Diplomarbeitspräsentationen der Fakultät für Informatik

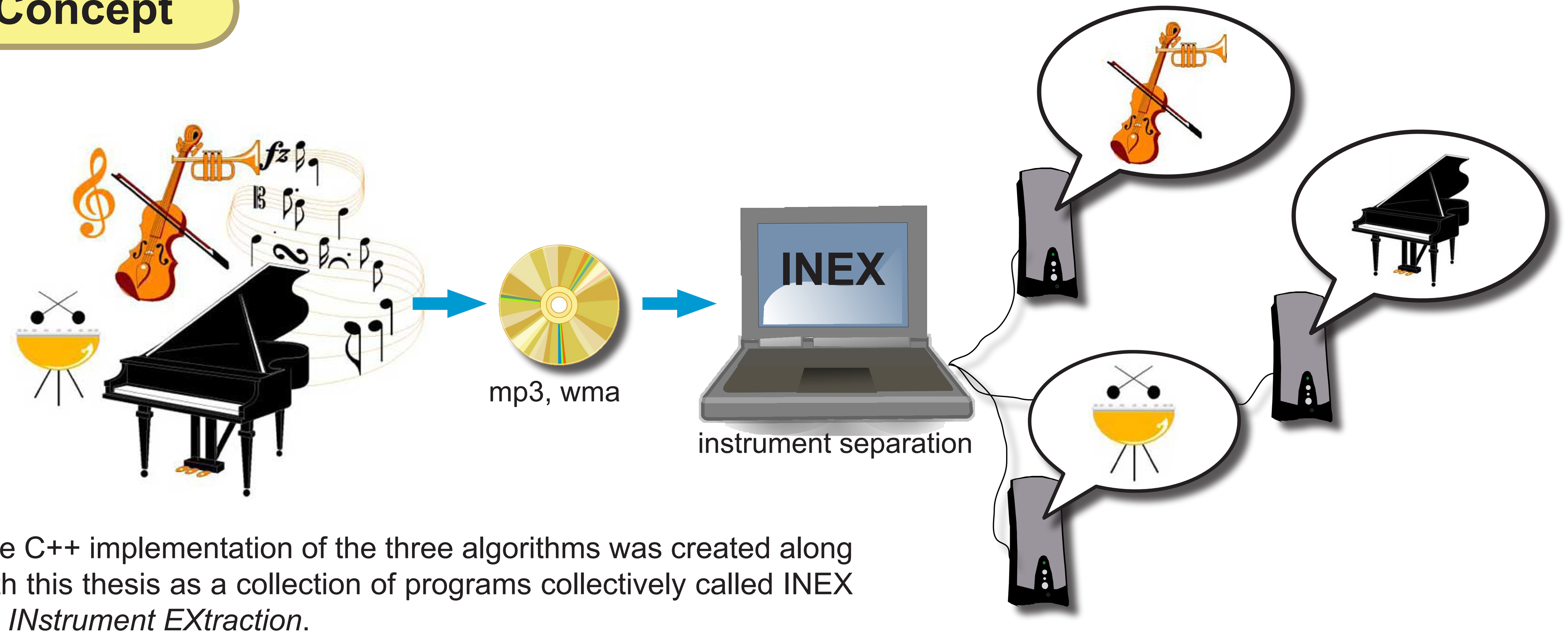
Musical Instrument Separation

Magisterstudium
Intelligente Systeme

Andrei Grecu
andrei.grecu@aon.at

Technische Universität Wien
Institut für Softwaretechnik und Interaktive Systeme
Arbeitsbereich: Information and Software Engineering Group
Betreuer: Ao.Univ.-Prof. Dr. Andreas Rauber

Concept



The C++ implementation of the three algorithms was created along with this thesis as a collection of programs collectively called INEX for *INstrument EXtraction*.

Goals and Applications

Main Goal: Given audio data containing music, separate all playing Instruments

Sub Goals

- **Template Based Approach**
  - Find all Instrument onsets
  - Find all tones
  - Synthesize each Instrument into an audio file, using its onsets and tones
- **Feature Histogram Based Approach**
  - Find all clusters in the feature histogram
  - Classify each frequency in the Input according to its cluster
  - Synthesize the frequencies of each class into a separate audio file

Applications: remixing, editing, denoising, automatic transcription

Direct Template Matching

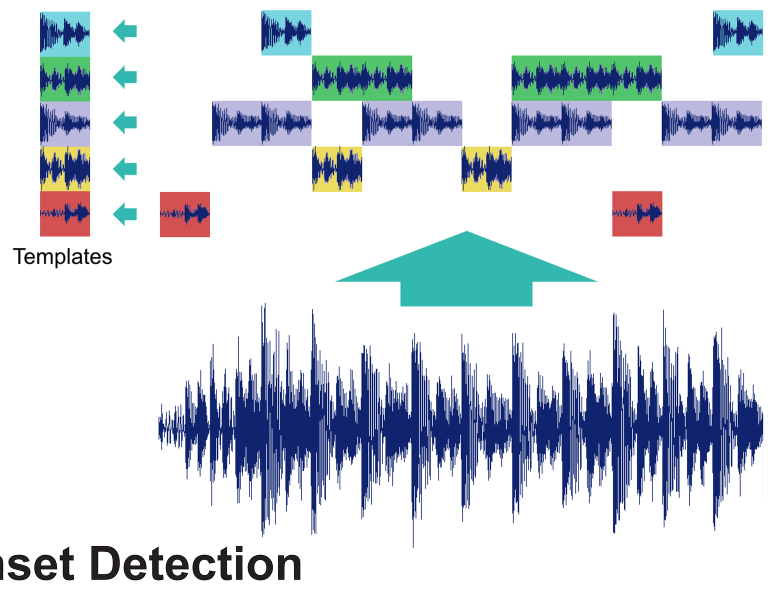
DTM

Observations

- Music consists of a limited number of tones
- Tones repeat during song duration
- Same tone sounds approximatively the same each time it is repeated

Idea

- Use templates to model tones
- Use the repetitions to separate tone from concurrently playing instruments



Onset Detection

- Find template occurences
- Needs to be at least sample-accurate
- Can be done using fast correlation
- Peaks in correlation result are filtered
  - to have values above some treshold
  - to be further apart than some minimum interval
  - to occur in both audio channels at approximatively the same time

Template adaptation

- Adapt templates to approximate the audio data at the onsets where they occur
- Use Newton's method for iterative adaptation

Resynthesis

- Render each template at its offsets
- Not implemented now: *templates belonging to the same instrument have to be grouped together before rendering*

Histogram Based BSS

HSBSS

Idea

- Use an easily visualizable feature space
- Use stereo cues for separation
- Work in the frequency domain

Histogram

- Visualization of the feature space
- Use inter-channel magnitude phase for the x-axis
  - 0° means frequency exists only in right channel
  - 45° means equal magnitude in both channels
  - 90° means frequency exists only in left channel
- Use inter-channel time shift for the y-axis
- Use magnitude as intensity of each bin
- Use frequency as color (red ~ low freq., blue ~ high freq.)

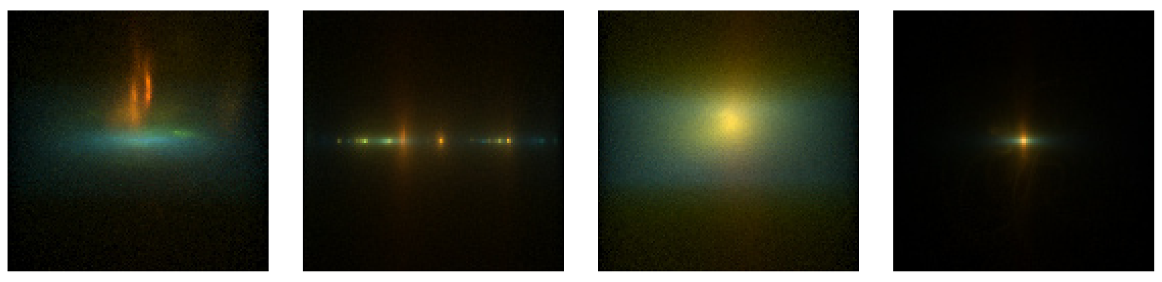


Fig. 1: a) good histogram, b) no time-shift information, c) high reverberation, d) mono with stereo sfx

Clustering

- Use a *radial basis function network* (RBFN) to estimate colors
- Segment histogram using decision boundaries from trained RBFN
- Classify frequencies according to segment they fall into

Fig. 2: a)-d) RBFN segmentation of correpsonding histograms shown in Fig. 1a-1d

Resynthesis

- Create a new spectrum for each class consisting only of its associated frequencies
- Transform spectra back to time domain

Problems

- Time shifts become ambiguous above a certain frequency
- Modern stereo recordings have little inter-chan. time shift information
  - Histogram practically becomes one line (Fig. 1b, 2b)
- Reverberation causes cluster smearing (Fig. 1c, 2c)
- RBFN shows poor estimation performance

Iterative Template Matching

ITM

Observations

- Direct Template matching has difficulties finding the correct onsets

Idea

- Same idea as DTM
- This time let onset vector self-organize

Steering Vector

- Let onsets happen at each time sample
- Onsets become weights, therefore onset vector becomes steering vector
- Good solutions contain only few non-zero weights

Algorithm

- Iterate over synthesizing and adaptation steps until convergence
- **Synthesizing**
  - Convolve steering vectors with templates
- **Adaptation**
  - Minimize synthesizing error by adapting steering vector and template
  - Use *resilient backpropagation* (RPROP)
  - Use additional non-sparseness error function for steering vector to encourage sparse solutions

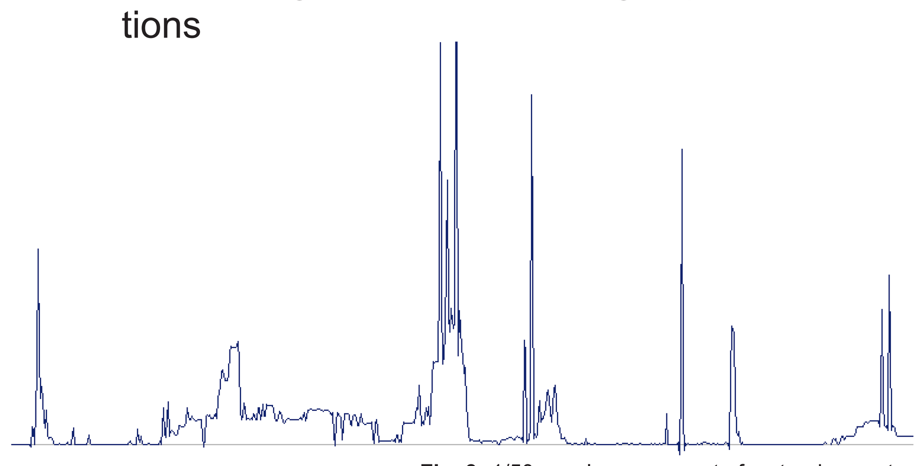


Fig. 3: 1/50 sec. long segment of a steering vector

Refinements

- Calculate templates exactly, use RPROP only for steering vector
- Work on upsampled input for less high frequency damping

Future Work

Direct Template Matching

- Implement missing tone clustering part
- Move concept to frequency domain, better decorrelation expected
- Redesign initialization procedure

Histogram Based BSS

- Use a time-shift disambiguation heuristic to minimize cluster spreading
- Detect number of clusters automatically
- Find new features (dimensions) for the histogram

Iterative Template Matching

- Find better non-sparseness cost function

Evaluation

Used corpora: BASS-dB*, IS, ISMIRgenre*, RWC*

Our own corpus, the *Instrument Separation* (IS) corpus contains:

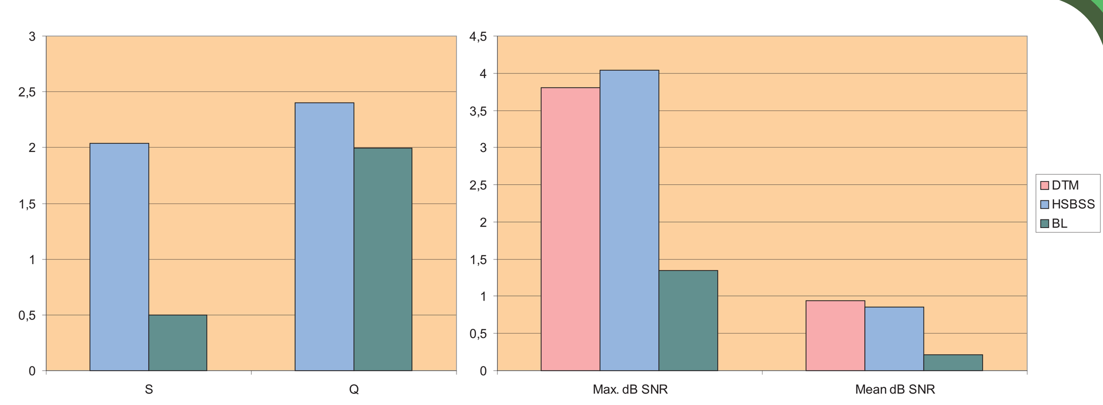
- Binaural recordings
- Recordings with reference tracks
- Module (MOD) files, decomposed into individual tracks

Subjective evaluation criteria

- Separation performance „S" (0-5)
- Remaining signal quality „Q" (0-5)

Objective error measure: *signal to noise ratio* (SNR)

Baseline (BL), produces sum and difference channel (L+R, L-R)



Results

- HSBSS separation performance 4x higher than baseline
- Baseline quality score almost as high as HSBSS
- DTM and HSBSS almost equal in terms of dB SNR
- Best separation values per title have 4x higher SNR than the mean values
- HSBSS also has objectively 4x higher SNR performance than baseline

*) Only parts of the corpus used