An Automatic Blues Band

Andrew Brothwell[‡] John ffitch[*]

[‡]University of Bath
[*]Codemist Ltd

LAC Köln, Feb 2008

The Problem

The Problem for Blues Guitarists

- practising and playing their instruments on their own can become dull and unfulfilling
- 'lead' instruments need to practice with a full band

So.....

- Find a band of competent musicians
- Use an artificial accompaniment

Earlier Work on Accompanist

There are many examples of automated band; **Band-in-a-Box** for example.

Does the band know the score?

Many attempts use Artificial Intelligence methods.

Our starting point was the work of Bryson which used a subsumption strategy

BUT

A Simple Accompanist?

Our desires are simply stated:

- a system that 'listens' by way of a simple microphone
- the band plays in time
- the band plays in tune
- cheap, basic hardware and software (we use Csound)

A Simple Blues Accompanist

We are restricting ourselves to twelve-bar blues.

- Lead Instrument (Acoustic/Electric Guitar, Piano, Mouth Organ etc)
- Rhythm Guitar
- Bass Guitar
- Drum Kit

There are four beats to each bar and twelve bars to each part. Each of the twelve bars will be played with a backing chord relative to the key of the song, in the order of: I, I, I, I, IV, IV, I, I, V, IV, I, I

Pitch Detection

We want the “band” to play in tune with the leader rather than *vice versa*.

So we need to detect the pitch of the lead. Currently we use average magnitude difference function (AMDF) which seems acceptable.

Tempo Detection

We want the “band” to play in time with the leader.

We have to establish the speed that the musician is playing at in beats per minute;
AND be able to predict when future beats will occur.

This requires some onset detection, which is not currently available in Csound.

During the count-in RMS is clearly sufficient to set initial tempo.

Staying in Time

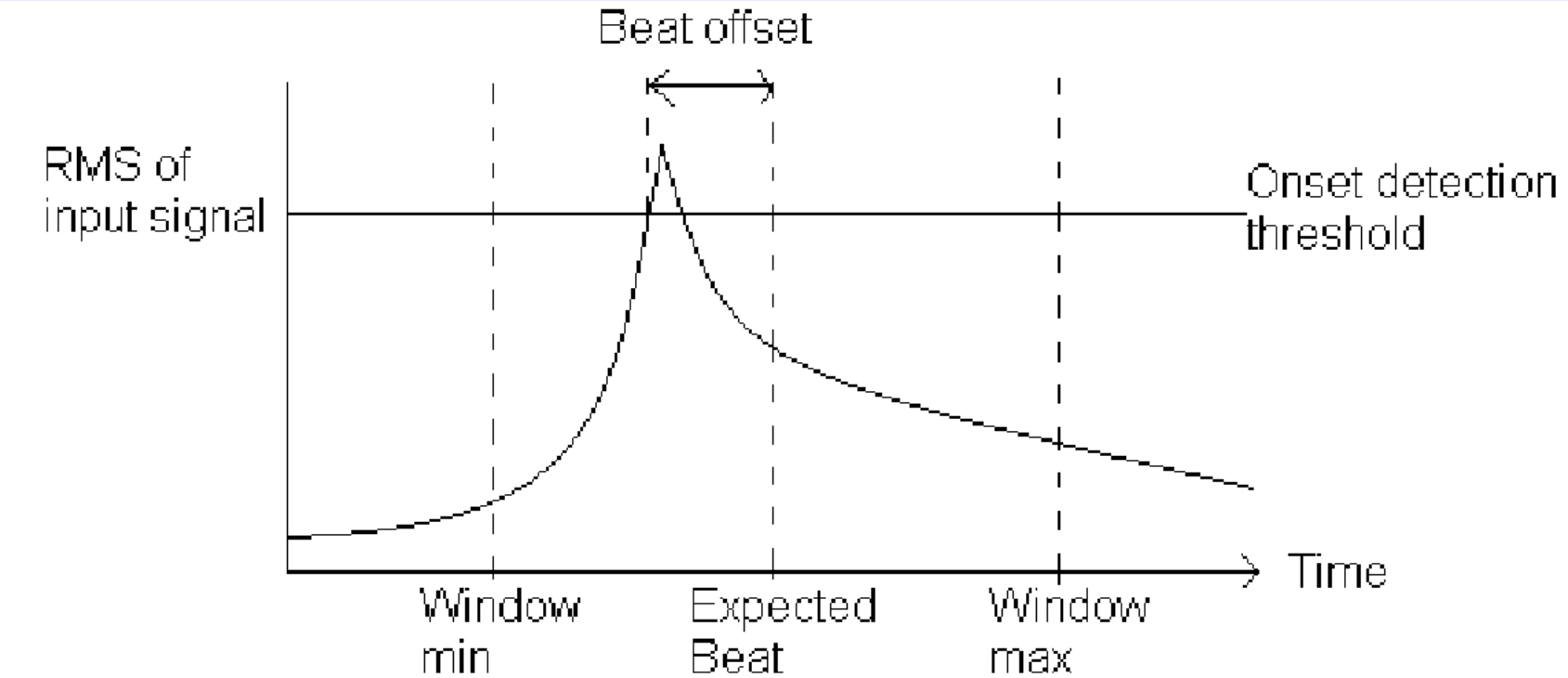
We want the “band” to play in time with the leader always.

Need a smooth reactive adjustment to tempo speed up or slow down.

After some experimentation we used a simple predictive model

```
If (Time = WindowStart AND
    InputRMS > Threshold) // ringing
 STOP CHECKING TILL NEXT WINDOW
ElseIf (WindowStart < Time < WindowEnd
        AND InputRMS > Threshold)
 RealBeatTime = Time
 TempoError = RealBeatTime -
              BelievedBeatTime
 STOP CHECKING TILL NEXT WINDOW
EndIf
```

Staying in Time



Beat Correction Algorithm

Staying in Time

Smooth change is achieved by a low-pass filter on measured speeds.

This is in general agreement with detailed studies of performance.

Staying in Tune

It was found that the system would sometimes play very slightly out of tune after the count-in. Only by a very small amount, but the system was sharp when an acoustic guitar was used as an input.

This system however was tuned to initial count-in, played hard and so sharp.

So needed to detect pitch, and adjust in performance, again with use of expectations.

```
For Every Beat
 If (BelievedRoot*0.97 <
     CurrentPitch<BelievedRoot*1.03)
   BelievedRoot =
      (BelievedRoot+CurrentPitch)/2
 EndIf
```

Stopping Together

This was harder than expected. Humans use visual clues as well as aural ones.

During a practise with one of the authors' band, an attempt was made to ascertain the processes that musicians go through when listening for an ending. When the band tried to play with their eyes closed they lacked their usual tightness and had moments of uncertainty, especially the transitions between loud and soft parts which were a challenge. They did manage to finish together sloppily.

We compromised with the band stopping when the leader stopped, and they reached the end of a 12-bar section.

In Use

Musicans criticised the instruments, but were reasonably happy with the band.

So, we need to demonstrate this with examples.....

Example of speed changes (45s)
Guitarist with Band (2m12s)
Second Guitarist in different style (1m55s)
A Song (3m)

We claim that we achieved our aim with no AI, low cost and low computational requirements. Your turn!

Acknowlegment

This work formed the final year project of Andrew Brothwell.
The full report is available

`http://www.cs.bath.ac.uk/department/technical-series-database/technical-report-dissertation.html`

Software available on the web at

`http://dream.cs.bath.ac.uk/software/acompanist.csd`