ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

SIGNAL PROCESSING INSTITUTE

Instrument Recognition and Transcription in Polyphonic Music

Detection of Saxophone Melody in a Jazz Quartet Recording

Term Project

by Alain Brenzikofer
supervisor:
Dr. Andrzej Drygajlo

Lausanne, June 2004

Contents

Preface

This project makes part of the author's graduate studies to obtain a masters degree in Electrical Engineering at ETH Zürich. The topic was proposed by the author. It took place at EPF Lausanne at the signal processing laboratory (LTS) during an exchange semester under supervision of Dr. Andrzej Drygajlo.

All source code is distributed under General Public License (GPL). For classification non-GPL toolboxes were used.

Introduction

Music Transcription is the task of analyzing a digital audio stream in terms of fundamental frequencies and transform the information into music notation (i.e. a MIDI stream). If there are different instruments playing, each instrument is to be assigned to a different notation score.

Instrument Recognition is the task of classifying a part of an audio stream to an instrument. This "part" of the audio stream may be a short time analysis window, one note, a whole melody or chord progression.

This project focuses on signal analysis and doesn't treat the task of transcription to a notation score. Instrument Recognition focuses on one instrument only; the tenor saxophone. The musical environment for the saxophone is restricted to a typical jazz quartet consisting of saxophone, piano, bass and drums. The acoustical environment is not restricted. The music may be recorded in a studio with artificial acoustic environments or it may be recorded live under difficult quality conditions.

The framework is structured in a way that would allow real-time processing of a continuous audio stream with few modifications. On a Pentium III calculation time is too long for a real-time application. However, it was not an aim of the project to optimize calculation time.

Contributions A new technique to detect multiple fundamental frequencies in polyphonic music has been developed during work on this project.

Unlike most of the previous work the focus in Instrument Recognition was on polyphony and not on the number of different instruments to be recognized. The number of notes played at the same time is unknown as is the number of instruments. Under these circumstances, possible features have been evaluated in terms of usability for classification.

Organization This report is organized as follows:

- Chapter 1 discusses previous work and underlying theory of importance to this project.
- Chapter 2 introduces the method for multiple F0 detection in polyphonic music.
- Chapter 3 discusses different features and models for classification.
- Chapter 4 presents performance tests for instrument classification.
- Chapter 5 summarizes the results and discusses possible improvements of the framework.

Chapter 1

Background

There is only little specific theory on instrument recognition and music transcription. However, speech / speaker recognition is a similar task with a wide research community and their theories can be used on music as well. In this section a brief overview of related theory and publications is given.

Before discussing possible techniques, we should have a look at the signal characteristics of music:

- There are harmonic and percussive instruments. Percussive instruments will be ignored for later discussion.
- Different sources can generate different "sounds" at different points in time and space.
- A sound consists of a fundamental frequency (F0) and harmonics (n·F0).
- For the music to sound agreeable to the human ear, F0s have certain ratios between each other, called intervals. These intervals are fractions of small integer numbers (prime: 1, octave: $\frac{1}{2}$, fifth: $\frac{2}{3}$ ...)
- Because of these ratios and the resulting overlapping of harmonics, different sources are:
  - not statistically independent
  - not every time-frequency-bin belongs to only one source (not W-disjoint orthogonal [2])
- In stereo live recordings with one stereo microphone, the spacing of different sources affects intensity and phase differences between left and right channel.

- In studio recordings there is no phase information because usually only intensity-stereophony is used. Even the intensity is mixed close around the center of the panorama today (unlike most "Beatles" recordings for example, where instruments are assigned either to the left or right channel).
- One saxophone usually plays at one pitch at the time. Polyphonic sounds are possible but rarely used.
- Piano and bass can play more than one note at the time. So one of these instruments produces more than one F0

1.1 Source separation

A first idea was to separate the instrument's signals with Independent Component Analysis (ICA), for example Blind Source Separation (BSS). But ICA relies on the assumption that the signal is statistically independent which is not at all the case. So ICA is not considered to be useful. In [2], Degenerate Unmixing Estimation Technique (DUET) is introduced. DUET separates signals by clustering the amplitude-delay histogram. According to spacial distribution the sources have different amplitude for left and right and a phase delay. DUET assumes W-disjoint orthogonality and therefore does not perform well for overlapping harmonics. Different possibilities to avoid that problem are introduced in [8], [9] and [10].

1.2 F0 estimation

For this project, source separation is an overkill. In fact there is no need to really separate the signals. All we need is information on characteristics of each of the signals in the mixture. First of all we need all F0s of all instruments. Techniques introduced in [7], [6] and [5] contributed to that part of the project.

1.3 Instrument recognition

Previous work on recognition of musical instruments includes [1] and [4] for recognizing monophonic samples of many different instruments.

In [3], two instruments may play at the same time. Gaussian Mixture Models are combined with missing feature theory.

This project focuses on polyphony, simplifying the problem by only deciding if the instrument is a saxophone or not.

Chapter 2

Polyphonic F0 Detection

The first problem to solve when detecting one instrument playing with others is developing a robust algorithm to obtain all fundamental frequencies (F0s) of each instrument at a given time. For the saxophone we can introduce some simplifications. A saxophone usually plays one note at a time, so there's only one fundamental frequency to catch. Moreover, saxophone is predominant in volume what makes it more likely to be detected. The treatment of that problem is discussed in chapter 2.2

The method described in this chapter performs well for saxophone, but it does not reliably detect all soft notes played by other instruments.

A second topic is to group the F0s to notes. This can not only be done by collecting F0s in semitone-bins because two instruments might play the same note for a while. Furthermore we need time-domain features for later classification. In Section 2.3 an agent structure based on the idea in [5] is described which groups F0 trajectories to notes.

2.1 Signal analysis

The signal analysis structure is described in Appendix B. Table 2.1 shows the parameters chosen for signal analysis.

2.2 F0 detection algorithm

An algorithm for multiple F0 detection should:

- be independent of the existence of the fundamental frequency in the signal (as is the human ear)
- avoid marking harmonics of F0s as new F0s. But:

sampling frequency	11'025	Hz
windowing function	raised cosine	
window size	1024	samples
window overlap	512	samples
FFT size	1024	samples
⇒frequency resolution	5.38	Hz
⇒lowest resolvable semitone	90	Hz
⇒time resolution	93	ms
lowest F0 freq.	100	Hz
highest F0 freq.	1000	Hz

Table 2.1: Parameters for signal analysis

- still recognize two different F0s in octave interval.
- detect overlapping harmonics to know which harmonics contribute to reliable information.

The first step is to take the autocorrelation function (ACF) of the spectrum because this solves the problem of missing fundamental frequency and enhances peaks over noise. Prewhitening the spectrum with the cepstrally lowpass liftered spectral envelope improves the ACF.

In the ACF, peaks of the quasi-periodic spectrum repeat at $n \cdot F0$ what leads to unwanted virtual F0s. The Enhanced Summary Autocorrelation Function described in [6] solves that problem but showed the disadvantage of canceling octaves and enhancing only strong F0s in this application. To solve these problems, the Cepstrally Enhanced Autocorrelation Function has been developed.

2.2.1 Cepstrally Enhanced Autocorrelation Function (CEACF)

The motivation to use the cepstrum to improve the spectrum-ACF is that harmonics repeat not at $n \cdot F0$ but at $\frac{1}{n \cdot T0}$. By multiplying ACF and cepstrum peaks, only true-F0s remain, as Figure 2.2.1 shows. There are some obstacles. First, the cepstrum is sampled uniformly on a time axis. This axis has to be inverted to be comparable with the spectrum-ACF frequency axis. Because the good resolution for high quefrencies is mapped to a lower resolution at low frequencies, peaks may be lost. To avoid this, peaks are detected before inverting the quefrency axis as shown by vertical lines in the middle

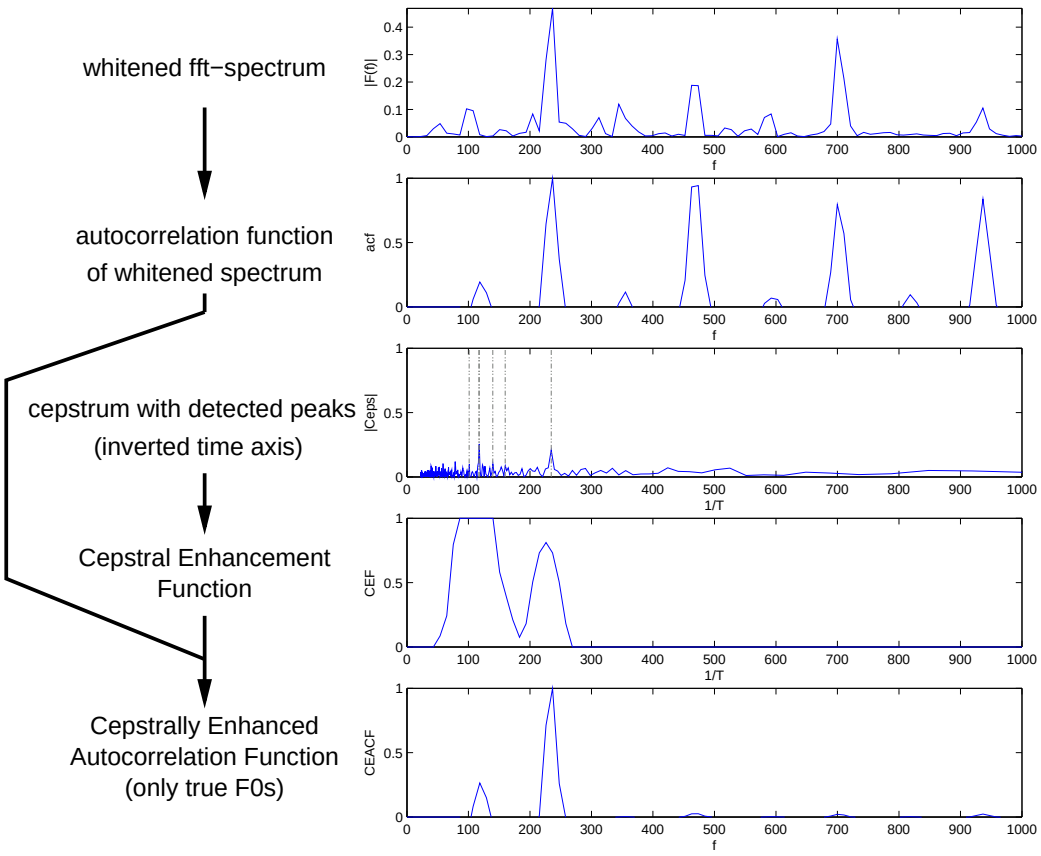

Figure 2.1: The different steps to obtain the CEACF for two F0s at about 110Hz and 220Hz

plot of Figure 2.2.1. Now, a virtual Cepstral Enhancement Function (CEF) is generated by convolving a dirac-representation of detected peaks with a raised cosine.

This CEF is now multiplied with the spectrum-ACF to obtain the CEACF. All peaks in the CEACF that are higher than an arbitrary threshold are taken as F0s.

2.2.2 Frequency interpolation

For transcription and even for detecting overlapping harmonics, the frequency resolution of 5.38 Hz is not sufficient. To get a resolution of 0.34 Hz the spectrum around each F0 is interpolated by a factor of 16.

2.2.3 Overlapping harmonics detection

To detect which harmonics are overlapping, every F0 detected is extended to all its harmonics (n·F0) within the given frequency range. Harmonics closer than a certain tolerance are masked as overlapping.

2.3 Agent structure

After finding the F0s we need to group them together to notes in time domain. To obtain characteristic information on notes we need to have the whole note from attack until the end, not only some fixed-width window's information. To be as flexible as possible, an agent structure similar to the one described in [5] is used. Agents are data structures that are created dynamically whenever a new possible F0 trajectory starts and keep track of that trajectory until that note ends. The rules are kept very simple. For a new set of F0s following rules apply:

- all F0s that are closer than a semitone to the last F0 of an existing agent are assigned to that agent (bending is possible),
- all agents without a new F0 are "killed". Their note has ended. If they have a minimal length they will be classified later,
- for every F0 not assigned to an agent, a new agent is created.

It was planned to extend these rules by some instrument adaptive algorithms, adapting instrument-specific features but also record-specific features like position in space (that could be obtained by Computational Auditory Scene Analysis, CASA). But finally these simple rules are performing well enough.

2.4 Results

Figures 2.2 and 2.4 show the output of F0 detection. Every line corresponds to one agent. The detection performance is good for strong F0s but softer ones are not reliably detected. In our case this might even be an advantage because the probability for saxophone F0s to be recognized is higher than for piano. However, it is a disadvantage not to get all F0s because we need them for detecting overlapping harmonics.

Figure 2.3 and 2.5 show only the strongest F0 for every window. Without any instrument recognition this already extracts the saxophone quite well for `chris`. But because the saxophone doesn't play the whole time we still need a classifier.

As you see in Figure 2.5 in sample `mobetter` the saxophone is not always predominant

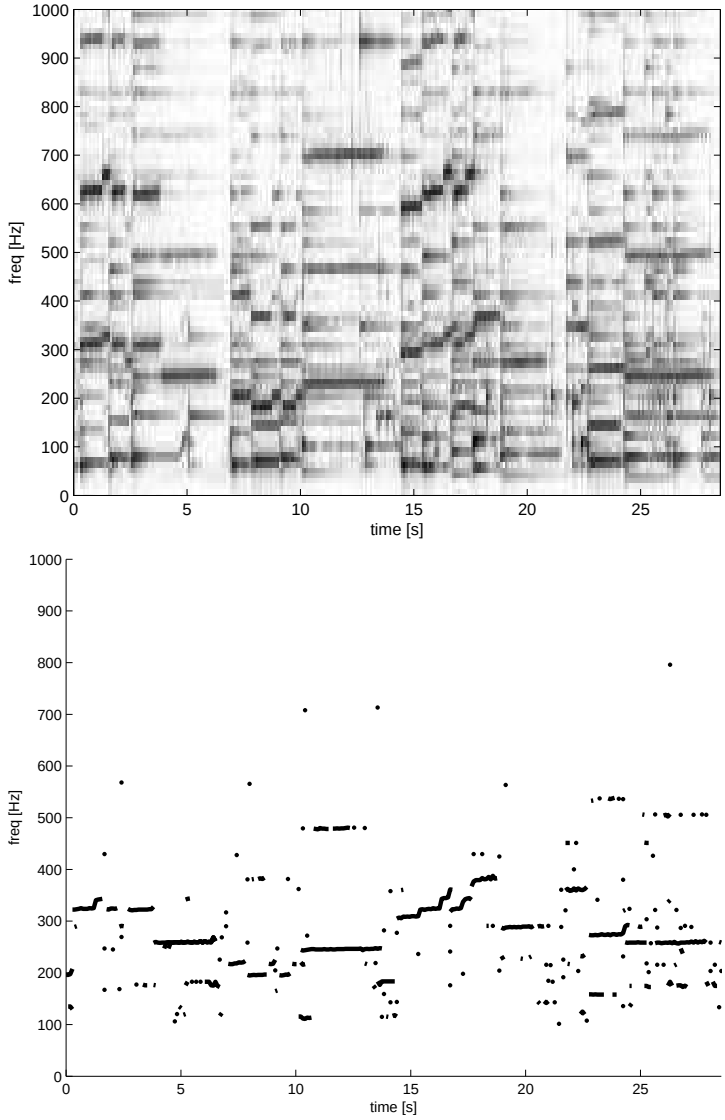

Figure 2.2: Top: Spectrogram of sample `chris`. Bottom: Detected F0s

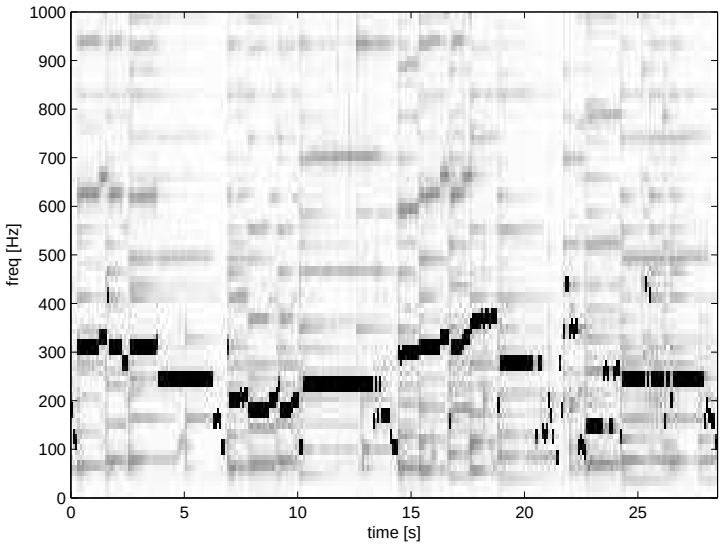

Figure 2.3: predominant F0 for sample `chris` printed in black over spectrogram. Almost always it's the saxophone's F0s being predominant.

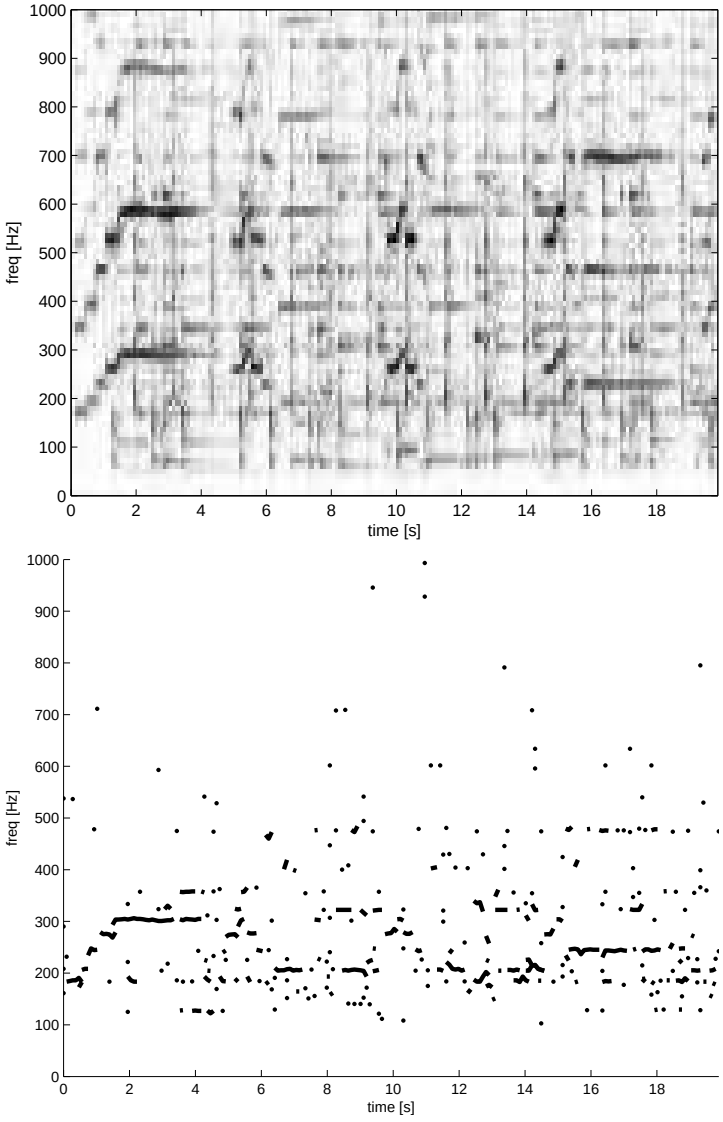

Figure 2.4: Top: Spectrogram of sample `mobetter`. Bottom: Detected F0s

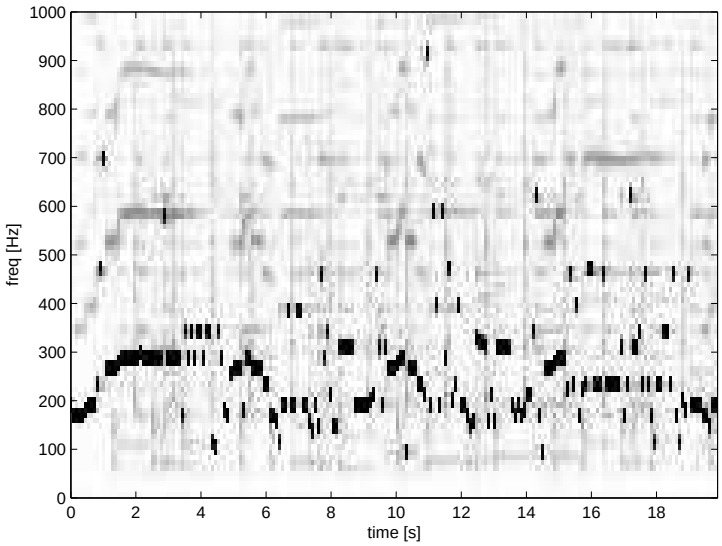

Figure 2.5: predominant F0s for sample `mobetter` printed in black over spectrogram

Chapter 3

Instrument Recognition

The second part of this project consists of recognizing saxophone sound. Each note detected with methods described in Chapter 2 has to be classified to either "sax" or "nosax".

3.1 Features

This chapter discusses all features that have been evaluated. Only a few of them were really used. The pairs of features being plotted against each other are chosen intuitively and might not be optimal in terms of statistical (in)dependence.

All features are normalized to their 2-σ-range. The feature plots show their 4-σ-range.

3.1.1 Spectral features

Spectral features are calculated for each frame of an agent. Because of the reasons explained in Section 3.1.2, only F0-dependent features are used.

The only features used for classification are *specEnvMSErel* and *harmEnvMSErel*.

specEnvMSErel In music there are no formants like in speech recognition, but the spectral envelope of an instrument is a characteristic feature. In polyphonic music we cannot just take the envelope of the spectrum because this envelope corresponds to a superposition of all instruments playing at that time.

The first step is to build an envelope consisting only of the amplitudes at harmonic frequencies of each F0. Without overlapping harmonics it would work. But this case is rare..

When harmonics overlap we don't know if they add up or cancel each other out as this depends on the phase. It was tried to find some phase-dependency between harmonics. But with given analysis framework, no correlation of phase between harmonics could be found.

To get a measure of how big the influence of the overlaps is, signal energy of a F0 is compared to the summed signal energy of all overlap contributors. Even if the signal energy itself cannot be calculated correctly when overlaps occur, it gives a good estimation. This energy ratio is taken as an indicator of reliability of harmonic amplitudes. In this context we define reliability as the ratio of own signal energy to the energy of all overlapping signals (Harmonic Overlap Energy Ratios, HOERs). No overlap means a reliability of 1. Figure 3.1 shows two model envelopes and one F0's harmonic amplitudes with reliabilities in the lower plot.

To compare one F0's harmonics to the model envelopes we assume that overlap errors are distributed symmetrically around zero for each harmonic and the sum of all errors for one F0 is zero. This way we can set the average amplitude to zero and calculate the distance of harmonics to each model envelope. Using the reliabilities as weights we can calculate a weighted Mean Square Error (MSE) which finally is represented by this feature.

harmEnvMSErel As discussed in [4], not only the spectral envelope of harmonics but also the inter-harmonic-amplitude ratios are of interest for classification. First it was tried to use the autocorrelation function of harmonic's amplitudes but finally it worked better to use harmonic ratios.

The first few harmonic amplitudes ($n{\cdot}$F0, $n = 2..5$) are divided by the amplitude of the fundamental frequency. Independent of F0 frequency we get a measure for ratios between first harmonics. The reason not to go beyond 5^{th} harmonic is the fact that given the sampling rate and maximum F0 frequency defined as 1 kHz, 5 is the minimal number of harmonics every F0 has.

logF0 Defining the range for possible F0s according to saxophone range already cuts away low bass notes and high piano notes, so this feature is of no use unless other features are dependent on frequency (which is already compensated for above features).

energyRel Relative signal energy of one particular F0 divided by total signal energy in analysis window. As mentioned in section 2.4 the relative

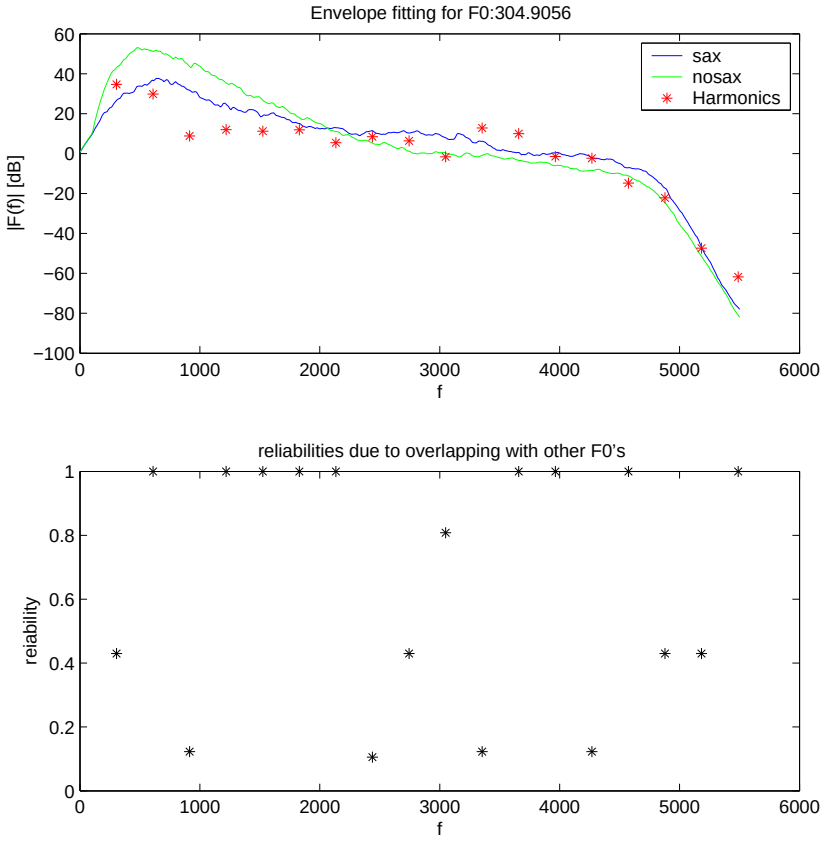

Figure 3.1: Spectral envelope models and reliability of harmonics

signal energy is a good indicator for saxophone because it is very often predominant. But here models are trained with one-class samples only, so there's no predominance and this feature is useless. For future work it should be considered to train models with mixtures of both classes.

3.1.2 Cepstral features

The two plots in Figure 3.4 show that cepstral coefficients would be very useful for classifying instruments when not playing together as in the training sets used.

Unfortunately tests showed that these features are useless when both classes of instruments play together. In this case, all agents obviously get the same cepstral features for the same frames what makes clear that "global" frame features are of no use.

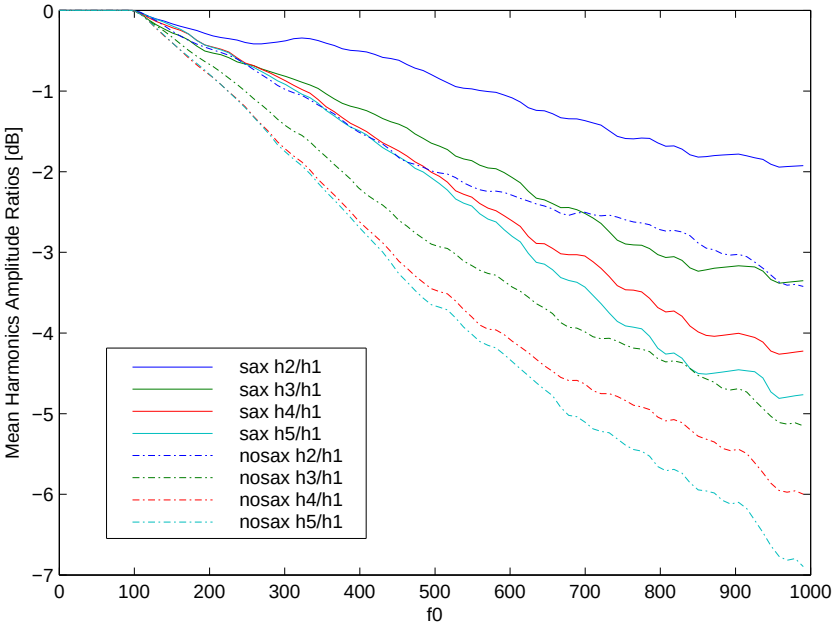

Figure 3.2: Harmonic ratio envelope models

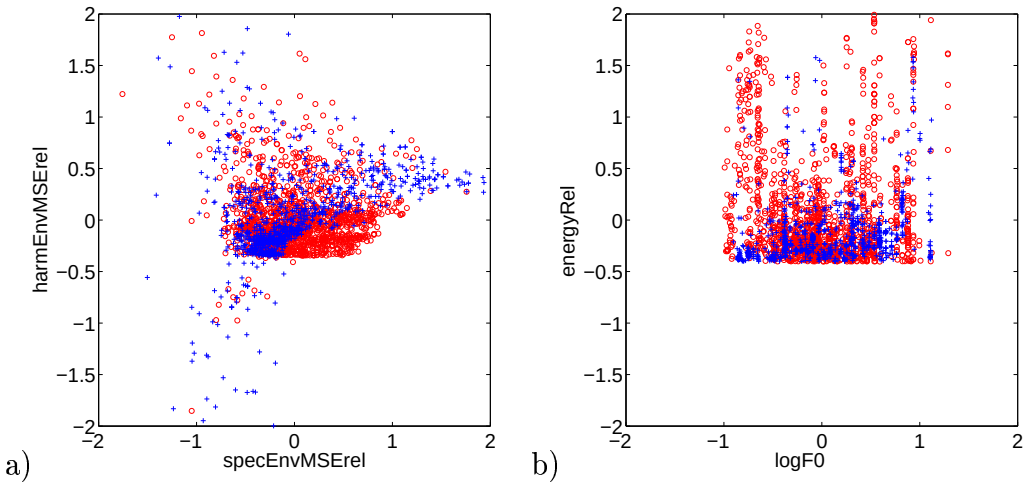

Figure 3.3: Distributions of spectral features

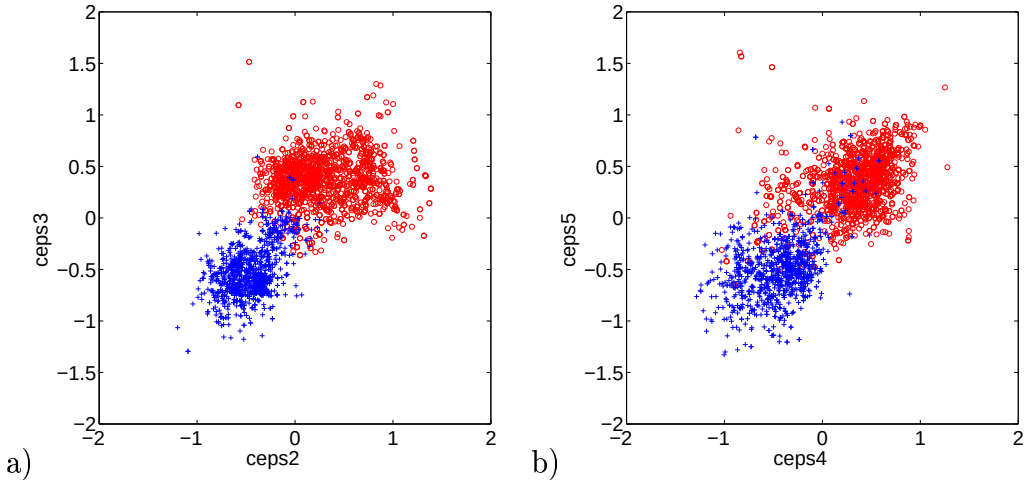

Figure 3.4: Distributions of cepstral features

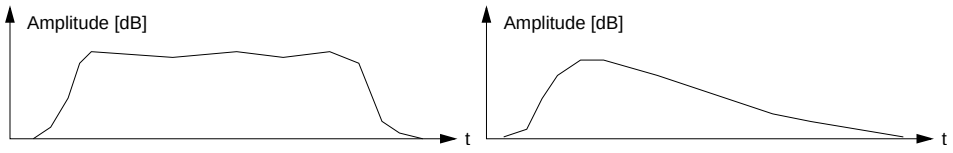

Figure 3.5: Typical temporal amplitude curves. Top: saxophone, bottom: piano / bass

3.1.3 Temporal features

Temporal features are particularly useful for longer notes, meaning more than 10 windows. Unfortunately the F0-detection usually does not catch the whole note from beginning to the end. By extending an agent to earlier and later windows the whole note can be found. This is done by reading out the spectrogram at the agent's frequency beyond both ends of the agent.

stdFreq Standard deviation if F0 frequency. Piano should have no change at all in frequency but saxophone can bend notes or play vibrato (fluctuation in pitch).

noteLen Note length. Saxophone can play longer notes than piano. This feature is weak because for short notes there's no difference a priori probability. The note length is obtained by first widening the agent in time and then count the number of time steps with an amplitude above a threshold.

medianAmpDiff Median of amplitude derivative

crest Ratio between maximum amplitude and mean amplitude

riseMax Maximum positive derivative. This feature could be improved substantially by increasing time resolution.

decayMax Minimum negative derivative. This feature could be improved substantially by increasing time resolution

3.2 Models

Three different classifiers have been evaluated: Gaussian Mixture Models (GMM), Support Vector Models (SVM) and Neuronal Networks (NN). GMMs showed to perform best on the weak features introduced in the last

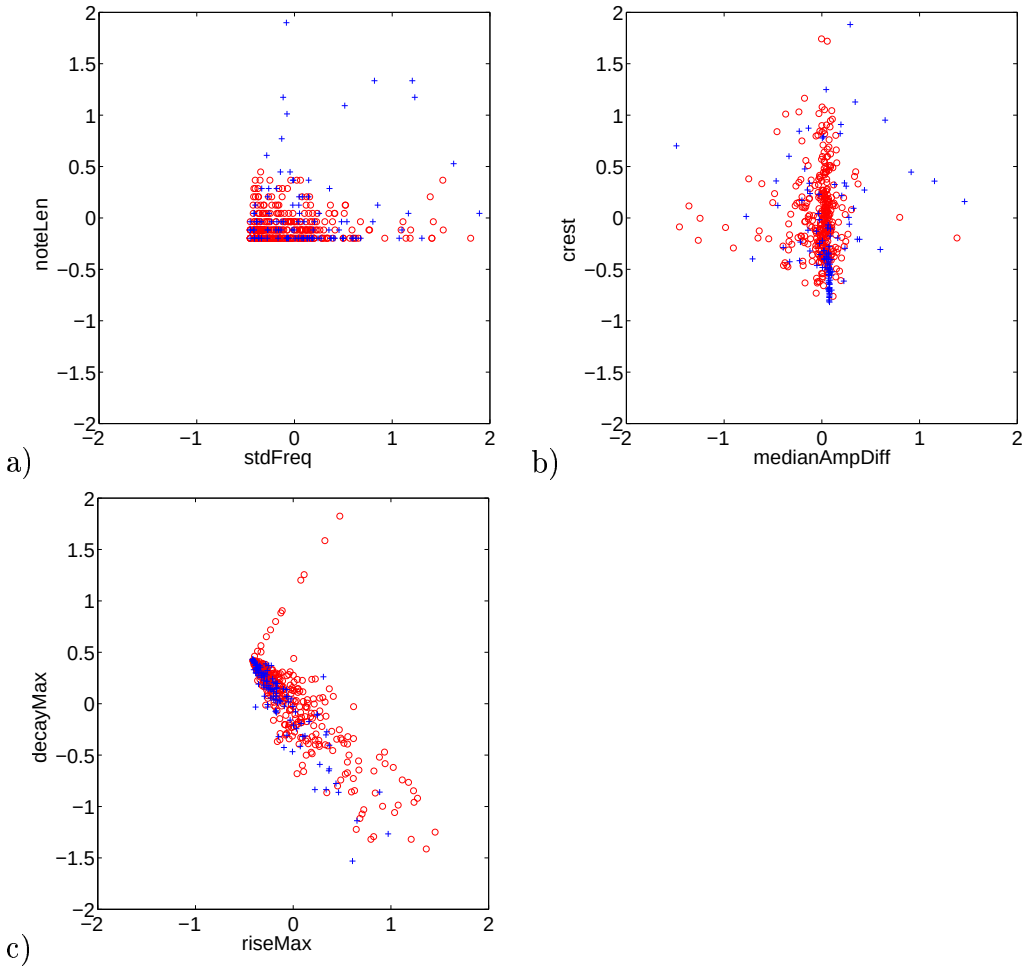

Figure 3.6: Distributions of temporal features

sections. SVM and NN sometimes showed similar performance but because of reported success with GMM in [3], work was concentrated on GMM. The following discussion will therefore be on GMM.

Spectral features used were *specEnvMSErel* and *harmEnvMSErel* introduced in Section 3.1.1. Temporal Features used were *stdFreq*, *noteLen*, *medianAmpDiff* and *crest* introduced in Section 3.1.3.

Two different GMMs are trained for each instrument class. One model is trained with spectral features and another one with temporal features. Table 3.1 lists samples used (see Appendix D for details on mentioned samples) and the number of training vectors for the spectral and temporal models. Because temporal features can only be obtained for every agent and not for every window as the spectral features, the training set for temporal model is small.

It would be possible to combine spectral and temporal models but then we would have to take the mean of all spectral features for one agent so the training set would be as small as for the temporal model.

No characteristic temporal dependency of spectral features could be found (i.e. short term spectral envelope that is used in speech recognition but is not useful here. This might be because of insufficient time resolution). So, for later classification spectral features will be averaged for one agent because this shows better results.

For the spectral features, a model with 50 Gaussians was trained, for temporal features 10 Gaussians. The final likelihood consists of the sum of the likelihood ratios for spectral and temporal features. Spectral and temporal features have the same weight.

A weak point that has to be improved is the choice of training samples. Only samples where only instruments of one class play are used. But finally we want to classify mixtures of all instruments. So the training set is not a good representation of the data to be classified later. (i.e. the predominance of saxophone in most mixtures is a very strong feature but is ignored when training one-class samples only.) The reason not to train mixtures was the amount of work to sort agents by instrument by hand.

Class	Used samples	total time	spec.feat.	temp.feat.
sax	`sax-range`, `sax-irgendwas`, `sax-chromatic`	106s	1032 vect.	117 vect.
nosax	`nino-bass`, `nino-pianobass`, `piano-range`, `amorous-piano`, `mobetter-pianobass`	142s	1493 vect.	346 vect.

Table 3.1: Training set data. Samples used (see Appendix D), total time and number of training vectors for spectral and temporal features.

Chapter 4

Experiments and Results

Because of the tight timing of the project only few performance tests have been done. First, the performance is tested on one-class samples, then two different jazz recordings of different quality are tested.

In Appendix D you find a list of all samples used for training and testing.

4.1 One-class samples

Samples where only instruments of one class appear were tested as a first performance check under easy conditions. If the aim would have been to classify one-class samples, the performance would have been better by choosing cepstral coefficients as features (See Section 3.1.2).

class	sample	correct notes	false alarms
saxophone only	`sax-range-freely`	34	7
piano only	`amorous-piano`	12	5
bass only	`invocation-bass`	10	0
piano & bass	`nino-pianobass`[1]	139	15
overall performance		88%	

Table 4.1: Classification performance for one-class samples

[1] made part of training set

4.2 Mixed instruments samples

Table 4.2 shows classification performance for sample `chris` (see appendix D), a studio recording with clear, predominant saxophone playing slow melody. In this favorable case the system performs reasonable

Table 4.3 shows classification performance for sample `mobetter` (see appendix D), a live recording with one stereo microphone in a bar. Classification performance is poor in this case.

See Figures 4.1 and 4.2 to compare true saxophone trajectories to classification output.

	is sax	is nosax	% correct
classified as sax	18	5	78%
classified as nosax	8	11	57%

Table 4.2: Classification performance for `chris`

	is sax	is nosax	% correct
classified as sax	7	8	46%
classified as nosax	22	24	52%

Table 4.3: Classification performance for `mobetter`

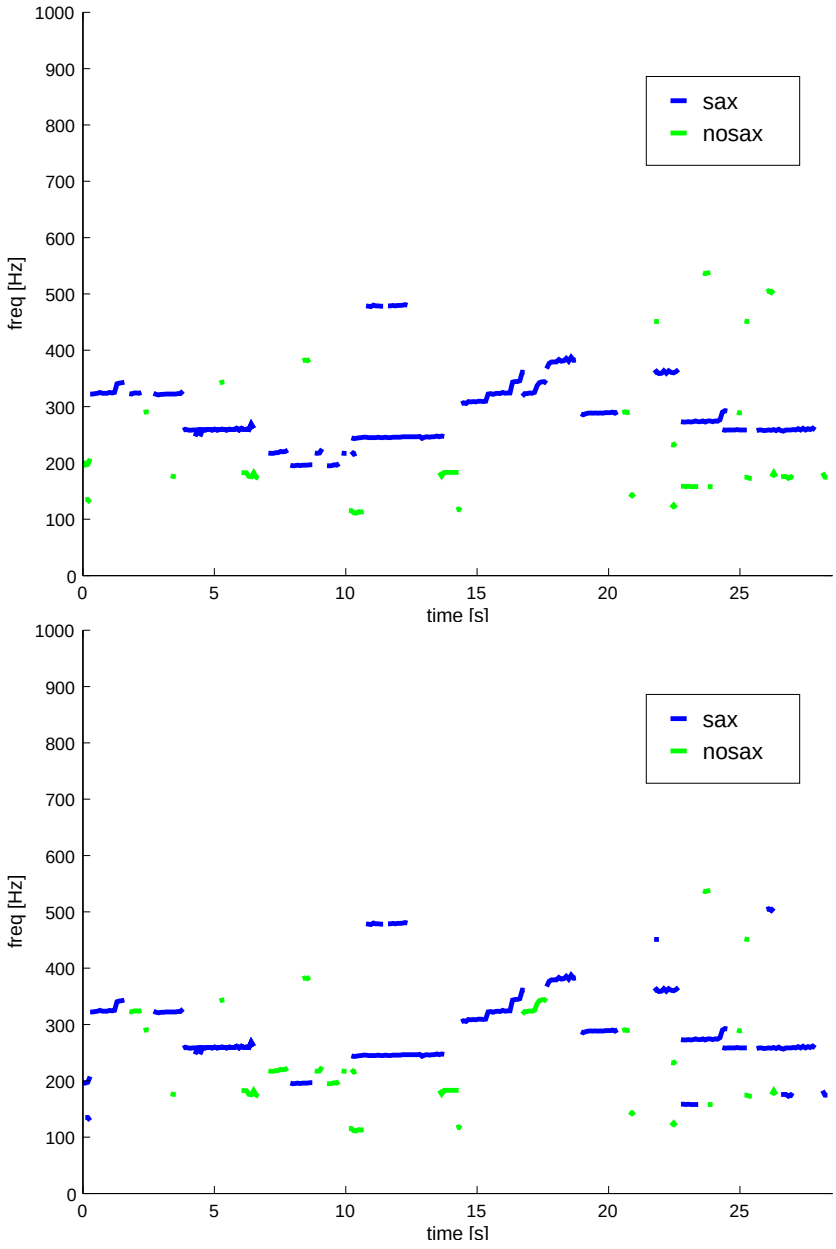

Figure 4.1: Classification for sample `chris`. Top: true classification. Bottom: GMM classification

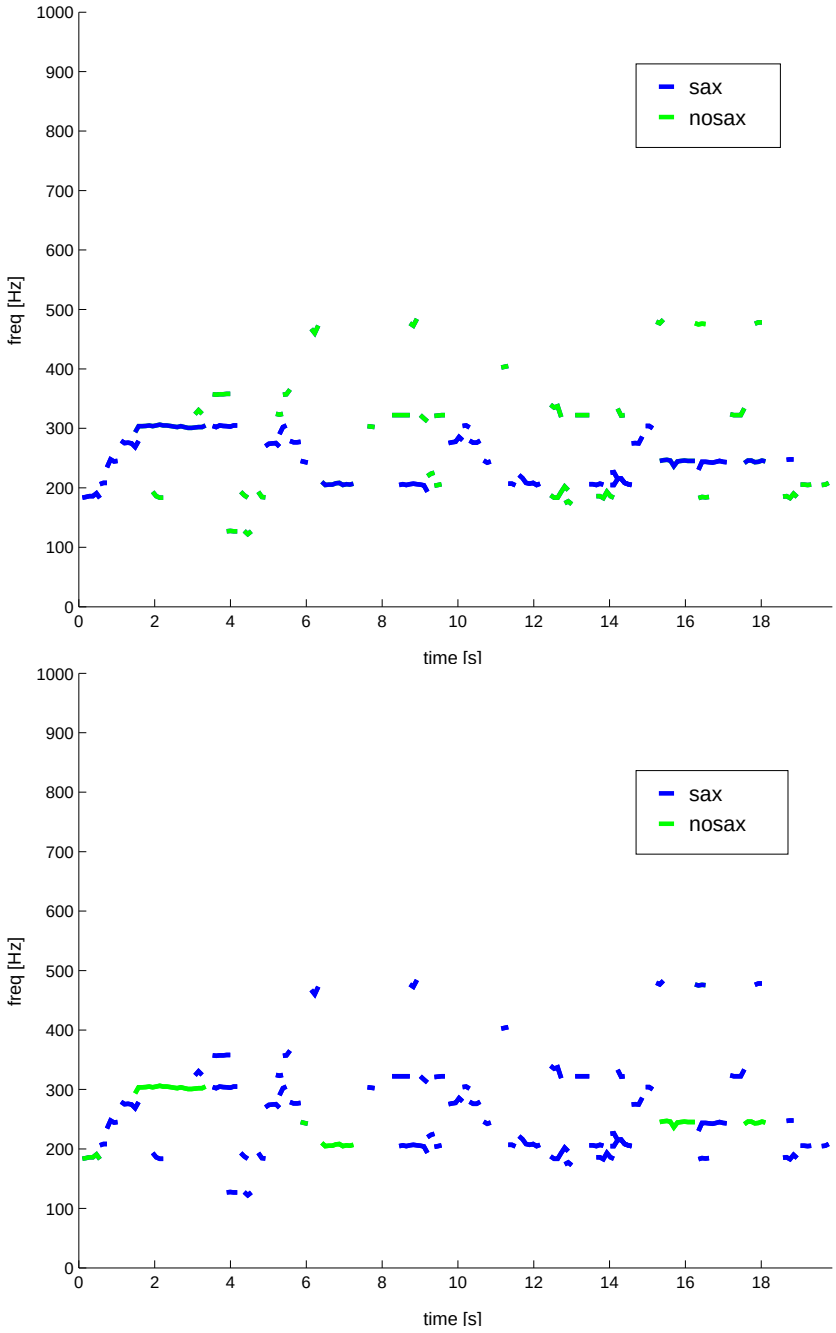

Figure 4.2: Classification for sample `mobetter`. Top: true classification. Bottom: GMM classification

Chapter 5

Conclusion

This report introduced a method for multiple F0 detection and instrument classification on single note basis.

There were not enough tests to exactly define the circumstances under which the system performance is satisfying. Generally, the saxophone has to be predominant for robust F0 detection. Long notes are easier to classify than short ones. So a slow melody is more likely to be classified correctly than a virtuous improvisation.

Major contributions are the "Cepstrally Enhanced Autocorrelation Function" (CEACF), introduced in Section 2.2.1, and the reliability indicator "Harmonic Overlap Energy Ratios" (HOERs) used to obtain spectral features introduced in Section 3.1.1.

CEACF is a useful function in multiple F0 detection because it enhances true F0 peaks in the spectrum and cancels harmonics. HOERs are an approach to the problem of information loss (amplitude, phase) at overlapping harmonics.

5.1 Suggestions for improvements

F0 detection The detection of F0 frequencies and their trajectory in time should be based on different FFTs to optimize frequency and time resolution independently.

CEACF depends on an arbitrary threshold. This could be avoided by modifying the peak detection algorithm to adapt the threshold dynamically.

Instrument recognition First of all, the training set is weak. It has too little data and, as discussed in Section 3.2, the training set is not exactly representing the data that should be classified. For training, one-class samples

were used, but the aim is to classify mixtures of all classes. This way we loose saxophone predominance in mixtures as a classification feature.

The analysis stops at note level. A further layer could group notes to melodies (i.e. by using Hidden Markov Models).

Appendix A

Usage Guide

This Chapter explains briefly the usage of the whole analysis framework. For a detailed list on function dependencies, see Appendix B.

A.1 Classifying a sample

We will step by step analyze sample `chris` in Matlab.

1. Load configuration

   ```
   >>init
   ```

2. Load models. See Section A.2 on how to obtain instrument models.

   ```
   >>load instModels
   ```

3. Load sample `chris`. If you want to analyze other recordings than the ones listed in Appendix D you have to edit `prepareData.m`.

   ```
   >>snd = prepareData('chris')
   ```

   The struct `snd` now contains the signal and some parameters.

4. Analyze the sample. This will take a while

   ```
   >>snd = analyzeAll(snd)
   ```

   F0s and their trajectories are saved to `snd`

5. Show spectrogram with detected F0s

```
>>showSpec(snd)
```

6. Classify all agents with GMM

   ```
   >>classifyAll(snd,instModels,'GMM')
   ```

7. If you want to have more information on an analysis window or agent, start

   ```
   >>evalSelect(snd,instModels,'GMM')
   ```

   Using the mouse you can select an agent or analysis window and all important analysis steps are displayed in different figures. You get even more plots by turning on debug mode

   ```
   >>sw.DebugMode=1
   ```

A.2 Building instrument models

1. Load configuration

   ```
   >>init
   ```

2. Load instrument training samples and analyze them. This takes a long time.

   ```
   >>insts=prepareModelData
   ```

3. Obtain feature vectors according to feature definitions in `getF0Features`, `getWinFeatures` and `getTemporalF0Feat`.

   ```
   >>insts=prepareModelFeatures(insts)
   ```

4. Train GMM, NN and SVM.

   ```
   >>generateModels
   ```

   Models are now saved in struct `instModels`

For feature evaluation use

```
>>evalFeatures(insts)
```

after step 3.

Appendix B

Function Dependencies

This Appendix lists all Matlab functions according to their dependencies.

analyzing samples

- init
- prepareData
  - reshapeWav
- analyzeAll
  - analyzeWin
    - ceacf
      - getQ0s
    - getF0s
    - getHarmonics
    - detectOverlaps
    - getSpacialDist
  - agencyInput
- showSpec
  - mySpecOverlay
- traceAgents
- evalSelect
  - .....

generating and using models

- prepareModelData
  - prepareData
    - .....
  - analyzeAll
    - .....
  - getHarmonicsModel
  - getSpecEnvModel
- prepareModelFeatures
  - collectFeatures
    - getAgentFeatures
      - getF0Features
        - getSpecModelMSE
        - getHarmModelMSE
      - getWinFeatures
      - getTemporalF0Feat
        - widenAgentAmp
- evalFeatures
- generateModels
  - trainGMM
  - trainNN
  - trainSVM
- classifyAll
  - classifyAgent

Appendix C

Data Structures Reference

This Appendix lists all data structs used.

- snd
  - [][] stereo — stereo signal
  - [] mono — mono signal
  - var — parameters
    - f
    - fLen
    - fMax
    - cepAx
    - t
    - tLen
    - tMax
  - an — analysis data
    - [][] Spec — spectrogram
    - [][] Ceps — cepstrogram
    - [][] SpecLiftered — liftered spectrogram
    - masks — masks to show with showSpec
      - [][] F0
      - [][] FirstF0
      - [][] Overlap
      - [][] CasaAmp
      - [][] CasaPhi
    - wins — all win structs
      - [] win
  - reSynth — re-synthesized sound
  - agents
    - [] agent

win

- [] ceps
- [] ACF
- [] CEACF
- [] ovlpInd
- [] ovlpFreq
- [] f0

f0

- freq
- ind
- a
- [] harmFreqs
- [] harmAmp
- energyAbs
- energy
- [] harmNACF — normalized acf of harmAmp
- [] harmInds
- [] reliab
- dA — intensity difference btw. L/R channel
- dPhi — delay bte. L/R channel

agent

- birth
- alive
- valid
- lastFreq
- [] trace — f0 indices for each window
- meanEnergy
- age
- startFreq
- death

- [] insts
  - snd
  - name
  - specEnv
  - harmEnv
- instModels
  - names — instrument names
  - GMM — GMM structs
    - [] specModel
    - [] tempModel
  - NN — NN structs
    - specModel
    - tempModel
  - SVM — SVM structs
    - specModel
    - tempModel
  - dataPoints — number of training vectors
    - numSpecData
    - numTempData
  - stats — parameters for feature normalization
    - specMu
    - specStd
    - tempMu
    - tempStd
  - [] specEnv — model spectral envelope for harminocs
  - [] harmEnv — model envelope for harmonic's ratios

Appendix D

Audio Data Sources

Class	Inst.	Label	Dur.	Properties
sax	ts	`sax-range`	34s	Tenor sax playing long forte notes over the whole usual instrument's range with short breaks between notes, played by the author
sax	ts	`sax-chromatic`	31s	Legato chromatic scale from low Bb to high G, played by the author
sax	ts	`sax-range-freely`	15s	Free, rather quick playing over whole range, played by the author
sax	ts	`sax-irgendwas`	11s	Free, rather quick playing, played by the author
nosax	p	`piano-range`	6s	Single piano notes in the range of interest, found on Internet
nosax	p	`amorous-piano`	28s	Piano solo taken from a recording of the song "amorous cat" (Stan Getz) from "bluetorangsch" jazz quartet

Class	Inst.	Label	Dur.	Properties
nosax	b	`nino-bass`	16s	Bass-only intro in Michael Brecker's "el niño"
nosax	b	`invocation-bass`	20s	Bass-only in Joshua Redman's "invocation"
nosax	p, b	`nino-pianobass`	75s	Piano solo with bass in Michael Brecker's "el niño"
nosax	p, b	`mobetter-pianobass`	17s	Piano solo with bass in live recording of the song "mo better blues" played by "bluetorangsch" jazz quartet recorded with one stereo microphone 2 meters in front of the stage in a bar
all	ts, p, b	`chris`	14s	First part of melody in Chris Potter's recording "gratitude". A studio recording with predominant saxophone playing long notes and soft accompaniment
all	ts, p, b	`mobetter`	20s	First part of melody in live recording of the song "mo better blues" played by "bluetorangsch" jazz quartet recorded with one stereo microphone 2 meters in front of the stage in a bar

List of Figures

List of Tables

Bibliography

[1] Eronen A. Comparison of features for musical instrument recognition. In *In Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, 2001.

[2] S. Rickard A. Jourjine and O. Yilmaz. Blind separation of disjoint orthogonal signals: Demixing n sources from 2 mixtures. In *Proceedings of the 2000 IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP2000)*, pages Volume 5, 2985–2988, 2000.

[3] J. Eggink and G.J. Brown. A missing feature approach to instrument recognition in polyphonic music. In *International Conference on Acoustics, Speech, and Signal Processing, ICASSP'03*, pages 553–556, 2003.

[4] Klapuri A. Eronen A. Musical instrument recognition using cepstral coefficients and temporal features. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2000*, pages 753–756, 2000.

[5] Masataka Goto. A robust predominant-f0 estimation method for real-time detection of melody and bass lines in cd recordings. In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2000)*, pages II–757–760, 2000.

[6] Matti Karjalainen and Tero Tolonen. Multi-pitch and periodicity analysis model for sound separation and auditory scene analysis. In *in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'99), vol. 2*, pages 929–932, 1999.

[7] Anssi P. Klapuri. Multipitch estimation and sound separation by the spectral smoothness principle. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2001.*, 2001.

[8] Aaron S. Master. Sound source separation of n sources from stereo signals via fitting to n models each lacking one source. In *Stanford University EE 391 Report*, 2003.

[9] H. Viste and G. Evangelista. An extension for source separation techniques avoiding beats. In *Proceedings of 5th International Conference on Digital Audio Effects (DAFx02)*, pages 71–75, Hamburg, Germany, September 2002.

[10] Harald Viste and Gianpaolo Evangelista. On the use of spatial cues to improve binaural source separation. In *Proceedings of 6th International Conference on Digital Audio Effects (DAFx-03)*, pages 209–213, London, UK, September 2003.