen der beiden Schläge im Spektrogramm. Diese Differenz, in Abbildung 11 dargestellt, gilt es auszugleichen.

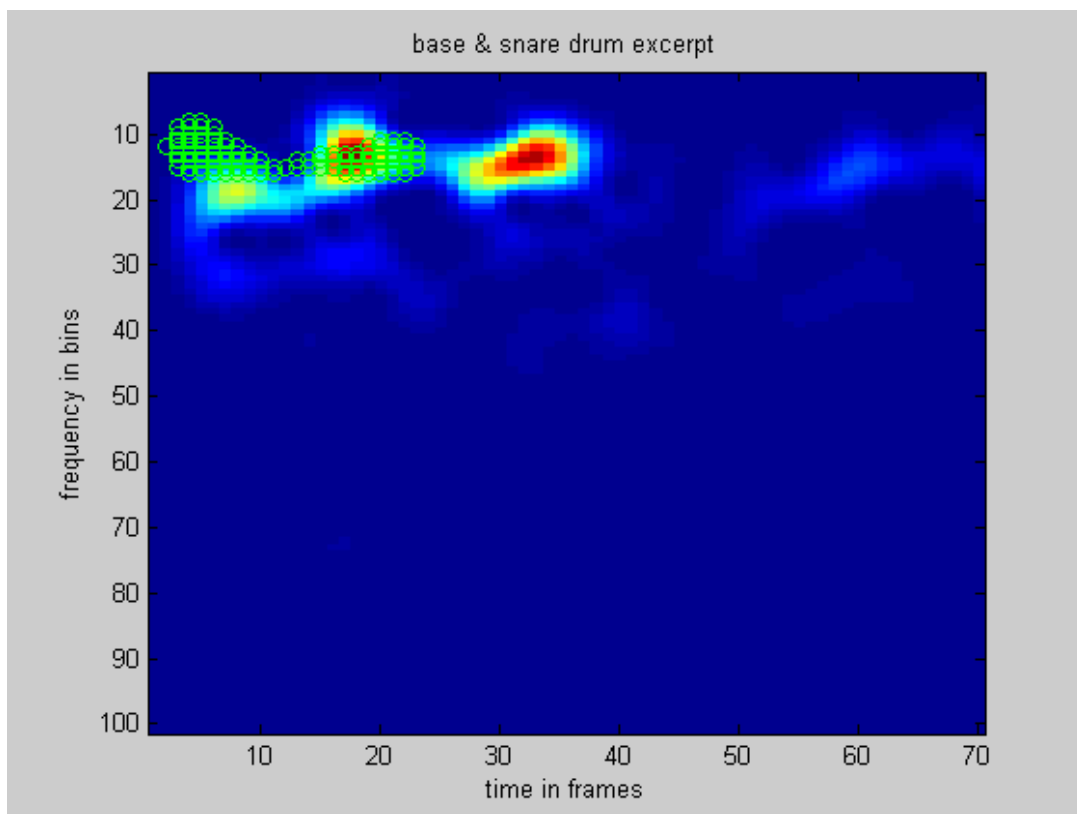


Abbildung 11: Bei den grünen Kreisen im Excerpt wird ein Flecken erwartet, aber nicht gefunden

Um den für das jeweilige Template genauen Onset zu suchen, muss eine geeignete Methode zum Messen der Übereinstimmung von zwei Flecken vorhanden sein. Nachfolgender Pseudocode beschreibt die verwendete Messmethode:

- Differenz zwischen Excerpt und Template an jedem der 100 charakteristischen Punkte berechnen. Dies ergibt Differenzvektor dif
- Die Differenzen an den fünf stärksten charakteristischen Punkten werden als dif_5 bezeichnet
- Differenzen gewichten: $dif_b = dif < del_ons_level$

Man beachte, dass dif_b ein boolescher Vektor mit Einsen und Nullen ist. del_ons_level stellt die Gewichtungsschwelle dar.

- Übereinstimmung u berechnen: $u = \sum dif_5 - \frac{\sum dif_b}{100} \cdot 5$

Diese Formel ist aus empirischen Untersuchungen entstanden.

Der Algorithmus ermittelt den genauen Onset in zwanzig Schritten. Für jeden Schritt werden die charakteristischen Punkte im Spektrogramm des Excerpts jeweils um ein Frame nach rechts verschoben. In jedem dieser zwanzig Zeitpunkte berechnet das Programm die Übereinstimmung. Für das jeweilige Template wird der Zeitwert der grössten Übereinstimmung als gültigen Onset betrachtet. Die diesem Onset zugehörigen, gegenüber dem Original zeitlich verschobenen charakteristischen Punkte, werden als ex_spec_points bezeichnet.

Lautstärkedifferenz zu Template ausgleichen Das menschliche Gehör ist unterschiedlich empfindlich auf verschiedene Frequenzen. D.h. ein Ton (harmonische Schwingung) von 80dB bei 4kHz empfinden wir lauter, als ein Ton von 80dB bei 100 Hz. Damit wir zwei Schläge als gleich laut empfinden, schlagen wir ohne es zu merken Instrumente mit tiefen Tönen stärker an als solche mit Frequenzen im Bereich von 4kHz. Deshalb erscheinen die Muster von Pauke und Snare ungleich stark im Spektrogramm (siehe Abbildung 11). Es gilt diese Differenz auszugleichen, allerdings nur, wenn es sich tatsächlich um ein Signal und nicht nur um „Rauschen“ handelt. Der Pseudocode dazu lautet wie folgt:

```

if max(excerpt(ex_spec_points)) – min(excerpt(ex_spec_points)) > noise_level
    excerpt = Mittelwert(excerpt)
end;

```

Charakteristische Punkte vergleichen Dieser Funktionsblock entscheidet, ob das gerade verwendete Template mit dem Excerpt übereinstimmt oder nicht. Der Vergleich findet nur an den charakteristischen Punkten statt und lässt sich beschreiben als:

$$dif = \sum ((excerpt(ex_spec_points) - template(tmp_spec_points)) < match_level)$$

$$match = dif < match_ratio$$

$match = 1$ → Übereinstimmung

$match = 0$ → keine Übereinstimmung

3.2.4 Grenzen des Template Matchings

Tom3 Schläge weisen ein sehr breites Spektrum an Frequenzen auf (siehe Abbildung 12). Es kommt häufig vor, dass sich starke Amplitudenanteile von einem Tom3 Schlag in den Bereichen der charakteristischen Punkte von anderen Instrumenten (insbesondere Snare) befinden. Folglich erkennt der Algorithmus Instrumente, die gar nicht gespielt wurden (vor allem Snare). So wird ein Tom3 Schlag vom Matching meistens als polyphoner Schlag aus Tom3 und Snare erkannt.

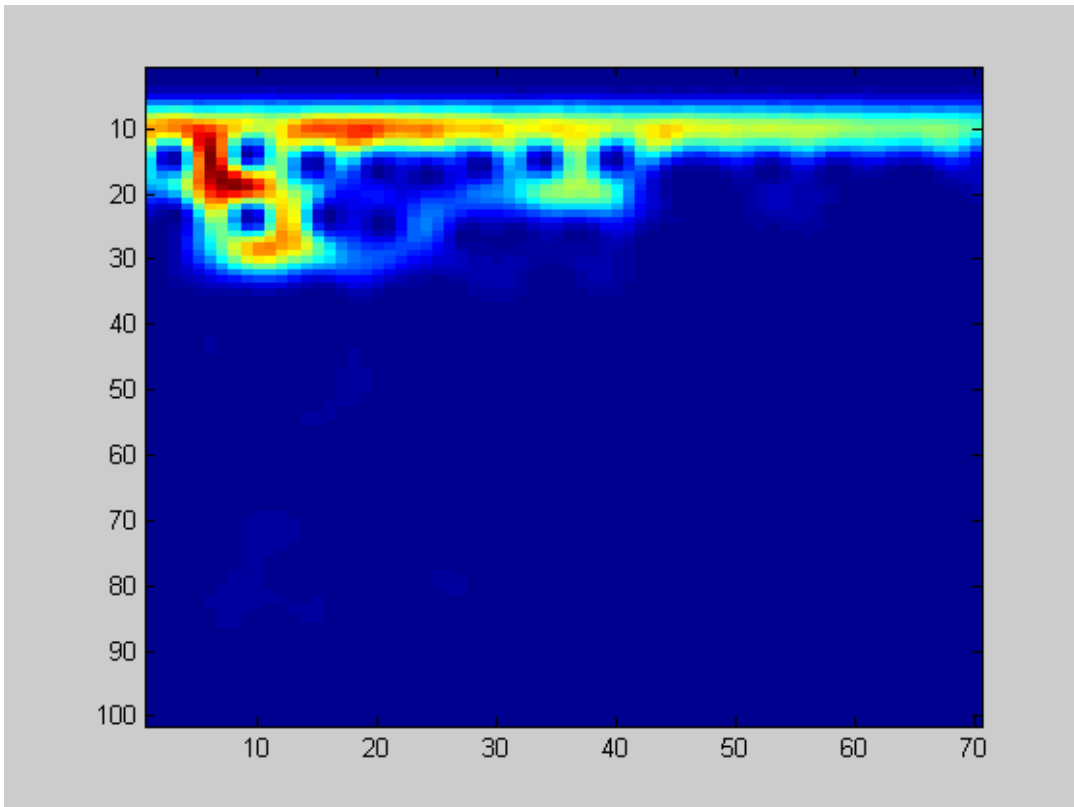


Abbildung 12: Ein Tom3 Schlag bringt das Spektrum „zum Leuchten“

Um diesem Problem auszuweichen haben wir versucht „Formen“ zu vergleichen. Doch auch dies schien uns nicht die Lösung zu sein. Beim polyphonen Schlag aus Snare und Pauke wie in Abbildung 13 rechts dargestellt, vermischen sich die charakteristischen Punkte:

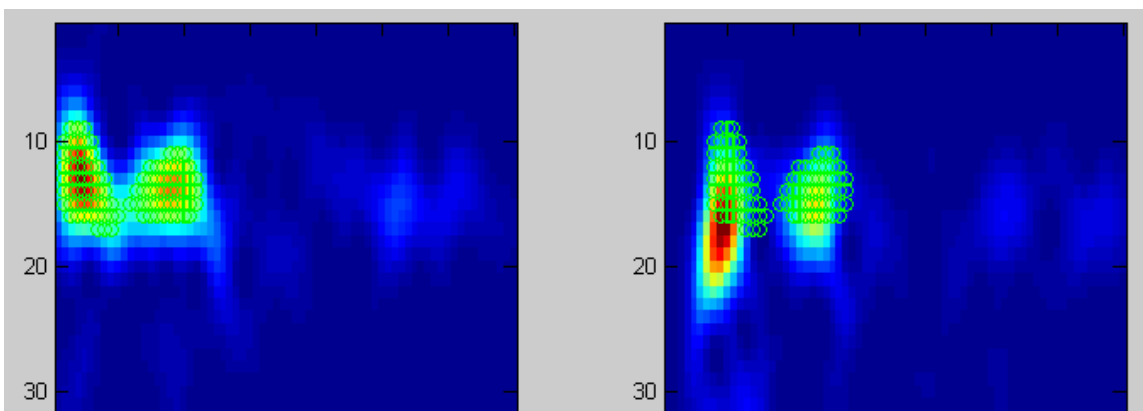


Abbildung 13: Links Pauken Template, rechts polyphoner Schlag aus Pauke und Snare

Die Formen in Abbildung 13 sind nicht gleich. Mit dem „Formen-Vergleich-Ansatz“ würde die Pauke nicht erkannt.

3.3 Features

3.3.1 Übersicht

Die Schlagzeuginstrumente weisen alle unterschiedliche Merkmale auf. Das Ziel des Feature-Algorithmus ist es, diese Merkmale zu nutzen um die Instrumente unterscheiden zu können.

Abbildung 14 zeigt den groben Ablauf zur Erkennung eines Instrumentes im Merkmalsraum.

Die beiden Teile „FeatValue – Merkmalswerte“ und „InstrClassification – Instrumentenklassifikation“ werden in den nachfolgenden Kapiteln detaillierter erklärt.

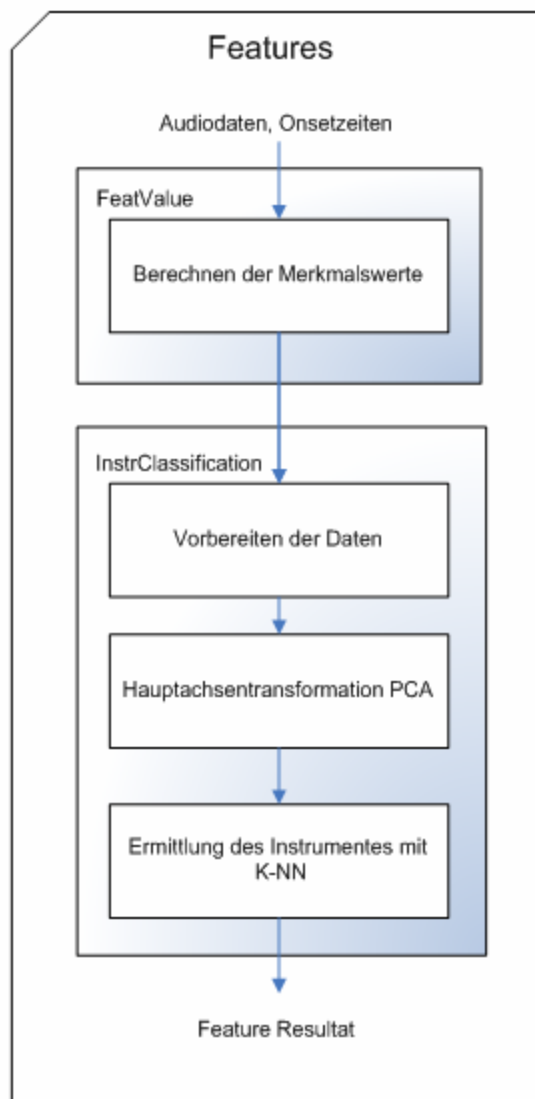


Abbildung 14: Allgemeiner Weg zur Klassifizierung im Merkmalsraum

3.3.2 *FeatValue* – Merkmalswerte

Um die verschiedenen Instrumente unterscheiden zu können, sind signifikante Merkmale (Features) notwendig. Von ursprünglich ca. 115 verschiedenen Features wurden mittels Tests die aussagekräftigsten ermittelt. Der Algorithmus benötigt so nur noch zwanzig Features, basierend auf sieben Featurearten. Diese werden nachfolgend genauer erklärt.

Zero Crossing Rate (ZCR)

Das Merkmal “Zero Crossing Rate” [6] ist definiert als Anzahl Nulldurchgänge des Audiosignals bezogen auf die Anzahl möglicher Nulldurchgänge.

Formel 3.1 beschreibt das ZCR-Merkmal.

$$ZCR = \frac{1}{M-1} \sum_{m=1}^{M-1} |\text{sign}(x(m)) - \text{sign}(x(m-1))| \quad (3.1)$$

M: Anzahl Samples

x(m): Wert des m-ten Samples

Die Berechnung der ZCR erfolgt am Signal von Onset zu Onset. Dieses Merkmal erweist sich als sehr hilfreich zur Unterscheidung von Metall- und Fellinstrumenten. Metallinstrumente weisen höhere ZCR-Werte auf als Fellinstrumente. Abbildung 15 links zeigt das Signal eines Hihat mit sehr vielen Signalnulldurchgängen (hoher ZCR-Wert) und Abbildung 15 rechts das Signal eines Tom1 mit sehr wenigen Signalnulldurchgängen (kleiner ZCR-Wert).

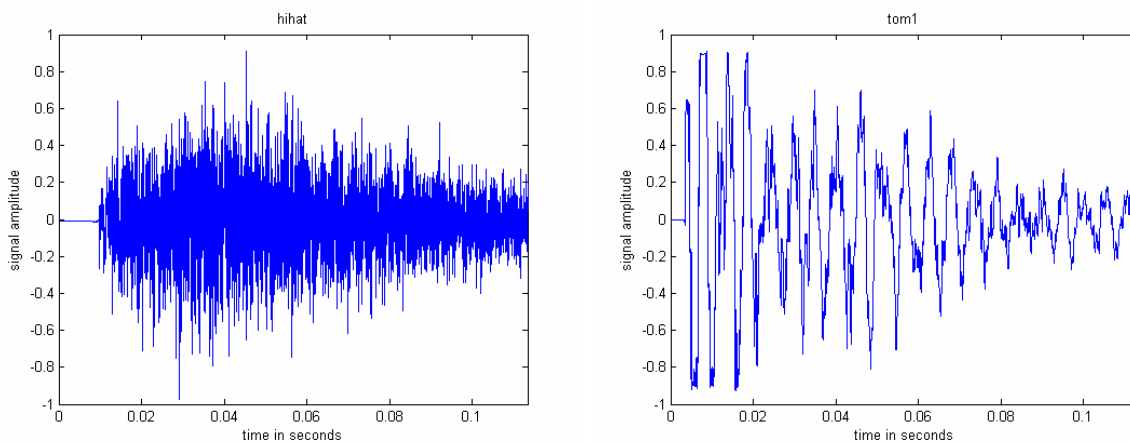


Abbildung 15 links: Signal eines Hihat-Schlages, rechts: Signal eines Tom1-Schlages

Spectral Centroid Frame

Das Merkmal „Spectral Centroid Frame“ [1] (SC) ist definiert als Schwerpunkt des Amplitudenspektrums.

Formel 3.2 beschreibt das Merkmal „Spectral Centroid Frame“.

$$SC_t = \frac{\sum_{k=1}^{N/2} X_t[k] \cdot F[k]}{\sum_{k=1}^{N/2} X_t[k]} \quad (3.2)$$

$X_t[k]$: Amplitudenspektrum bei Frequenzbin k und Frame t

$F[k]$: Frequenz bei Frequenzbin k

N : FFT-Ordnung

„Spectral Centroid Frame“ wird anhand des Spektrogrammes errechnet. Für jedes Frame erhält man somit einen SC-Wert. Als Merkmalswert dient uns schlussendlich der Mittelwert der SC-Werte aller Frames. Abbildung 16 zeigt die Amplitudenspektren eines zufällig gewählten Frames eines Pauke- und eines Hihatschlages. Der rote Kringle auf der Nulllinie kennzeichnet den berechneten Spectral Centroid Wert.

Dieses Feature erweist sich als sehr hilfreich für die Erkennung von Pauken und Tom-Schläge.

Eigenschaften Spectral Centroid-Spektrogramm:

- Sampling Frequenz: 44100Hz
- FFT-Länge: 1024, entspricht einer Frequenzauflösung von 43Hz
- Fensterlänge (Hanning Fenster): 1024
- Überlappung: 936
- Frame-Länge: Fensterlänge – Überlappung = 88 Samples, entspricht 2ms
- Audiosignallänge ab Onsetzeitpunkt: 150ms

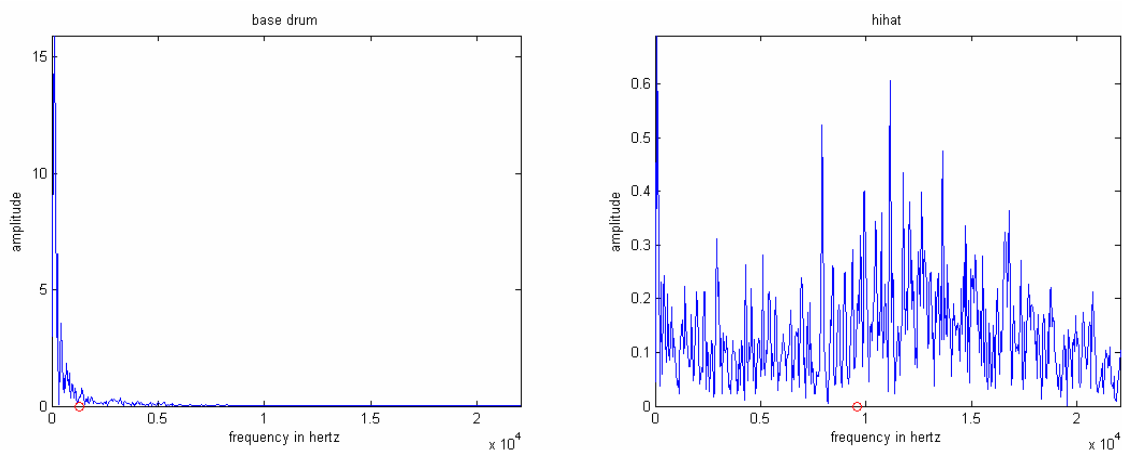


Abbildung 16: Amplitudenspektrum eines Paukenschlages (links) und eines Hihat-Schlages (rechts)

Spectral Rolloff

„Spectral Rolloff“ [1] ist definiert als Frequenzbin R , wobei 85% der gesamten Amplitudenspektrum-Verteilung unterhalb R liegen.

Die mathematische Beschreibung des „Spectral Rolloff“ Merkmals ist in Formel 3.3 zusehen.

$$\sum_{k=1}^R X[k] \approx 0.85 \sum_{k=1}^{N/2} X[k] \quad (3.3)$$

$X[k]$: Amplitudenspektrum bei Frequenzbin k

k : Frequenzbin

N : FFT-Ordnung (N wird auf eine gerade Zahl abgerundet)

R : Spectral Rolloff Frequenzbin

Die Berechnung des „Spectral Rolloff“ erfolgt nicht framewise sondern wird anhand der FFT über die ersten 150ms ab Onsetzeitpunkt berechnet. Abbildung 17 zeigt den Verlauf des aufsummierten Amplitudenspektrums (blaue Linie) für einen Tom2-Schlag (links) und einen Hihat-Schlag (rechts). Die roten Kringel kennzeichnen die Frequenzbins, wo 85% der gesamten Amplitudenspektrum-Verteilung erreicht wird. Im Hintergrund ist zudem das Amplitudenspektrum der Schläge abgebildet.

Dieses Merkmal eignet sich zur Erkennung einerseits von Pauken und Tom-Schlägen andererseits von Hihat-Schlägen.

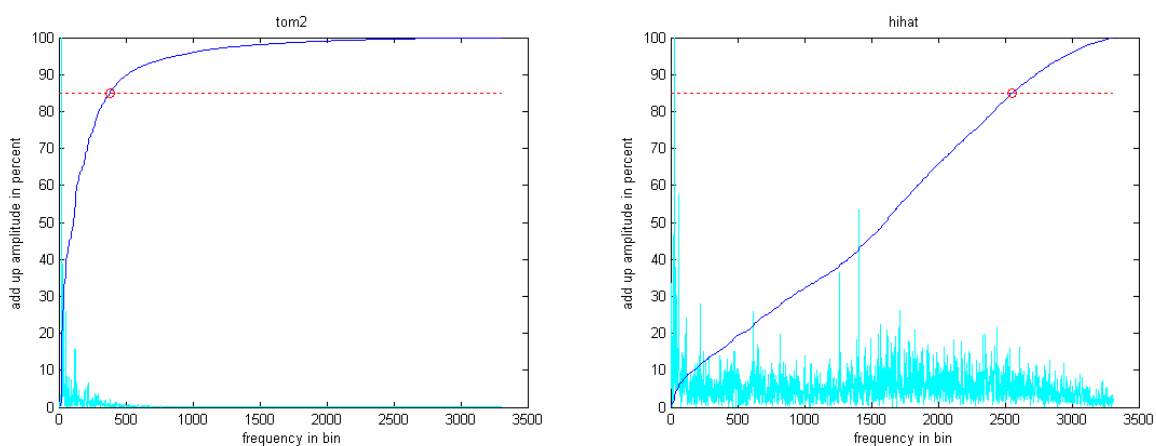


Abbildung 17 Aufsummiertes Signal des Amplitudenspektrums eines Tom2-Schlages (links) und eines Hihat-Schlages (rechts).

Spectral Flatness

„Spectral Flatness“ [5] ist definiert als das Verhältnis zwischen dem geometrischen Mittel und dem arithmetischen Mittel des Amplitudenspektrums.

$$SF = \frac{\left(\prod_{k=1}^{N/2} X[k] \right)^{\frac{1}{N/2}}}{\frac{1}{N/2} \sum_{k=1}^{N/2} X[k]} \quad (3.4)$$

$X[k]$: Amplitudenspektrum bei Frequenzbin k

k : Frequenzbin

N : FFT-Ordnung (N wird auf eine gerade Zahl abgerundet)

Abbildung 18 zeigt links das Amplitudenspektrum eines Hihat-Schlages (grosser Spectral Flatness Wert) und rechts eines Kuppel-Schlages (kleiner Spectral Flatness Wert). Die Berechnung dieses Features erfolgt anhand der FFT über die ersten 150ms ab Onsetzeitpunkt.

„Spectral Flatness“ wird eingesetzt zur Erkennung der Hihat-Schläge.

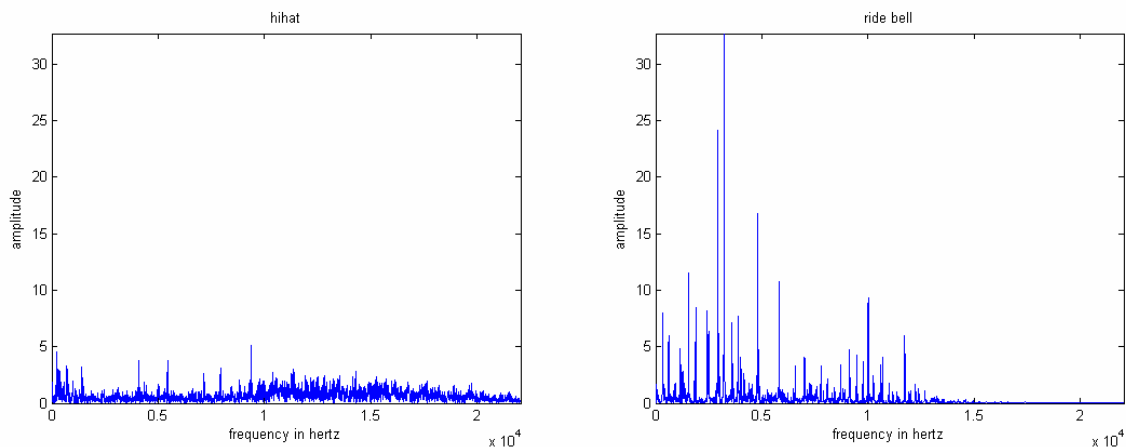


Abbildung 18: Amplitudenspektrum eines Hihat-Schlages (links) und eines Kuppel-Schlages (rechts)

Spectral Strong Peak Sample

Das Amplitudenspektrum des Signals wird für die Berechnung des „Spectral Strong Peak Sample“ Merkmals mit einem Gaussfilter geglättet um so den groben Verlauf des Spektrums zu erhalten (Abbildung 19 blaue Linien). In Abbildung 19 sind auf halber Peakhöhe des Signales zwei rote Kringle eingezeichnet. Die Anzahl Frequenzbins zwischen diesen beiden Punkten wird als Merkmalswert verwendet. Dieses Merkmal dient zur Erkennung von Hihat-Schlägen.

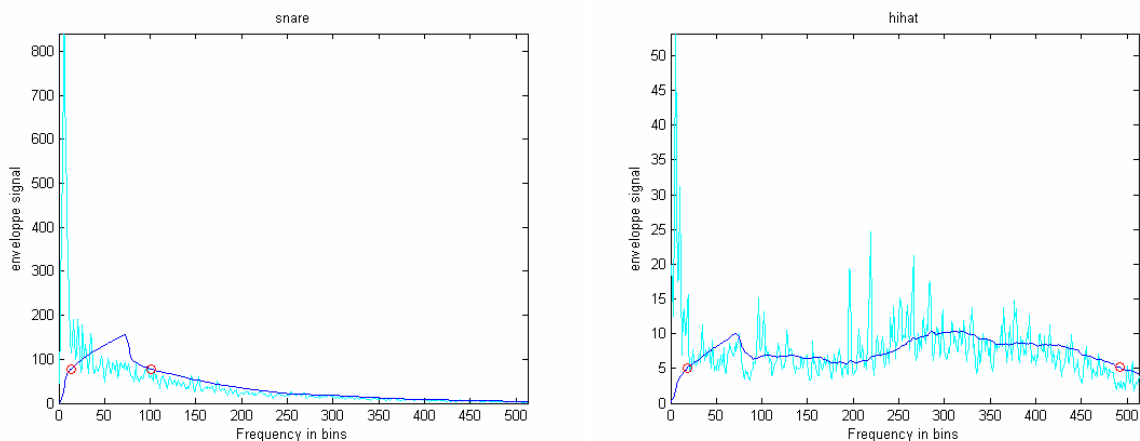


Abbildung 19: „Spectral Strong Peak Sample“-Signal eines Snare-Schlages (links) und eines Hihat-Schlages (rechts).

Eigenschaften Spectral Strong Peak Sample-Spektrogramm.

- Sampling Frequenz: 44100Hz
- FFT-Länge: 1024, entspricht einer Frequenzauflösung von 43Hz
- Fensterlänge (Hanning Fenster): 1024
- Überlappung: 936
- Frame-Länge: Fensterlänge – Überlappung = 88 Samples, entspricht 2ms
- Audiosignallänge ab Onsetzeitpunkt: 150ms

Die Berechnung erfolgt mittels Spektrogramm. Für die Ermittlung des Spektrum-Verlaufes wird das über alle Frames aufsummierte Amplitudenspektrum verwendet.

Bark Features

Die Bark Frequenzskala [8] ist eine psychoakustische Skala, die der Frequenzskala des Ohres (der Cochlea) nachempfunden ist. Die Eckpunkte der 24 Barkfrequenzbänder sind gegeben durch die folgenden Frequenzen in Hertz: 0, 100, 200, 300, 400, 510, 630, 770, 920, 1080, 1270, 1480, 1720, 2000, 2320, 2700, 3150, 3700, 4400, 5300, 6400, 7700, 9500, 12000, 15500. Für unsere Untersuchungen wurden die untersten zwei Bänder jeweils nochmals halbiert, so dass wir schlussendlich 26 Bark Bänder zur Verfügung haben.

Als Merkmalswerte werden die relativen Energien der Bänder, also die Summe der Energie aller Frames eines Frequenzbandes bezogen auf die Gesamtenergie aller Bänder, das Verhältnis der Energien tiefer Frequenzbänder zu mittleren Frequenzbänder sowie die relativen Standardabweichungen der Framewerte der einzelnen Bänder verwendet. Abbildung 20 zeigt den Energieverlauf der Gesamtenergie über alle Bänder (rote Linie) und den Energieverlauf des zweiten Bandes (blaue Linie) am Beispiel eines Pauken- und eines Ride-Schlages.

Die Bark Features eignen sich zur Erkennung von Paukenschlägen und der Unterscheidung verschiedener Metallinstrumente.

Eigenschaften Bark-Spektrogramm.

- Sampling Frequenz: 44100Hz
- Fensterlänge (Hanning Fenster): 512
- Überlappung: 256
- Frequenzvektor: siehe Text oben
- Frame-Länge: Fensterlänge – Überlappung = 256 Samples, entspricht 5.8ms
- Audiosignallänge ab Onsetzeitpunkt: 150ms

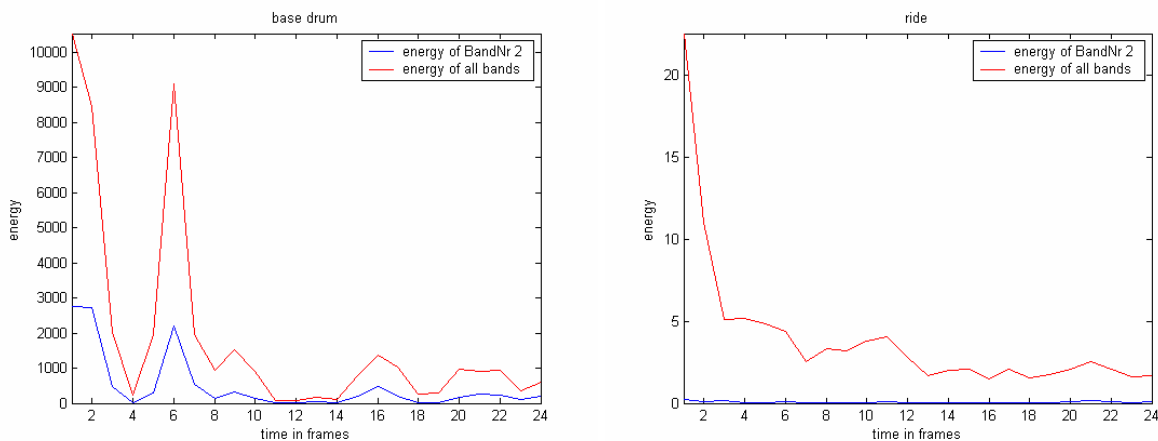


Abbildung 20: Energieverlauf eines Pauken-Schlages (links) und eines Ride-Schlages (rechts)

Mel Frequency Cepstral Coefficients (MFCC)

Die „Mel Frequency Cepstral Coefficients“ [7], wichtige Features in der Untersuchung von Sprachsignalen, zeigen sich auch nützlich für unsere Schlagzeugsignale. Nachfolgender Pseudocode zeigt die einzelnen Schritte zur Berechnung der MFCC's.

- Unterteilung des Audiosignals in Frames
- Berechnung der Diskreten Fourier Transformation (DFT) für jedes Frame
- Zusammenfassen zu Frequenzbänder gemäss der mel-Skala
- Berechnung des logarithmischen Leistungsspektrums
- Berechnung der Diskrete Cosinus Transformation der erhaltenen Werte

Schritt 1 und 2 entsprechen der Berechnung des Spektrogramms.

Schritt 3 dient der Anpassung an das nicht lineare Empfinden verschiedener Frequenzen des menschlichen Ohres. Die subjektive Tonhöhenempfindung nennt man Tonheit. Sie wird in Abbildung 22 illustriert (mel-Skala). Durch das Zusammenfassen in Frequenzbänder entstehen 13 MFCC-Merkmale.

Schritt 4 dient der Anpassung an die nichtlineare Lautstärkeempfindung des menschlichen Ohres.

Schritt 5 dekorreliert die Komponenten für den Feature Vektor.

Als Merkmalswert dient jeweils der Mittelwert der berechneten MFCCs über aller Frames (in Abbildung 21 rot gestrichelt: Mittelwert, blau: Verlauf des 5. MFCC Merkmals). Die MFCCs werden zur Erkennung von fast allen Instrumenten in irgendeiner Form verwendet.

Eigenschaften MFCC-Spektrogramm.

- Sampling Frequenz: 44100Hz
- FFT-Länge: 256, entspricht einer Frequenzauflösung von 172Hz
- Fensterlänge (Hamming Fenster): 256
- Überlappung: 156
- Frame-Länge: Fensterlänge – Überlappung = 100 Samples, entspricht 2.3ms
- Audiosignallänge ab Onsetzeitpunkt: 150ms

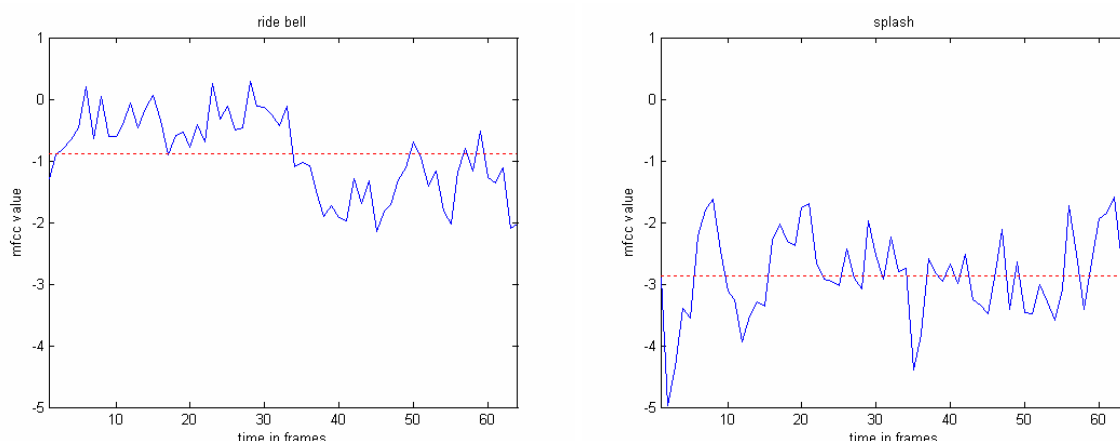


Abbildung 21: Verlauf der MFCC-Werte links eines Kuppel-Schlages und rechts eines Splash-Schlages.

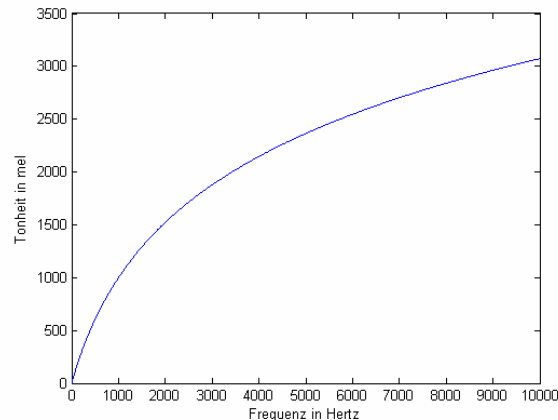


Abbildung 22: Mel-Frequenzskala

3.3.3 *InstrClassification* – *Instrumentenklassifikation*

Die Aufgabe der *InstrClassification* ist das richtige Zuordnen eines Testsignals zu einem Instrument. Als Eingangsdaten dienen die zuvor mit *FeatValue* berechneten Merkmalswerte. *InstrClassification* kann in folgende drei Bereiche unterteilt werden:

- Vorbereiten der Daten
- Hauptachsentransformation PCA
- Ermittlung des Instrumentes mit K-NN

Vorbereiten der Daten

Das System enthält für jedes Instrument vorgängig eingespielte Trainingsbeispiele. Diese Trainingsdaten, welche mit *FeatValue* berechnet wurden, werden zur Weiterverarbeitung mittelwertfrei gemacht und mit der Standardabweichung normiert. Auch das zu klassifizierende Testsignal wird mit dem Trainingsdaten-Mittelwert und der Trainingsdaten-Standardabweichung verarbeitet.

Hauptachsentransformation PCA

Ziel der Hauptachsentransformation [9] ist die Reduktion der Dimension des Merkmalraumes. Mit der Hauptachsentransformation können die Daten in einen neuen Merkmalsraum transformiert werden, in dem die Merkmale untereinander minimal korreliert sind. Aus den ursprünglichen Merkmalen entstehen neue Merkmale, von welchen nur noch die für die Klassifikation relevanten verwendet werden.

Ermittlung des Instrumentes mit K-NN

Das zu klassifizierende Testsignal wird somit zusammen mit den Trainingsdaten in den PCA-Merkmalraum transformiert. In diesem Raum werden nun die fünf dem Testsignal ähnlichsten Trainingspunkte bestimmt. Anhand dieser sogenannten nächsten Nachbarn kann entschieden werden, zu welcher Instrumentengruppe das Testsignal gehört. Es wird der Instrumentengruppe, welche in den fünf Nachbarn am häufigsten vertreten ist, zugewiesen. Ist keine häufigste Instrumentengruppe auszumachen, gibt der am nächsten liegende Trainingspunkt den Ausschlag [10].

3.3.4 Merkmalsräume

Der Feature Algorithmus kann 22 Instrumente und Instrumentenkombinationen erkennen. Diese sind nachfolgend aufgelistet:

- Crash
- Crash&Pauke
- Crash&Snare
- Kuppel
- Kuppel&Pauke
- Kuppel&Snare
- Ride
- Ride&Pauke
- Ride&Snare
- Splash
- Splash&Pauke
- Splash&Snare
- Hihat closed
- Hihat half
- Pauke&Hihat
- Snare&Hihat
- Pauke
- Tom1
- Tom2
- Tom3
- Snare
- Snare&Pauke

Für die Klassifikation dieser Instrumentenarten reicht ein Merkmalsraum nicht aus. Der Algorithmus verwendet deshalb insgesamt 6 verschiedene Merkmalsräume. In allen Räumen geschieht die Zuweisung wie in den Kapiteln 3.3.2 und 3.3.3 beschrieben.

Abbildung 23 gibt einen Überblick über die verwendeten Merkmalsräume und deren Zusammenwirken.

Die unten aufgeführten Merkmalsräume werden nachfolgend genauer erklärt:

- Hihat
- PaukeTomsMono
- PaukeSnareMetallinstrumente
- PaukeMetallinstrumente
- MetallinstrumenteMono
- SnareMetallinstrumente

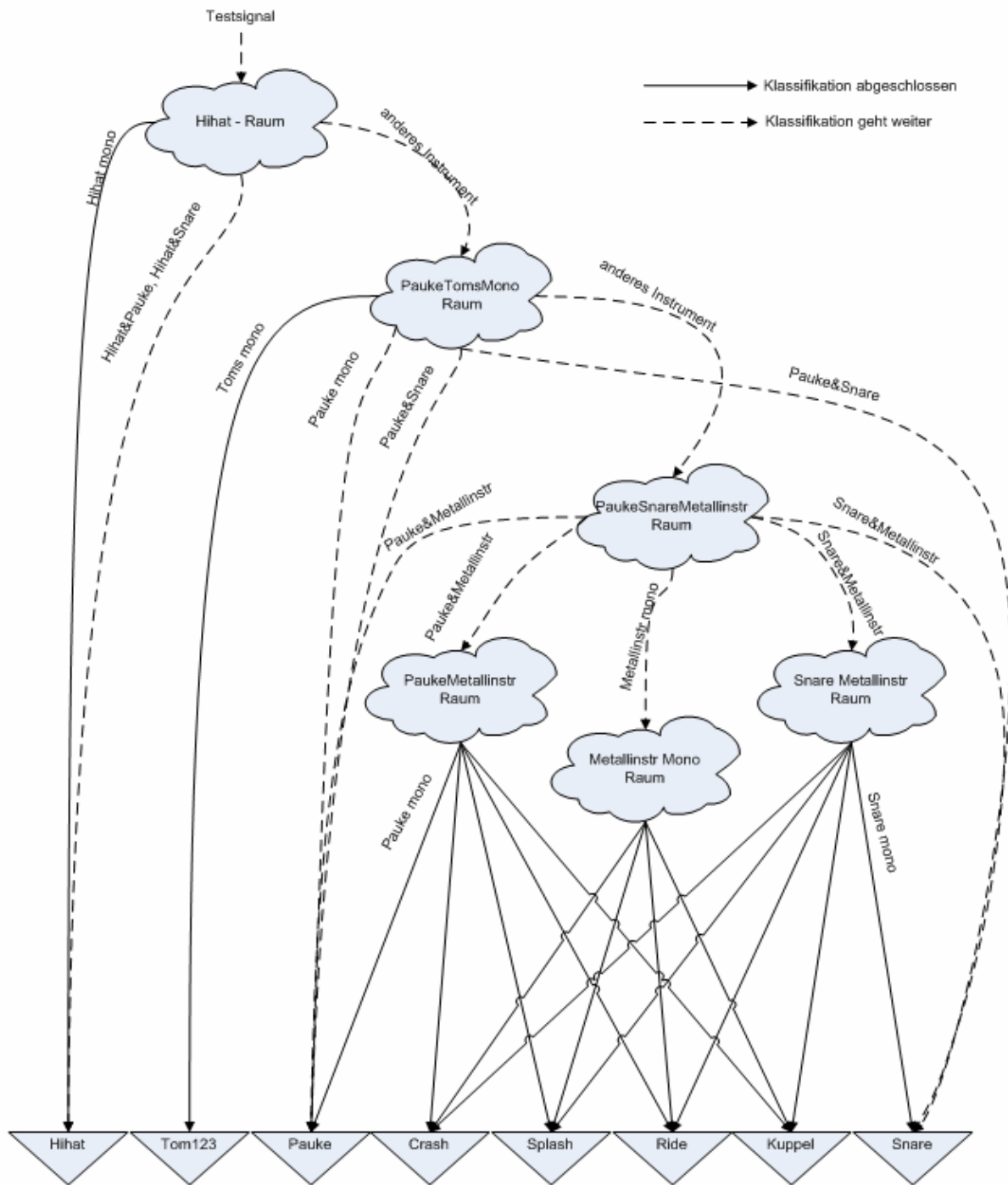


Abbildung 23: Übersicht der Merkmalsräume

Merkmalsraum Hihat

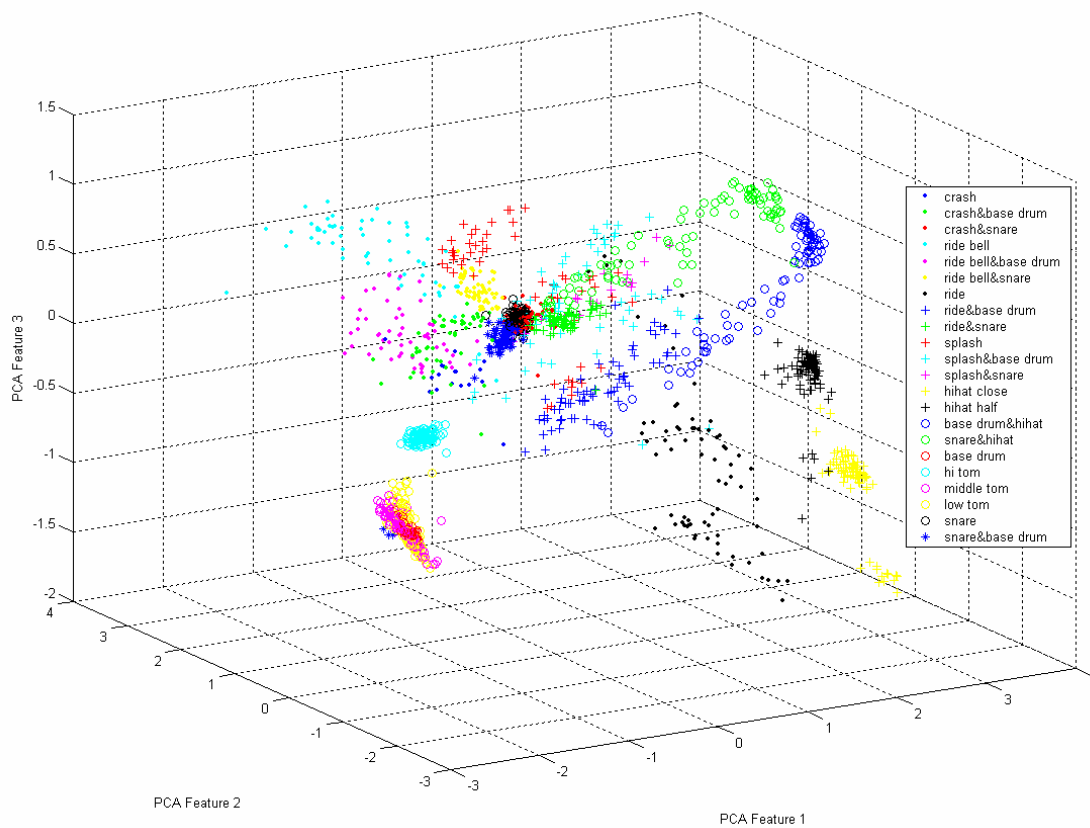


Abbildung 24: Hihat-Merkmalsraum

Der Hihat-Raum dient ausschliesslich der Erkennung von Hihat-Schlägen. Dabei wird unterschieden in „Hihat mono“ (Open oder Closed), „Hihat mit Pauke“ und „Hihat mit Snare“. Wurde das Testsignal als Hihat mono erkannt, ist die Klassifikation bereits abgeschlossen.

Behandelte Instrumente

alle

Verwendete Features

Spectral Rolloff, Spectral Flatness, Spectral Strong Peak Sample, MFCC 3

Dimension des Merkmalsraums

4

Merkmalsraum PaukeTomsMono

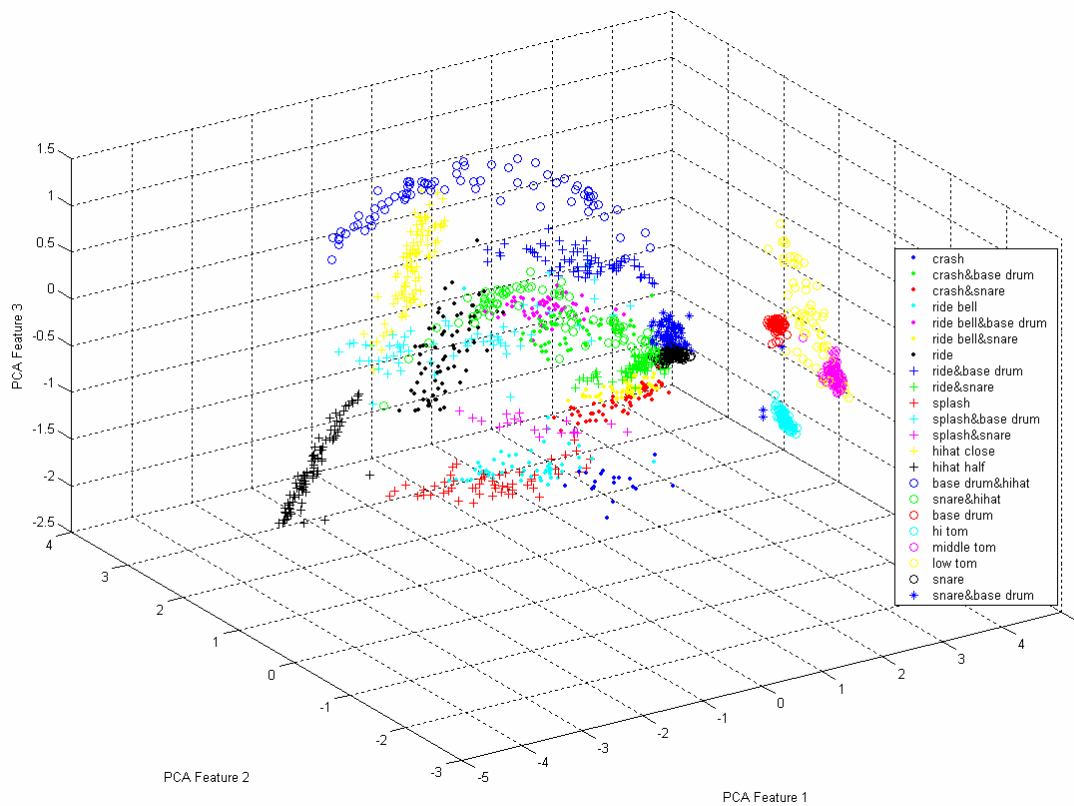


Abbildung 25: PaukeTomsMono-Merkmalsraum

Der PaukeTomsMono-Raum dient der Erkennung von Pauke mono und Toms - Schlägen. Rechts im Raum von Abbildung 25 ist sehr schön zu sehen, wie die vier Clusters von den anderen Daten abgetrennt sind.

Behandelte Instrumente

alle

Verwendete Features

ZCR, Spectral Centroid, Spectral Rolloff, MFCC 2, Bark relEnergy 2, Bark relStd 2

Dimension des Merkmalsraums

5

Merkmalsraum PaukeSnareMetallinstrumente

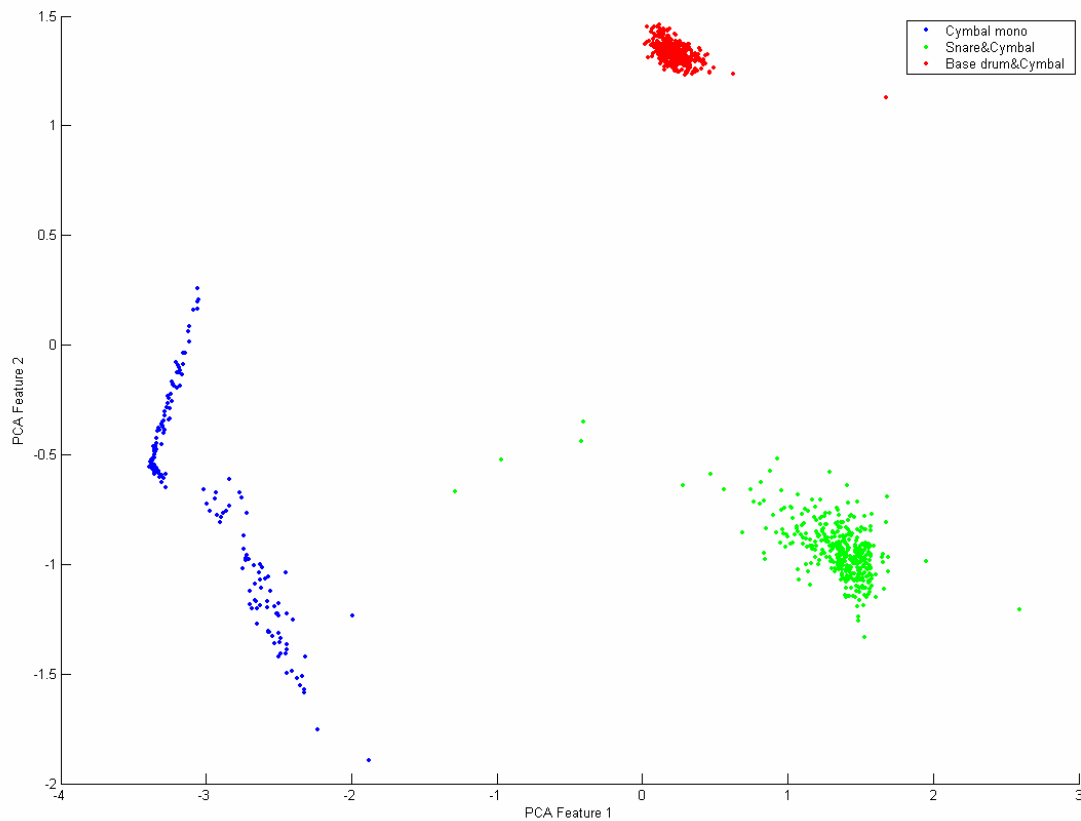


Abbildung 26: PaukeSnareMetallinstrumente-Merkmalsraum

Der PaukeSnareMetallinstrumente-Raum dient der Unterscheidung in die drei Gruppen: „Metallinstrumente Mono“, „Snare mit Metallinstrumenten“ und „Pauke mit Metallinstrumenten“.

Behandelte Instrumente

Alle ausser Hihat mono und Toms

Verwendete Features

Bark rel Energy 2, 3, Bark relStd 3

Dimension des Merkmalsraums

3

Merkmalsraum PaukeMetallinstrumente

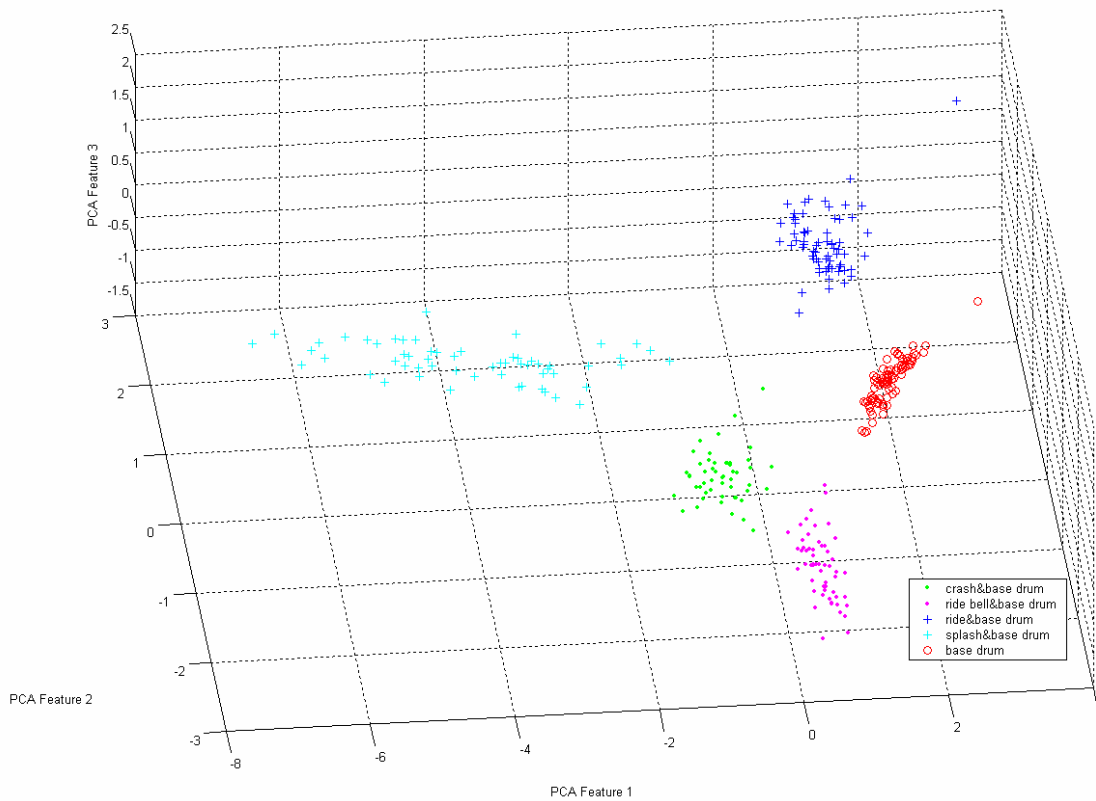


Abbildung 27: PaukeMetallinstrumente-Merkmalsraum

Der PaukeMetallinstrumente-Raum dient der Erkennung der verschiedenen Metallinstrumente, welche zusammen mit der Pauke angeschlagen wurden. Auch Pauke mono kann hier klassifiziert werden.

Behandelte Instrumente

Crash&Pauke, Kuppel&Pauke, Ride&Pauke, Splash&Pauke, Pauke mono

Verwendete Features

ZCR, MFCC 2, 3, 5, 6, 9, Bark rel Energy 10, Bark Ratio

Dimension des Merkmalsraums

5

Merkmalsraum MetallinstrumenteMono

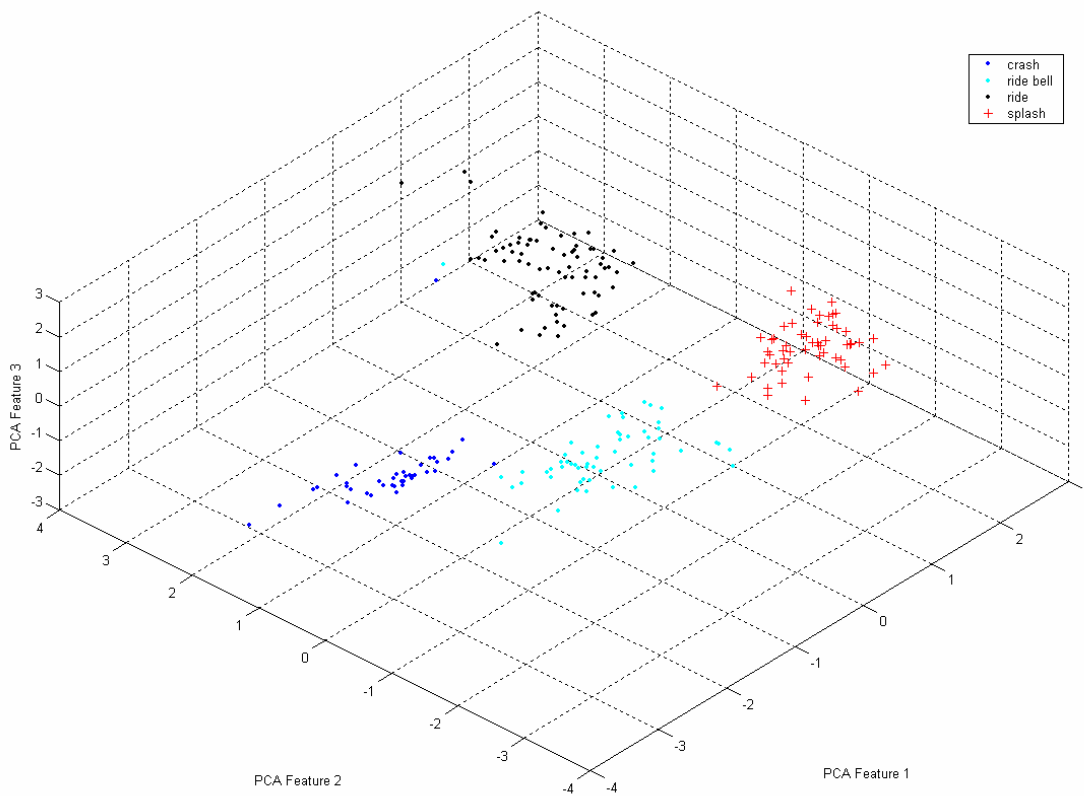


Abbildung 28: MetallinstrumenteMono-Merkmalsraum

Der MetallinstrumenteMono-Raum klassifiziert die einzelnen mono Schläge der Metallinstrumente.

Behandelte Instrumente

Crash, Kuppel, Ride, Splash

Verwendete Features

ZCR, MFCC 3,5, Bark relEnergy 7, 11, 15, Bark Ratio

Dimension des Merkmalsraums

4

Merkmalsraum SnareMetallinstrumente

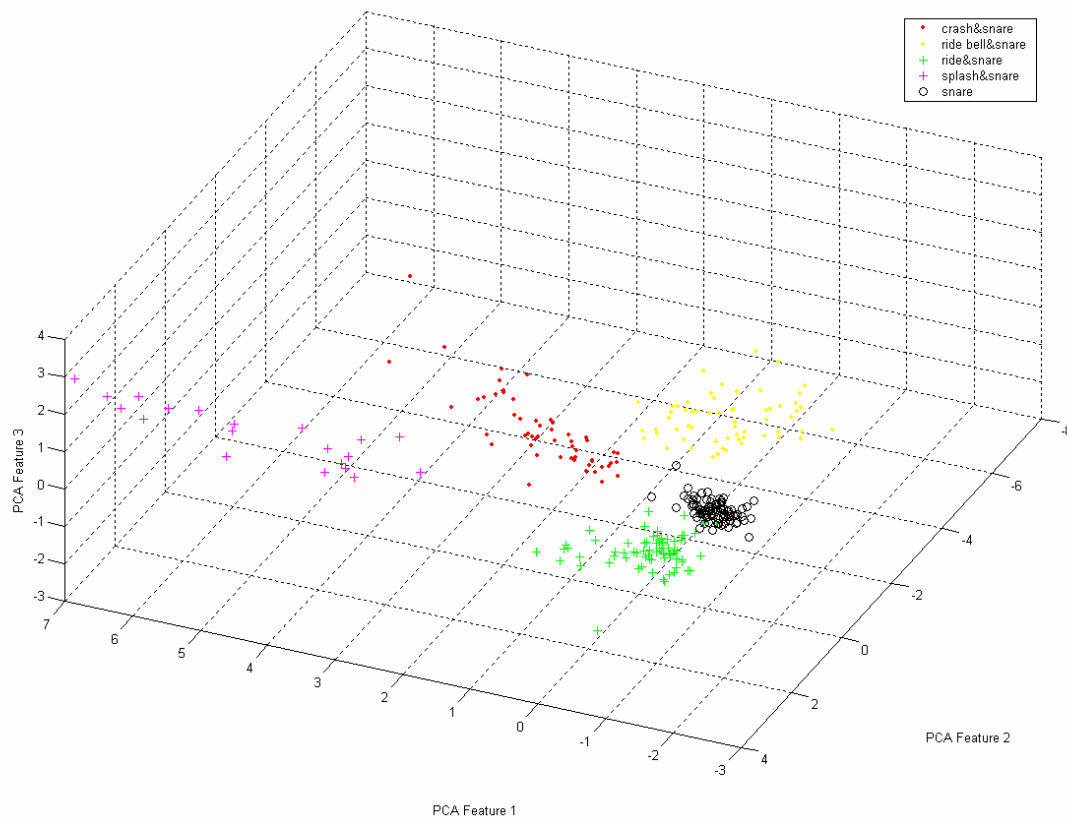


Abbildung 29: SnareMetallinstrumente-Merkmalsraum

Der SnareMetallinstrumente-Raum dient der Erkennung der verschiedenen Metallinstrumente, welche zusammen mit der Snare angeschlagen wurden. Auch Snare mono wird in diesem Raum klassifiziert.

Behandelte Instrumente

Crash&Snare, Kuppel&Snare, Ride&Snare, Splash&Snare, Snare mono

Verwendete Features

ZCR, MFCC 3, 5, 6, Bark relEnergy 7, 8, 10

Dimension des Merkmalsraums

5

3.4 Vereinigung von Matching- und Featureresultaten

Aus den Resultaten der beiden Klassifizierungsalgorithmen muss ein gemeinsames Ergebnis entstehen. Abbildung 30 zeigt anhand eines Schemas wie die beiden Resultate verknüpft sind.

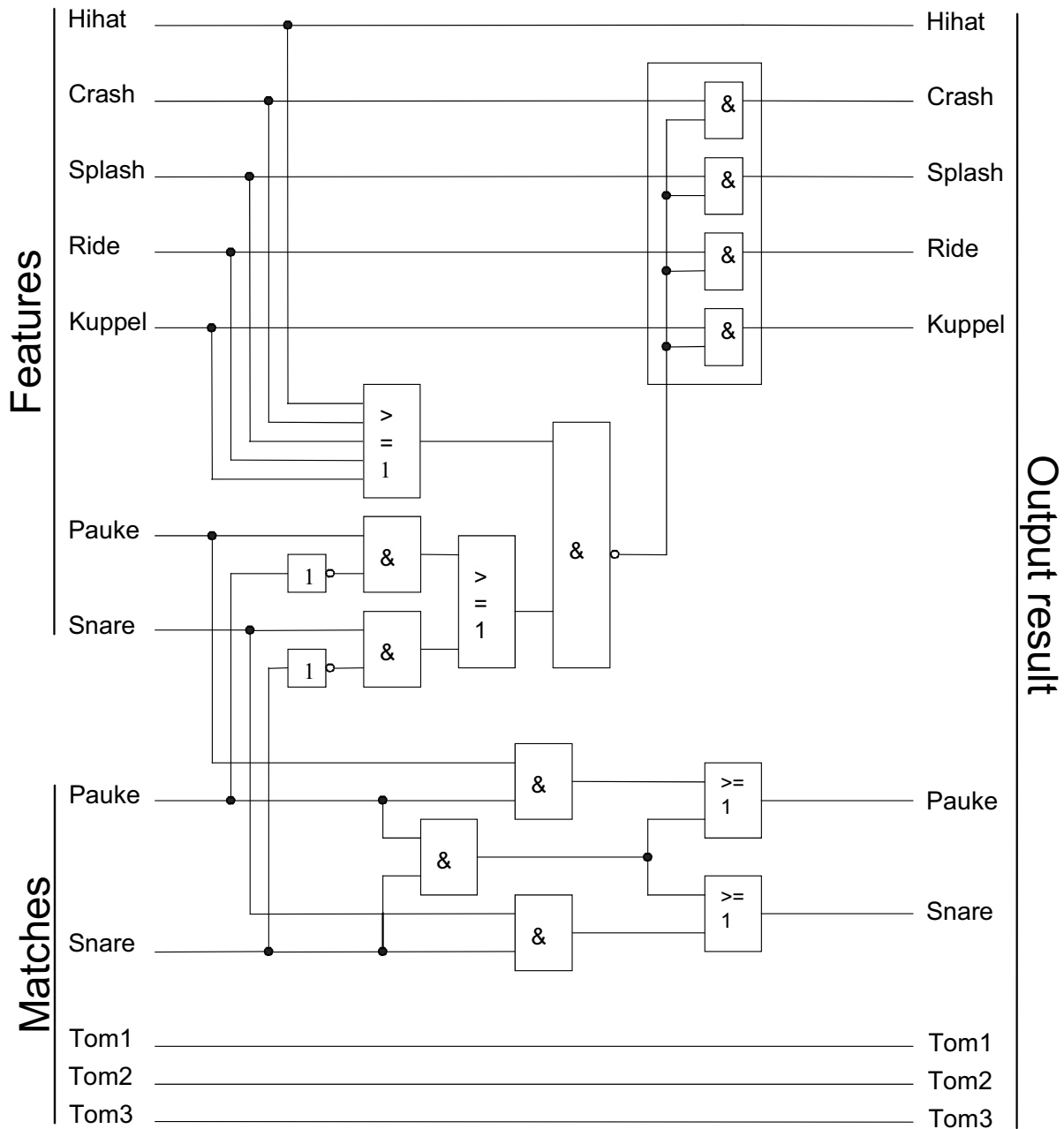


Abbildung 30: Die logischen Verknüpfungen zwischen den beiden Methoden

Pauke, Snare

Für die Erkennung der Pauken- und Snareschläge würde die Matchingmethode alleine nicht genügen, da sie Snareschläge erkennt, die gar nicht vorhanden sind. Sie stammen meistens von einem Tom3 Schlag, der Grund ist unter Kapitel 3.2.4 näher erläutert. Die Featuremethode arbeitet diesbezüglich zuverlässiger. Dafür führt bei ihr das Nachklingen eines Snareschlages zur mehrfach Klassifikation. Diese doppelten oder sogar dreifachen Snareschläge verändern den Klang des Gespielten erheblich. Da die Matchingmethode diesen Fehler nicht hat, verknüpfen wir die beiden Resultate mit einer UND-Funktion.

Das Analoge gilt für die Pauke.

Das Erkennen von gleichzeitiger Pauke und Snare funktioniert beim Matching sehr zuverlässig, und hat deshalb direkten Einfluss auf den Output.

Toms

Die Erkennungsleistung des Matching Algorithmus für Tomschläge ist zuverlässig. Die Featuremethode kann nur monophone Tom Schläge zuverlässig erkennen, was aber in der Realität selten vorkommt. Aus diesen Gründen werden die Tom Schläge direkt vom Matching übernommen.

Metallinstrumente

Das Hihat wird von den Features gut erkannt.

Gewisse Metallinstrumente klingen lange aus. Wenn z.B. nach einem Crashschlag ein Tom erklingt, so wird fälschlicherweise ein zweiter Crashschlag mit einer Snare oder Pauke identifiziert. Dies liegt an der Überlagerung von Tom- und Crashklang. Das Feature-System kennt Snare mit Metallinstrumenten oder Pauke mit Metallinstrumenten, nicht aber Toms mit nachklingenden Metallinstrumenten. Deshalb werden oben genannte Schläge entweder als Snare mit Metallinstrument oder Pauke mit Metallinstrument klassifiziert. Dieser Fehler wird mit der implementierten Logik in Abbildung 30 korrigiert. Schläge von Metallinstrumenten mit Snare oder Pauke von den Features werden nur akzeptiert, wenn beim Matching die Pauke oder die Snare auch erkannt wurde. Mit dieser Verknüpfung lassen sich Fehler eliminieren, die Performance steigt.

Onsets

Die Onsetzeiten der Matches sind nicht exakt die gleichen wie diejenigen der Features. Die Feature-Onsetzeiten enthalten (auch wenn nicht mit dem exakt gleichen Wert) aber diejenigen der Matches. Deshalb wird für das Erstellen der MIDI-Datei die Menge der Feature-Onsetzeiten gewählt.

4 Beschrieb der Software

4.1 Übersicht

Das Drum2Midi-System besteht aus den Teilen Onset Detection, Template Matching, Features und der Vereinigung von Matching- und Featureresultaten. Abbildung 31 gibt einen Überblick über die verwendeten Algorithmen. Diese sind in den Kapiteln 3.1-3.4 erklärt.

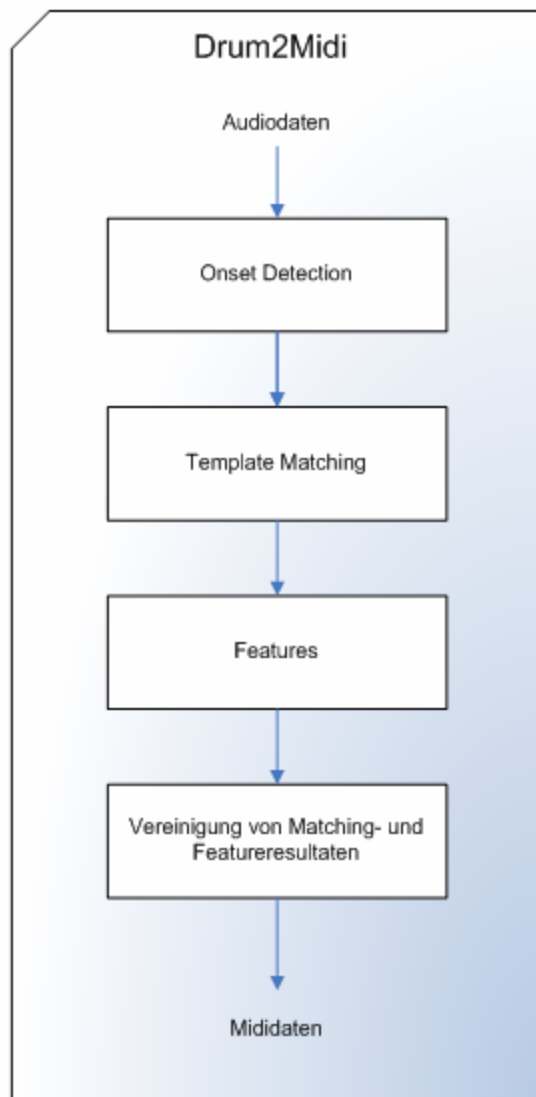


Abbildung 31: Übersicht über den Ablauf und die verwendeten Algorithmen

4.2 Betriebsmodi

4.2.1 Live2Midi

Der Live2Midi-Modus erlaubt es von einem live gespielten Schlagzeugstück die dazugehörige Midi-Datei zu erstellen. Abbildung 32 zeigt den dazugehörigen Ablauf.

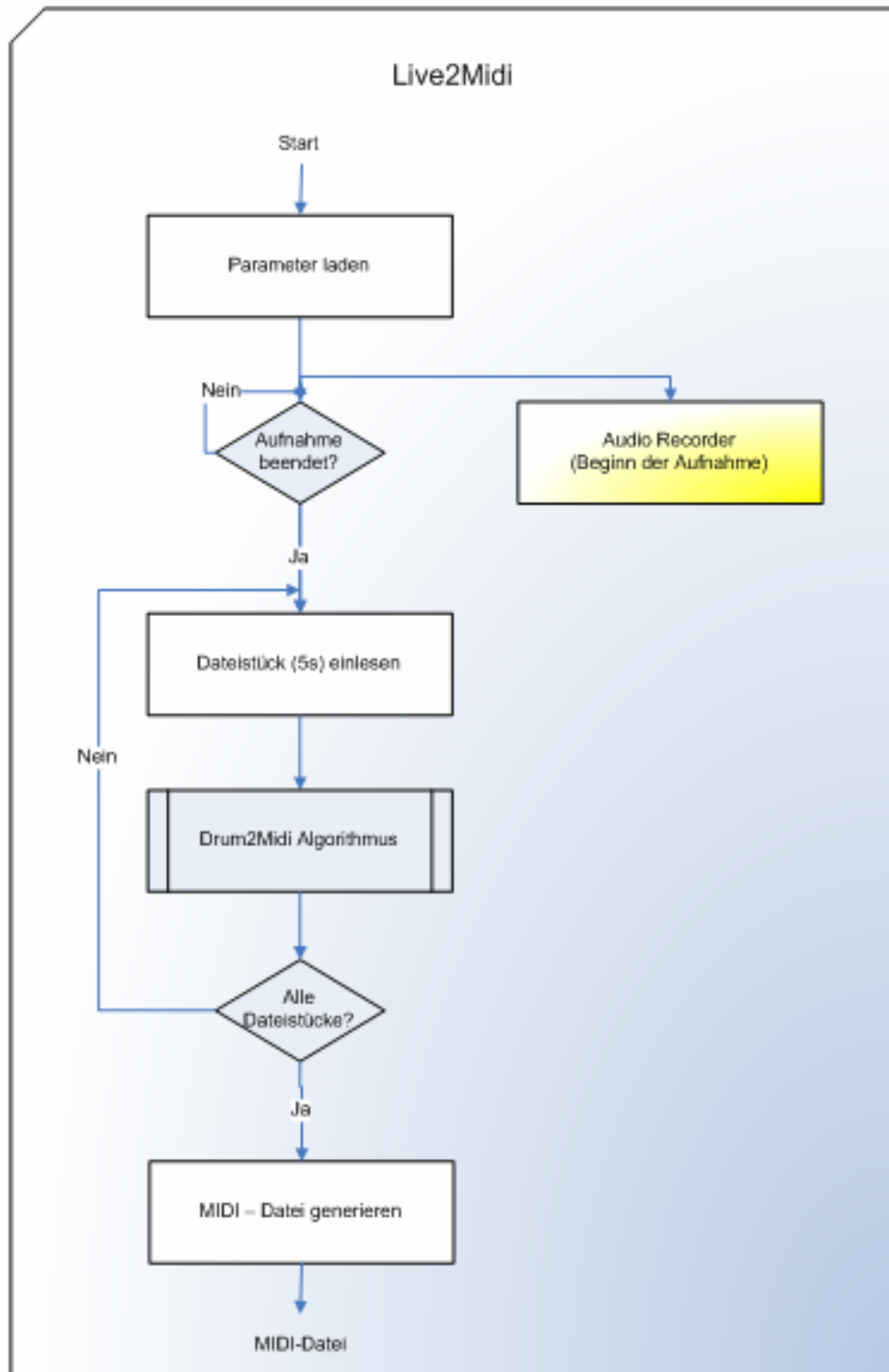


Abbildung 32: Ablauf des Live2Midi-Betriebsmodus

Parameter laden

Es werden die für die Algorithmen benötigten Parameter und Daten geladen.

Audio Recorder

Der Audio Recorder ist eine eigenständige Applikation zum Aufnehmen von Musikstücken. Er speichert die Daten der Soundkarte periodisch alle 5 Sekunden als Wav-Dateien ab. Der Live2Midi Algorithmus startet den Audio Recorder und wartet solange bis der Recorder und somit die Aufnahme durch Betätigen des „Stop Recording“-Buttons beendet wird. Der Audio Recorder ist in Anhang A noch genauer erklärt.

Dateistück einlesen

Die vom Audio Recorder erzeugten Wav-Dateien müssen nun der Reihe nach abgearbeitet werden. Dabei kann es vorkommen, dass kurz vor Ende eines 5s-Audiostückes ein Onset eintrifft. Dieser Schlag wird somit vom Audio Recorder in zwei Teile getrennt. Ein erster Teil des Schlages am Ende eines 5s-Audiostückes und der zweite Teil des Schlages gerade am Anfang des nachfolgenden 5s-Audiostückes. Werden nun die beiden Stücke unabhängig voneinander mit dem Drum2Midi-Algorithmus bearbeitet, können Fehlklassifikationen und falsche Onsetzeiten auftreten, da der Schlag nicht als Ganzes bearbeitet wurde. Aus diesem Grunde werden die Audiostücke überlappend verarbeitet.

Abbildung 33 zeigt ein Audiosignal als Ganzes (oben) und in die überlappenden Teilstücke unterteilt (unten).

Drum2Midi Algorithmus

Der Drum2Midi Algorithmus klassifiziert die Audiostücke. Abbildung 31 gibt einen Überblick über die verwendeten Algorithmen, welche im Kapitel 3 erläutert sind. Als Resultat liefert der Drum2Midi Algorithmus die für die Erstellung der MIDI-Datei relevanten Daten. Dies sind einerseits die Onsetzeiten und andererseits die zu diesen Zeitpunkten gespielten Instrumente.

MIDI-Datei generieren

Damit eine MIDI-Datei erzeugt werden kann, muss die Information in Noten bereitstehen. Die Information gewinnt man aus den Onsets und den dazugehörigen Instrumentenschlägen. Die Onsets sind Zeiten, welche aus Sekunden und Bruchteile davon bestehen. Der Schritt von der bruchteilbehafteten Zeit in Sekunden zu der Zeit in ganzen Noten geschieht wie folgt:

- Alle Zeitintervalle zwischen aufeinanderfolgenden Onsets bestimmen. Dies ergibt die Interonsetintervalle (IOI).
- Eine Zeit derart bestimmen, dass sie ein ganzzahliger Bruchteil eines jeden IOI ist. Dies kann nur innerhalb gewissen Toleranzen geschehen. Diese Zeit entspricht der kürzesten Notendauer und ist für die Tempobestimmung massgebend.
- Jedes Interonsetintervall durch die kürzeste Notendauer dividieren und auf eine ganze Zahl runden. Man erhält die relativen Onsetzeiten in Noten.
- Das Aufsummieren der relativen Onsetzeiten in Noten ergibt die gesuchte Onsetzeit in Noten.

Jetzt gilt es eine Matrix aufzubauen, welche die MIDI-Information wie folgt enthält:

Onsetzeit in Noten	Instrument	Dauer	Lautstärke	MIDI-Kanal
0	32 (Pauke)	Immer 1	Immer 64	Immer 10
1	38 (Snare)	1	64	10
...	...	1	64	10

Diese Matrix enthält die nötigen Informationen um eine MIDI-Datei zu generieren

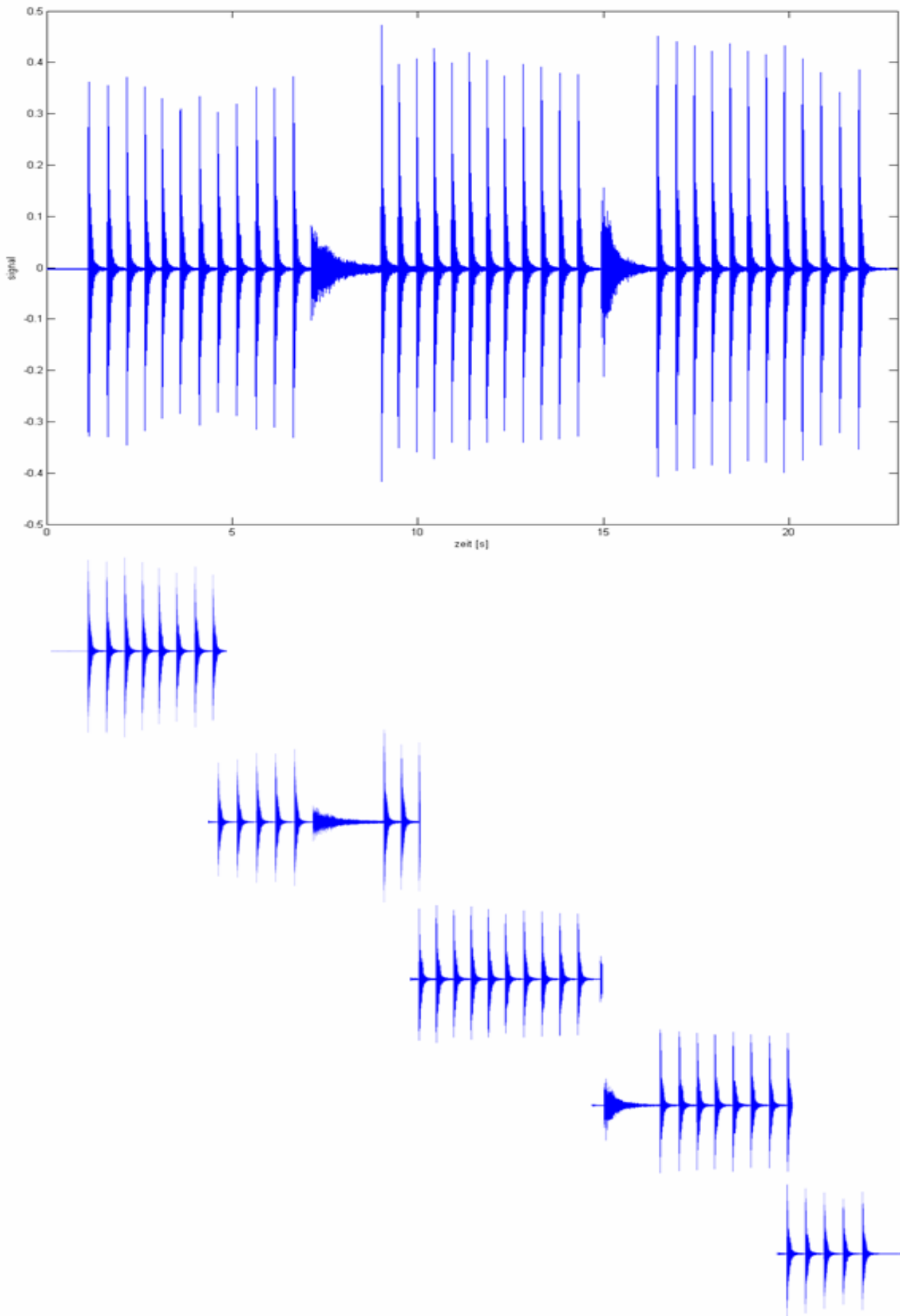


Abbildung 33: Visualisierung der 5s-Audiostücke

Eine MIDI-Datei ist in Brocken (engl. chunks) organisiert [12]. Die ersten acht Bytes eines jeden Chunks sind fix und beschreiben den Chunk-Typ und die Länge des Chunks. Der erste Brocken in einer MIDI-Datei ist der Header Chunk. In ihm sind MIDI-Format, Anzahl Tracks und Tempoinformationen enthalten. Auf den Header Chunk folgen die Track Chunks. Jedes Instrument (Schlagzeug, Klavier, Flöte, ...) hat seinen eigenen Track Chunk. Wir benötigen lediglich einen. Jeder Track Chunk beginnt wiederum mit vier Bytes Chunk Identifikationsdaten und vier Bytes Längenangabe. Danach folgen die MIDI-Events. Mit ihnen werden die Noten ein- (NoteOn Event) oder ausgeschaltet (NoteOff Event). Am Schluss eines jeden Track Chunks steht das EndOfTrack Event, welches den Track Chunk abschliesst.

Das Programm verwendet die MIDI-Toolbox [11]. Mit ihr lassen sich die MIDI-Dateien aus obiger Matrix schreiben. Es müssen lediglich die MIDI-Events „SetTempo“ und „EndOfTrack“ zusätzlich eingebaut werden.

4.2.2 *Wav2Midi*

Der Wav2Midi-Modus ermöglicht das Verarbeiten von zuvor aufgenommenen Wav-Dateien. Die Wav-Datei wird nicht als Ganzes verarbeitet, sondern wie beim Live2Midi Modus in 5s-Stücke unterteilt. Der Ablauf entspricht dem Live2Midi Algorithmus (siehe Abbildung 32), jedoch ohne Aufruf des Audio Recorders.

4.2.3 *DemoMode*

Der DemoMode verleiht dem Benutzer Einblick in die Verarbeitung der Audiodaten. Zu jedem wichtigen Schritt werden die Daten visualisiert und mit einem Infotext kommentiert. Als Beispiele dienen Demosamples von verschiedenen Instrumenten, mit welchen anschaulich die Funktionsweise des Systems getestet werden kann.

4.3 Leistungsfähigkeit

4.3.1 Instrumente

Das System ist für folgende Mono- respektive Polyschläge ausgelegt

Mono Schläge	Poly Schläge
Snare	Snare-Pauke
Pauke	
Tom1	Snare-Tom3
Tom2	Snare-Hihat
Tom3	Snare-Crash
Hihat (closed oder half)	Snare-Ride
Crash	Snare-Splash
Ride	Snare-Kuppel
Splash	
Kuppel	Pauke-Hihat
	Pauke-Crash
	Pauke-Ride
	Pauke-Splash
	Pauke-Kuppel

Es ist zu ergänzen, dass im Prinzip alle möglichen Kombinationen der Fellinstrumente (Pauke, Snare, Tom1, Tom2, Tom3) untereinander zu Erkennen möglich sind. In der Tabelle oben sind nur die für uns zentralen Kombinationen aufgeführt.

4.3.2 Geschwindigkeit

In der Musik ist es üblich, das Tempo in Schlägen pro Minute anzugeben. Als Referenz dient die Viertelnote. Das System funktioniert zuverlässig für Geschwindigkeiten bis 320 Schlägen pro Minute für die Snare und das Hihat. Ein Wirbel ist zu schnell um korrekt erkannt zu werden. Je mehr ein Instrument nachklingt, desto tiefer liegt der Wert.

4.3.3 Dynamik

Der Begriff Dynamik bezeichnet die Differenz zwischen leise- und laut gespielten Schlägen. Dynamik ist wichtig, damit ein Schlagzeugspiel lebendig wirkt und nicht als maschinellen Lärm daherkommt. Je dynamischer gespielt wird, umso schwieriger ist es das Gespielte zu erkennen. Technische Angaben über die Dynamik für verschiedene Instrumente sind schwierig zu machen. Deshalb verweisen wir hier auf die Testaufnahmen welche auf der beiliegenden CD im Ordner „Test Files“ zu finden sind.

5 Bedienungsanleitung

5.1 Systemanforderungen

Die Computer auf denen das System entwickelt wurde, haben folgende Eigenschaften:

System

Microsoft Windows XP Professional

Version 2002

Service Pack 1

Hardware

Pentium 4 CPU 2.4GHz

1 GB RAM

SoundMAX Integrated Digital Audio (Onboard-Soundkarte)

Mikrofon

Stage Line ECM 40

Frequenzbereich 30–18000Hz

Die oben genannten Eigenschaften sollten aus Geschwindigkeitsgründen nicht wesentlich unterschritten werden. Das System läuft zwar auch auf weniger leistungsfähigen Maschinen, nur sind grössere Berechnungszeiten zu erwarten.

5.2 Installation

- Kopieren Sie zur Installation des Drum2Midi-Systems die Drum2MidiSetup.exe-Datei von der beiliegenden CD in das gewünschte Verzeichnis.
- Führen Sie Drum2MidiSetup aus. Es werden alle benötigten Dateien in das Verzeichnis entpackt.
- Starten Sie das Programm mit drum2midi.exe

5.3 Die Benutzeroberfläche

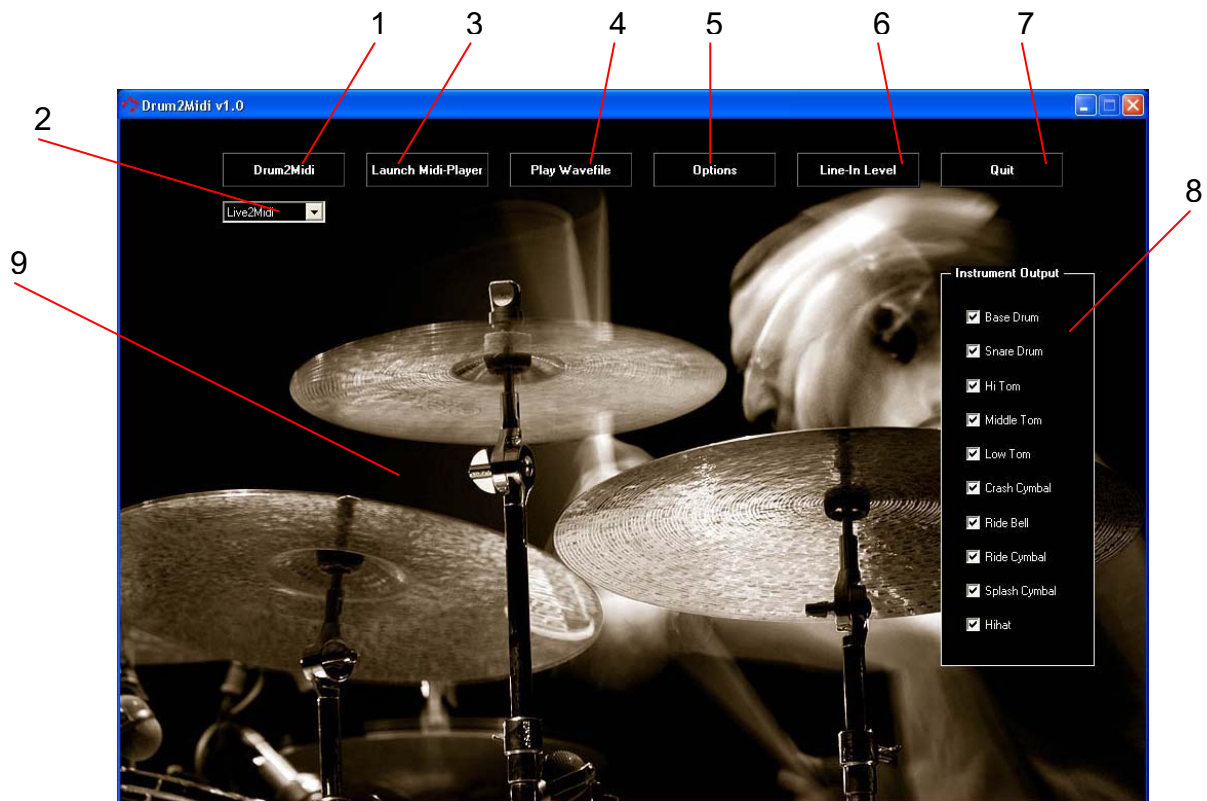


Abbildung 34: Benutzeroberfläche des Drum2Midi Programms

- 1 Drum2Midi starten
- 2 Betriebsmodus wählen
- 3 Starten des Midi-Players
- 4 Starten des Wav-Players
- 5 Einstellungen
- 6 Line-IN Level einstellen
- 7 Beenden
- 8 Instrumenten Output
- 9 Demonstrationsfläche

1 Drum2Midi starten

Startet den unter 2 eingestellten Betriebsmodus

2 Betriebsmodus wählen

Folgende Betriebsmodi stehen zur Auswahl:

Live2Midi

Dieser Modus erlaubt das Aufnehmen von Schlagzeugmusik und anschliessendem Klassifizieren. Zur Aufnahme wird automatisch die Applikation „Audio Recorder“ gestartet (siehe Anhang A). Bei Betätigen des „Stop Record“-Buttons wird die Aufnahme beendet und der Klassifikations-Algorithmus unmittelbar danach gestartet.

Wav2Midi

Dieser Modus erlaubt das Klassifizieren von zuvor aufgenommenen Wav-Dateien. Bei Betätigen der „Drum2Midi“-Taste wird ein Dialog-Fenster zum auswählen einer Wav-Datei aufgerufen. Wurde diese ausgewählt und mit OK bestätigt, startet der Klassifikations-Algorithmus. Die resultierende MIDI-Datei wird unter gleichem Namen und gleichem Ort wie die Wav-Datei abgespeichert.

DemoMode

Bei diesem Modus werden die einzelnen Schritte des Algorithmus anschaulich dargestellt. Der Benutzer hat die Möglichkeit aus einer Auswahl von verschiedenen Instrumentbeispielen auszuwählen. Mit den Buttons „Next“ und „Back“ kann zwischen den einzelnen Schritten hin- und her gewechselt werden. Ein Infotext erklärt die einzelnen Algorithmsgschritte.

3 Starten des Midi-Players

Nach der Klassifikation eines Schlagzeugstückes kann die MIDI-Datei abgespielt werden. Der Player kann unter 5 ausgewählt werden.

4 Starten des Wav-Players

Nach der Klassifikation eines Schlagzeugstückes kann die Wav-Datei zur Kontrolle abgespielt werden. Der Player kann unter 5 ausgewählt werden.

5 Einstellungen

Die Einstellungen werden unter Kapitel 5.3.1 erläutert.

6 Line-IN Level einstellen

Mithilfe des Windows-Record-Mixers kann der Line-IN Level eingestellt werden. Der Ausschlag des Signals kann anhand einer Balkenanzeige auf Übersteuern überprüft werden.

7 Beenden

Schliesst das Drum2Midi Programm und speichert die verwendeten Einstellungen. Beim nächsten Aufruf des Programms werden die zuletzt verwendeten Einstellungen wieder geladen.

8 Instrumenten Output

Mithilfe der Checkboxen lassen sich gezielt Instrumente sperren. D.h. sie kommen im Outputresultat (MIDI-Datei) nicht vor.

9 Demonstrationsflächen

In diesem Bereich wird einerseits ein Balkendiagramm zum Einstellen des Line-IN Levels dargestellt, andererseits wird dieser Bereich beim DemoMode zur Visualisierung der Algorithmen verwendet.

5.3.1 Einstellungen

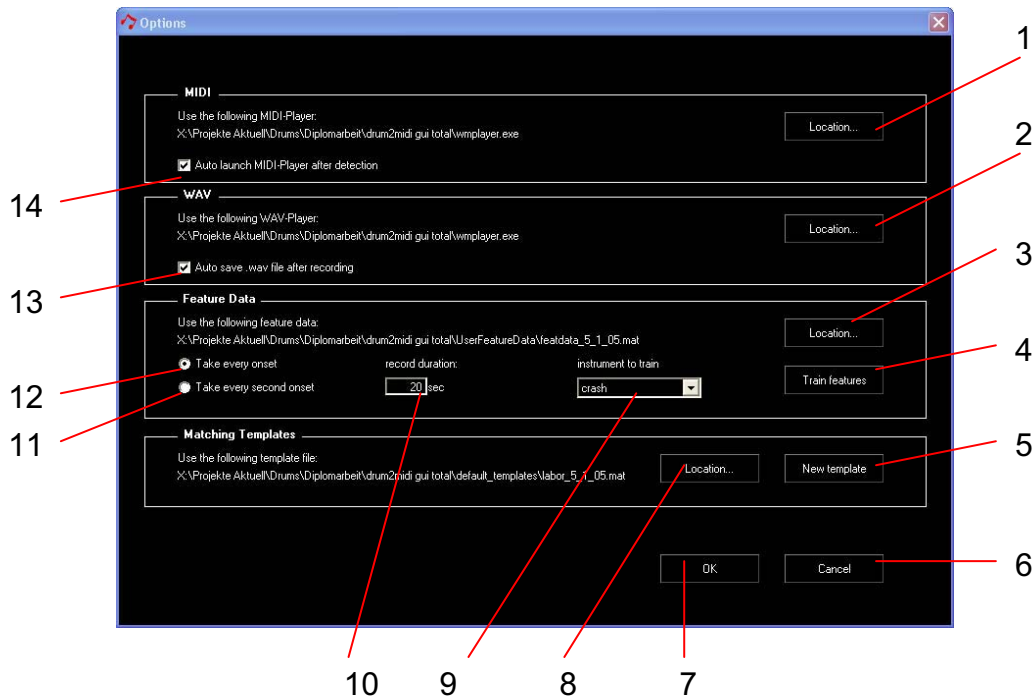


Abbildung 35: Dialogfenster für die Einstellungen

1 Pfad Midi-Player

2 Pfad Wav-Player

3 Pfad Feature Daten

4 Trainieren der Feature Daten

5 Erstellen neuer Templates

6 Abbrechen

7 OK

8 Pfad Matching-Templates

9 Zu trainierendes Instrument

10 Aufnahmedauer

11 Trainingsoption

12 Trainingsoption

13 Automatische Speicherung der Wav-Datei

14 Automatisches Starten des Midi-Players

1 Pfad Midi-Player

Auswahl eines geeigneten Midi-Players.

2 Pfad Wav-Player

Auswahl eines gewünschten Wav-Players.

3 Pfad Feature Daten

Auswahl der verwendeten Feature-Klassifikations Daten.

4 Trainieren der Feature Daten

Trainieren der Feature-Trainings Daten. Beachten Sie die Einstellungen bei 9, 10, 11, 12. Der Trainingsablauf ist sehr simpel und im Prinzip selbsterklärend. Trainingssamples können vor der Übernahme ins System überprüft und gegebenenfalls verworfen werden. Die gewünschten Trainingssamples werden automatisch den unter 3 eingestellten Feature-Klassifikations Daten angehängt. Sind die Default-Daten ausgewählt, werden Sie aufgefordert Ihre Trainingssamples unter anderem Namen abzuspeichern.

5 Erstellen neuer Templates

Erstellen neuer Templates für den Template Matching Algorithmus. Der Ablauf für das Erstellen dieser Templates ist gut kommentiert und im Prinzip selbsterklärend. Es müssen der Reihe nach acht Schläge der Instrumente Pauke, Snare, Tom1, Tom2 und Tom3 gespielt werden. Haben Sie sich vertan und beispielsweise neun Schläge gespielt, so erkennt dies die Software und gibt Ihnen eine neue Möglichkeit die acht Schläge einzuspielen. Der Vorgang kann jederzeit mit „Cancel“ abgebrochen werden.

6 Abbrechen

Verlassen des Einstellungen-Dialogfensters und verwerfen der geänderten Einstellungen.

7 OK

Bestätigen und verlassen des Einstellungen-Dialogfensters. Vorgenommene Einstellungen werden gespeichert.

8 Pfad Matching-Templates

Auswahl der verwendeten Matching-Templates.

9 Zu trainierendes Instrument

Auswahl der Instrumentenkombination zum Trainieren der Feature Klassifikations Daten.

10 Aufnahmedauer

Einstellen der Aufnahmedauer für das Trainieren der Feature Klassifikations Daten.

11 Trainingsoption

Nimmt nur jeden zweiten gespielten Schlag zum Trainieren des ausgewählten Instruments. Dies wird beispielsweise benötigt, wenn das Ausklingen von einem vorangehenden Schlag mittrainiert werden soll.

12 Trainingsoption

Nimmt jeden gespielten Schlag für das Trainieren des ausgewählten Instruments.

13 Automatische Speicherung der Wav-Datei

Checkbox zum Einstellen, ob die Wav-Datei nach Beendigung der Klassifikation gespeichert werden soll.

14 Automatisches Starten des Midi-Players

Checkbox zum Einstellen, ob die MIDI-Datei nach Beendigung der Klassifikation automatisch abgespielt werden soll.

6 Testresultate

Um die Leistungsfähigkeit unseres Systems zu beurteilen, führten wir Tests mit verschiedenartigen Rhythmen, Geschwindigkeiten und variabler Dynamik durch. Die aufgenommenen Teststücke sowie das Resultat als MIDI-Datei sind auf der beiliegenden CD zu Rekonstruktions- und Demonstrationszwecken im Ordner „Test Files“ abgespeichert. Die für die Tests verwendeten Feature Daten und Matching Templates sind ebenfalls in diesem Ordner vorhanden.

6.1 Einleitung

Die Systemtests fanden mit der Hardware, welche unter Kapitel 5.1 beschrieben ist, statt. Der Raum ist grossflächig, ein Laborraum, von einer Akustik wie man es zu Hause auch kennt. Erwähnenswert sind die diversen Hintergrundgeräusche, welche von EDV - Geräten und anderen Personen verursacht wurden.

Es folgen nun die Testresultate zu den einzelnen Stücken und anschliessend eine Zusammenfassung zu den Tests. In der Musik wird das Tempo beispielsweise wie folgt angegeben: ♩ = 120. Dies bedeutet, dass 120 Viertelnoten pro Minute gespielt werden. Die Testresultate werden durch einen kurzen prägnanten Kommentar, sowie mit einer subjektiven Note festgehalten.

6.2 Tests ab Notenblatt

Die Folgenden Beispiele sind aus dem Übungsheft „Today's Sounds for Drumset“ von Murray Houllif [2].

Lesson 1_11

Simpler, grundlegender Rhythmus für Rockbeats mit viertel und achte Noten. Gespielt mit Hihat, Pauke und Snare. ♩ = 120.

Resultat: Der erste Schlag fehlt in der MIDI-Datei. Das Erkannte ist korrekt. Note 5.8

Lesson 2_10

Ebenfalls Rhythmus des Rock Beat, mit viel Paukenschlägen, Hihat- und Snareschlägen. Enthält viertel und achte Noten. ♩ = 120.

Resultat: Es wird 1 Ride-Schlag erkannt, der nicht gespielt wurde. Note 5.8

Lesson 3_3

Rock Rhythmus mit sechzehntel Noten bei Snare, Pauke und Hihat mit achte Noten. ♩ = 120.

Resultat: Das Gespielte wird korrekt erkannt. Note 6

Lesson 8_2

Disco beat mit sechzehntel Noten bei Hihat. ♩ = 120.

Resultat: Ein Tomschlag erkannt, der nicht gespielt wurde. Note 5.8

Lesson 15_10

Swing-Jazz beat mit polyphonen Schlägen aus Ride, Pauke und Snare. Verwendet Triolen, ♩ = 120.

Resultat: 1 Single-Paukenschlag und 2 Rideschläge nicht erkannt. 8 Paukenschläge in Pauke-Ride-Snareschlag nicht erkannt. Note 5

Lesson 24_1a

Cha Cha gespielt mit der Kuppel, Snare, Pauke und Tom1. Polyphonie aus Kuppel, Snare und Pauke sowie Kuppel, Tom1 und Pauke. ♩ = 120.

Resultat: Bei Schlägen mit Tom und Ride wird das Ride nicht erkannt. Bei polyphonen Schlägen unter den Fellinstrumenten wird die Pauke nicht erkannt. Note 4.5

Lesson 25_2a

Tango mit Polyschlägen aus Snare und Tom3 sowie Snare und Pauke. ♩ = 120.

Resultat: Pauke- und Tomschläge fehlen in polyphonen Schlägen. Da der Rhythmus ausschliesslich aus Polyschlägen besteht, wirken die Fehlklassifikation als sehr störend. Note 2

Lesson 25_3a

Bossa Nova, wobei die Instrumente anders angespielt werden müssten, um den richtigen Klang für Bossa Nova zu erzeugen. Rim Click wird durch Snare ersetzt, weil die Software nicht darauf trainiert ist. ♩ = 120.

Resultat: Zuviel Rideschläge erkannt. Dies liegt daran, dass das System nicht auf Polyschläge aus Hihat, Snare und Pauke trainiert ist. So werden sie als Snare mit Ride erkannt. Die Pauke fehlt bei den polyphonen Schlägen unter den Fellinstrumenten. Das Öffnen des Hihats mit dem Fuss führt zu mehrfachen Hihatschlägen. Auch dafür ist das System nicht trainiert, dieses Stück zeigt die Grenzen des Systems auf. Note 4.5

Lesson 25_4d

Jazz-Samba

Resultat: Die starke Polyphonie in diesem Stück führt dazu, dass viele Schläge in der MIDI-Datei fehlen. Dies liegt daran, dass das System nicht dafür ausgelegt ist. Note 4

6.3 Test mit eigenen Rhythmen

Pretest2

Rhythmus mit lauter Toms

Resultat: Alle Schläge korrekt erkannt. Note 6

Pretest3

Rock Rhythmus mit Tomschlägen

Resultat: Alle Schläge korrekt erkannt. Note 6

Pretest6

Rock Rhythmus mit Beckenschlägen

Resultat: Nicht alle Beckenschläge korrekt erkannt. Nachfolgende Paukenschläge fehlen teilweise. Dies liegt am langen Nachklingen der Becken. Note 5

Pretest7

Rock Rhythmus mit Beckenschlägen, wobei die Becken so angespielt wurden, dass sie länger ausklingen als bei Pretest6.

Resultat: Alle Beckenschläge korrekt, ausser dass 2 Rideschläge erkannt wurden, die nicht gespielt sind. Note 5.5

Pretest8

Rock Rhythmus mit Pauke, Snare und Hihat.

Resultat: Alles korrekt erkannt. Note 6

Pretest9

Rhythmus mit Ride, Snare und Pauke.

Resultat: Es werden 3 Kuppel-Schläge zuviel erkannt. Note 5.5

Pretest11

Rhythmus mit Ride, Hihat, Snare, Pauke und Splash.

Resultat: Kompliziertes Stück. Ein paar Rideschläge zuwenig erkannt. Note 5

Pretest12

Schwieriger Rhythmus mit Ride, Hihat, Tom1, Tom2, Tom3, Snare und Pauke

Resultat: Sehr kompliziertes Stück. Der ursprüngliche Rhythmus wird einigermaßen wiedergegeben. Note 4

Groove4

Stück mit Crash und Splash

Resultat: 1 Crash zuwenig und 1 Ride zuviel. Note 5.5

Groove5

Stück mit Ride und Kuppel.

Resultat: 1 Rideschlag nicht erkannt. Note 5.8

Groove9

Knacknuss. Hihat, Crash, Splash, Pauke, Snare, Tom1, Tom2 und Tom3

Resultat: Zuviel Ride-Schläge erkannt. Die Rideschläge könnten mit der Instrumentenauswahl ausgeschaltet werden. Note 4.5

Groove10

Ähnlich wie Groove9. Hihat deutlicher gespielt, was zu weniger falschen Ride-Klassierungen führt.

Resultat: Die Crashschläge werden mehrfach erkannt. Note 5

6.4 Dynamik- und Geschwindigkeitstests

Speed2 (Hihat Test)

Alle Hihatschläge werden korrekt erkannt. Endgeschwindigkeit 320 Schläge pro Minute. Note 6

Speed3 (Snare Test)

Die Snareschläge stimmen mit dem Original überein, allerdings sind in der MIDI-Datei auch Tom- und Hihatschläge zu hören, die nicht gespielt wurden. Note 5

Speed4 (Hihat und Snare Test)

Schneller Rhythmus mit Pauke, Snare und Hihat. Note 5

Speed6 (Tom1 Test)

Zu den Tom1 Schlägen Tom3 Schläge erkannt, die nicht gespielt wurden. Note 5

Dynamik1

Leise Schläge fehlen in der MIDI-Datei. Note 4

Dynamik2

Leise Schläge fehlen in der MIDI-Datei. Note 4

6.5 Zusammenfassung der Tests

Die Testresultate sind durchaus gut. Über alle Tests gemittelt beträgt die Note 5.0. Das Erkennen polyphoner Schläge unter Fellinstrumenten funktioniert nicht ganz befriedigend. Das mehrfache Erkennen der Beckenschläge wirkt für den Hörer störend. Die Fehler entstehen aus zwei grundlegenden Problemen.

- Polyphonie unter Fellinstrumenten, Problem des Matching-Algorithmus

Der Matching-Algorithmus ist von seiner Art her für jede beliebige Kombination geeignet. Es ist nicht nötig, die polyphonen Schläge zu trainieren. Sie aber korrekt zu erkennen ist wie bereits in Kapitel 3.2.4 erwähnt sehr schwierig.

- Nachklingen der Becken, Problem des Feature-Algorithmus

Der Feature-Algorithmus kommt dem menschlichen Hören näher als die Matching-Methode. Er klassifiziert, was er in einem kleinen Ausschnitt aus einem Musikstück „hört“. Leider „hört“ er auch das Nachklingen der Beckeninstrumente und klassifiziert unter Umständen die Schläge doppelt. Die Problematik ist unter Kapitel 3.4 näher erläutert.

7 Schlusswort und Diskussion

7.1 Diskussion der Ergebnisse

Entstanden ist ein System mit einer modernen Benutzeroberfläche zur automatischen Umwandlung (Transcription) von Schlagzeugmusik in Notenschrift. Die Leistungsfähigkeit ist für die am meisten gebrauchten Instrumente (Hi-hat, Snare, Pauke) sehr gut. Bei polyphonen Schlägen und der Überlagerung von lang ausklingenden Metallinstrumenten treten Probleme auf. Die Performance des Systems kann einerseits mit dem Ausschalten nicht benutzter Instrumente, andererseits mit spezifischem Trainieren der Feature Daten fortlaufend gesteigert werden. Zusammenfassend kann man sagen, dass die entwickelte Software für die Leistungsansprüche eines Schlagzeuganfängers bestens geeignet ist. Für einen erfahrenen Schlagzeuger, welcher komplizierte Improvisationen in MIDI-Form umwandeln möchte, ist das System dagegen weniger geeignet.

7.2 Verbesserungsvorschläge

Anhand von diversen Tests sind uns immer wieder Ideen eingefallen, mit denen man das System verbessern könnte. Aus zeitlichen Gründen jedoch war es unmöglich diese umzusetzen.

Mögliche Verbesserungen:

- Den Feature Algorithmus für mehr Instrumentenkombinationen auslegen. Mit dem Trainieren aller Kombinationen die gespielt werden, würden Fehlklassifikationen verringert.
- Beim Template Matching Algorithmus könnte eventuell mit Methoden der Bildverarbeitung die Treffsicherheit erhöht werden.
- Die Performance könnte mit Informationen höherer Ebene gesteigert werden. Eine solche Information könnte zum Beispiel der gespielte Musikstil oder Ähnliches sein.

Mögliche Erweiterungen:

- Die Dynamik der gespielten Schläge könnte auch in die MIDI-Datei eingebracht werden. Das Resultat wäre der Realität näher.
- Die Benutzeroberfläche könnte mit Zusatzfunktionen (Beurteilung der Tempogenaugigkeit des Spielers, Beurteilung der Dynamik, Trainerfunktion: Gespieltes mit Vorlage vergleichen -> Fehler markieren etc.) ausgestattet werden.

7.3 Schlusswort

Wir blicken auf eine interessante und sehr intensive Diplomarbeitszeit zurück. Das Aufgabengebiet war umfangreich. Die Hürden, die es zu bewältigen galt, erschienen manchmal hoch, doch mit der Unterstützung die wir von Betreuern, Dozenten und Assistenten erhielten, haben wir unser Ziel erreicht. Besonderen Dank gilt den beiden betreuenden Dozenten, welche bei offenen Fragen nie um eine Antwort verlegen waren. Ein grosses Dankeschön möchten wir Ivo Oesch aussprechen, der uns immer wieder bei kleineren Programmierschwierigkeiten zur Seite stand. Auch ein Merci an René Widmer der für Fragen beim Programmieren in C / C++ Anlaufstelle Nummer eins war. Last but not least besten Dank an Michael Bernhard für seine hilfreichen Tips.

8 Literaturverzeichnis

- [1] Tzanetakis, G, Perry Cook: Musical Genre Classification of Audio Signals, IEEE Princeton University, USA, Juli 2002
- [2] Bernhard, M und Straubhaar, A: Beat Detection – akustisch und optisch, HTI Burgdorf, Schweiz, Januar 2004
- [3] Herrera, P, Yeterian, A und Gouyon, F: Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques, Universität Pompeu Fabra Barcelona, Spanien
- [4] Houllif, M: Today's Sound For Drumsset, Kendor Music, Delevan New York, USA
- [5] Herrera, P, Dehamel, A und Gouyon, F: Automatic labeling of unpitched percussion sounds
Universität Pompeu Fabra Barcelona, Spanien
- [6] Andersson, T: Audio Classification and Content Description,
Department of Computer Science and Electrical Engineering Division of Signal Processing, Ericsson Research, Corporate Unit Lulea, Schweden
- [7] Logan, B: Mel Frequency Cepstral Coefficients for Music Modeling,
Cambridge Research Laboratory, England, 2000
http://www.hpl.hp.com/personal/Beth_Logan/
- [8] Abel, J. S: Bark and ERB Bilinear Transforms,
Center for Computer Research in Music and Acoustics, Stanford University, USA, November 1999
- [9] Haykin, S: Neural Networks: A Comprehensive Foundation, Prentice Hall PTR, Upper Saddle River, USA
- [10] Melzer, T: Statistische Mustererkennung,
Institut für Photogrammetrie und Fernerkundung , TU Wien, Österreich, 2003
- [11] Clausen, M: Matlab-MIDI Tools,
Institut für Informatik III, Universität Bonn, Deutschland
<http://www-mmdb.iai.uni-bonn.de/download/matlabMIDItools/>
- [12] The Sonic Spot: Standard MIDI Files,
<http://www.sonicspot.com/guide/midifiles.html>
- [13] Yoshii, K, Goto, M, Okuno, H. G: Automatic Drum Sound Description for Real World Music using Template Adaptation and Matching Methods,
Department of Intelligence Science and Technology, Kyoto University, Japan, 2002

Anhang A: Hilfsprogramme

AudioRecorder

Der AudioRecorder ist ein Programm für den Betriebsmodus "Live2Midi". Er liest von der Soundkarte die Audiodaten ein und speichert sie in 5 Sekunden-Abständen als temporäre Wav-Dateien ab. Diese temporären Dateien werden im Anschluss an die Aufnahme vom Drum2Midi-Algorithmus verarbeitet.

Das Programm basiert auf einer Version einer kleinen Multimedia Applikation aus dem Internet [Holme, T: How to play and record sound, <http://www.codeproject.com/audio/fister.asp>], welches für unsere Zwecke abgeändert wurde. Der AudioRecorder wurde in Visual C++ entwickelt. Abbildung 36 zeigt das Klassendiagramm. Nachfolgend werden die Klassen kurz erläutert.

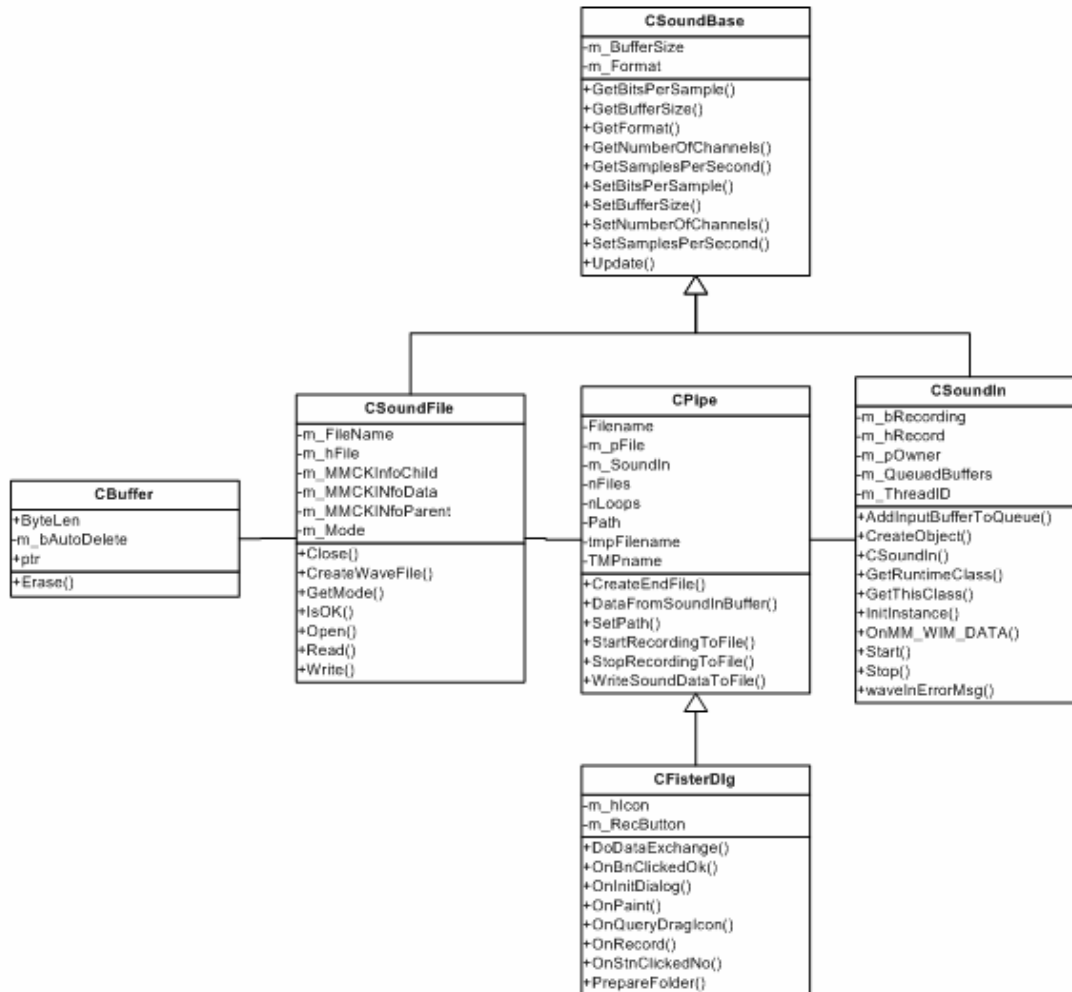


Abbildung 36: Klassendiagramm des AudioRecorders

CSoundBase

Basisklasse für CSoundFile und CSoundIn. In dieser Klasse werden die für die Wav-Datei relevanten Formatparameter (Fs, nBits, nChannel etc.) festgelegt.

CSoundFile

Stellt Funktionen für das Erzeugen, Beschreiben und Schliessen von Wav-Dateien zur Verfügung.

CBuffer

Diese Klasse dient zum Buffern der von der Soundkarte eingelesenen Audiodaten.

CSoundIn

Diese Klasse organisiert das Buffer-Handling mit den eingelesenen Audiodaten.

CPipe

Dies ist die eigentliche Hauptklasse. Sie regelt das Öffnen und Schliessen der Wav-Files, greift dabei auf die Funktionen der Klassen CSoundFile und CSoundIn zu.

CFisterDlg

Diese Klasse ist zuständig für die Darstellung des Dialog-Fensters (Abbildung 37), das Erstellen des temporären Audiodatenordners sowie der Abfrage des „Recording STOP“-Buttons. Da CFisterDlg von CPipe erbt, stehen ihr alle Kontrollfunktion der Hauptklasse CPipe zur Verfügung.



Abbildung 37: Dialog-Fenster des Audio Recorder Programms

C-Interface

Aus Geschwindigkeitsgründen mussten C-Funktionen erstellt und mit Matlab zu folgenden dynamic link libraries (dlls) compiliert werden:

match.dll		
Vergleicht die Excerpts mit den Templates (Template Matching)		
Aufruf in Matlab: <code>out = match(excerpt, template)</code>		
Input	Typ	Beschreibung
excerpt	$1 \times n$ cellarray of $f \times ?$ double matrices	Spektrogramme der zu untersuchenden Schläge
template	$1 \times m$ cellarray of $f \times T$ double matrices	Spektrogramme der Templates
Output		
out	$n \times m$ char matrix	Match-Tabelle Zeilen: gleich lang wie except Spalten: gleich lang wie template 1 = Übereinstimmung 0 = keine Übereinstimmung

Hinweis: Die Matrizen von excerpt und template müssen alle gleich viel Frequenzwerte (Zeilen) aufweisen.

? steht für beliebige Länge der jeweiligen Matrix

findpeak.dll		
Bestimmt die Peaks in einem Signal (Onset Detection)		
Aufruf in Matlab: <code>peaks = findpeak(t, S, sensitivity, range)</code>		
Input	Typ	Beschreibung
t	$1 \times n$ double vector	Zeitwerte zum Signal
S	$1 \times n$ double vector	Signal
sensitivity	1×1 double value	Empfindlichkeit (mimimaler Abszisse-Abstand)
range	1×2 double vector	Ordinaten-Bereich, indem Peaks gesucht werden Spalte 1: Minimalwert Spalte 2: Maximalwert
Output		
peaks	$m \times 2$ double matrix	Abszissen- und Ordinatenwert des Peaks Spalte 1: Zeitwert Spalte 2: Signalwert Anzahl Zeilen = Anzahl gefundene Peaks

Nachfolgende dll musste erstellt werden, weil die in Matlab eingebaute Funktion zum Aufruf von Programmen in der compilierten Form nicht funktioniert.

createproc.dll		
Ruft ein externes Programm auf		
Aufruf in Matlab: createproc(process, argument)		
Input	Typ	Beschreibung
process	string	Auszuführendes Programm
argument	string	Kommandozeilen Parameter für Programm