Score Following

Making an artificially intelligent piano accompanist

Anna Jordanous

24/07/2007

Score Following - Making an artificially intelligent piano accompanist

- What is Score Following?
- When is Score Following used in real-life situations?
- Making an artificially intelligent Score Follower

  - Using Hidden Markov Models

  - How my Score Follower works
- Work still to be done

What is Score Following?

Score
The written music that a musician reads when they play music

Score Following
When you follow a musician’s playing by tracking them through the score

When do we use Score Following?

Imagine you are at a concert. A saxophonist is performing a solo piece, with a piano player providing accompaniment.

The piano player listens to what the saxophonist is playing, to ensure their accompaniment matches the saxophonist.

The saxophonist may occasionally not perform the piece exactly as it is written in the score. In these cases the piano player must adjust their accompaniment.

Why might the saxophonist not perform the piece exactly as it is?

Changes may be added by mistake…

- Plays the wrong note
- Adds extra notes
- Skips notes out
- Gets lost
- Speeds up or slows down

Why might the saxophonist not perform the piece exactly as it is written?

… Or changes may be added deliberately, to add their own interpretation of the music

- Adds embellishments such as trills, to ‘decorate’ the notes
- Speeds up or slows down deliberately, for musical effect

How could we make an artificially intelligent Score Follower?

A number of approaches have been tried.

Many research efforts are currently focused on using **Hidden Markov Models** (e.g. Raphael 1999 onwards, IRCAM 2001 onwards)

My hypothesis is that:

> *Using an HMM to model a musical score is an efficient and practical way to implement score following.*
>
> *In particular, it lends well to providing real-time accompaniment to a human performer.*

What is a Hidden Markov Model?

A Hidden Markov Model (HMM) is a way of modelling real-world systems using probabilities.

A system modelled with an HMM can be considered to be in one of a finite number of states at any given time. We can determine what state the system is currently in by examining recent outputs from the system (‘observations’).

E.g: a sequence of coin tosses, using either a biased or normal coin (but it is unknown to us which coin is being used).

STATES:	‘biased coin tossed’, ‘normal coin tossed’
OBSERVATIONS:	‘Head’, ‘Tail’.

At any one time, which coin is being tossed?

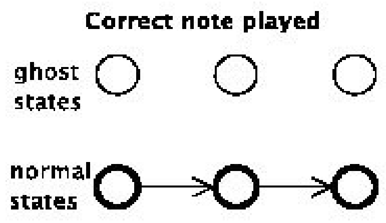
Score Following using HMMs

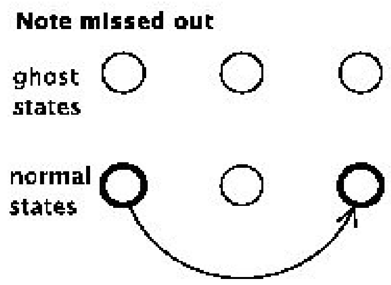
A music score is divided into a sequence of events (e.g. one note = one event)

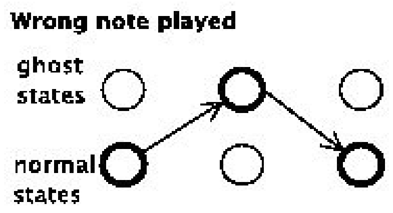
The Score Follower uses an HMM decoding algorithm (e.g. Viterbi) to determine what state the performer is most likely to be in at that time

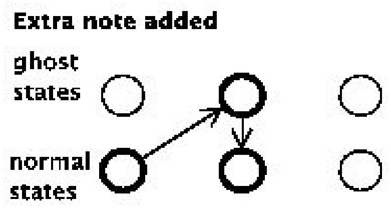
For each event in a score, there are 2 corresponding HMM states *(IRCAM '03)*

- 'Normal' state – when the performer has played the expected note
- 'Ghost' state – when the performer has played an unexpected note

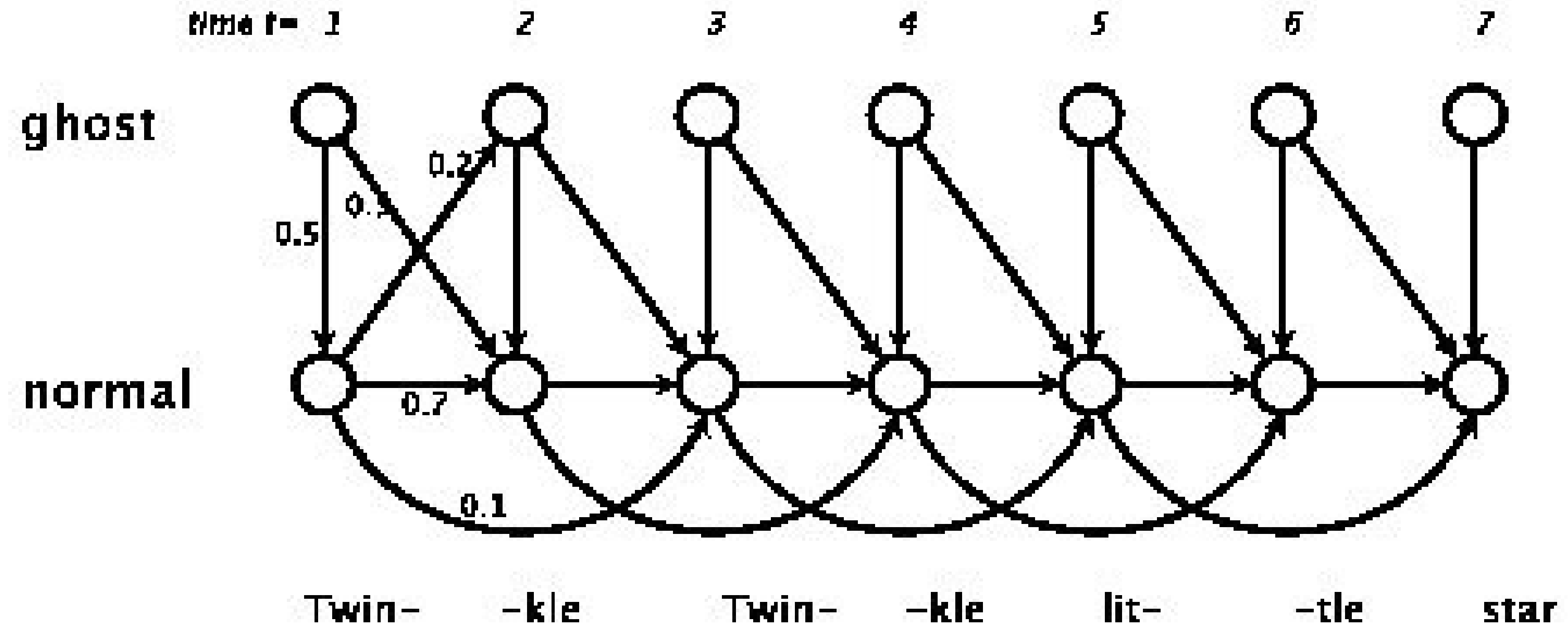








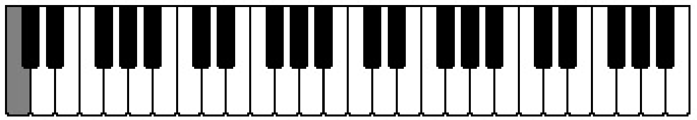
An HMM model of a music score

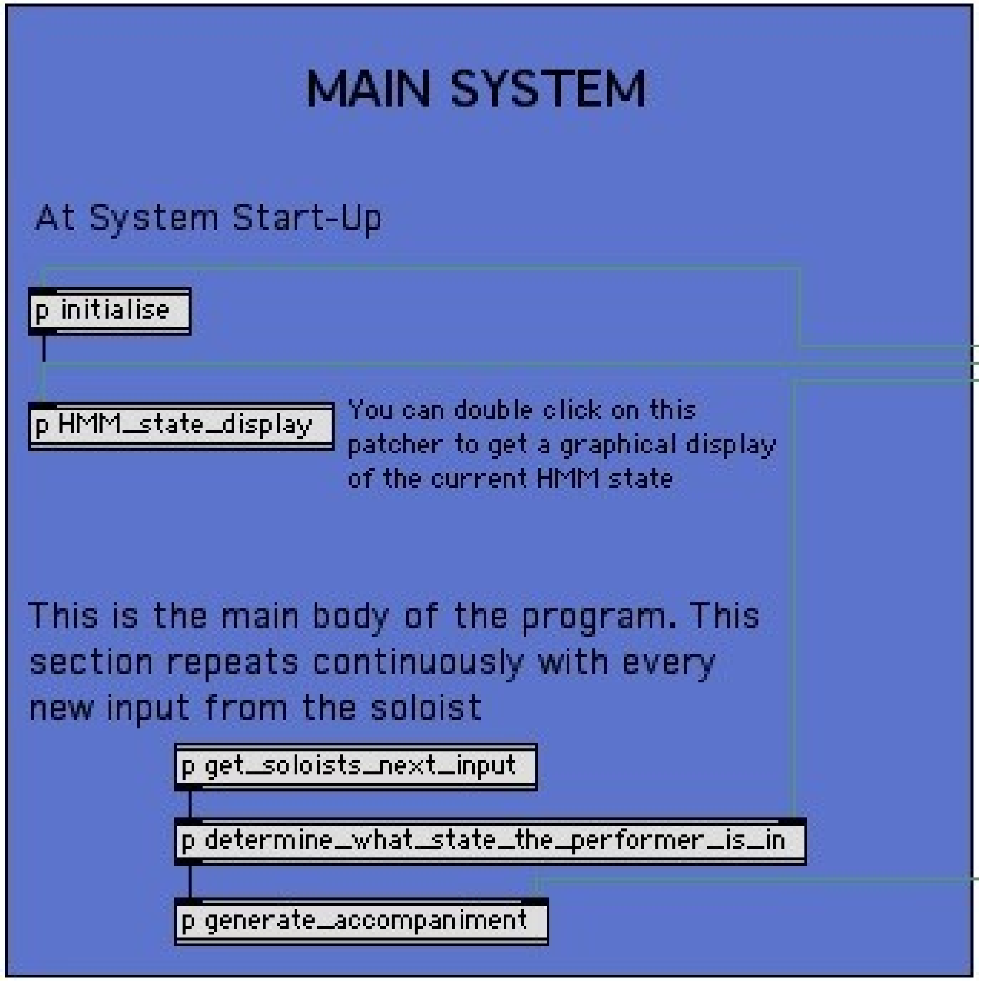


My Score Following system

Written in Max/MSP (a real-time music interaction programming environment)

Runs on a computer that is attached to a MIDI keyboard: a keyboard which sends ands receives data to/from the computer in the musical format MIDI (numeric messages about the note pitch, volume etc)





MAIN SYSTEM

At System Start-Up

p initialise

p HMM_state_display

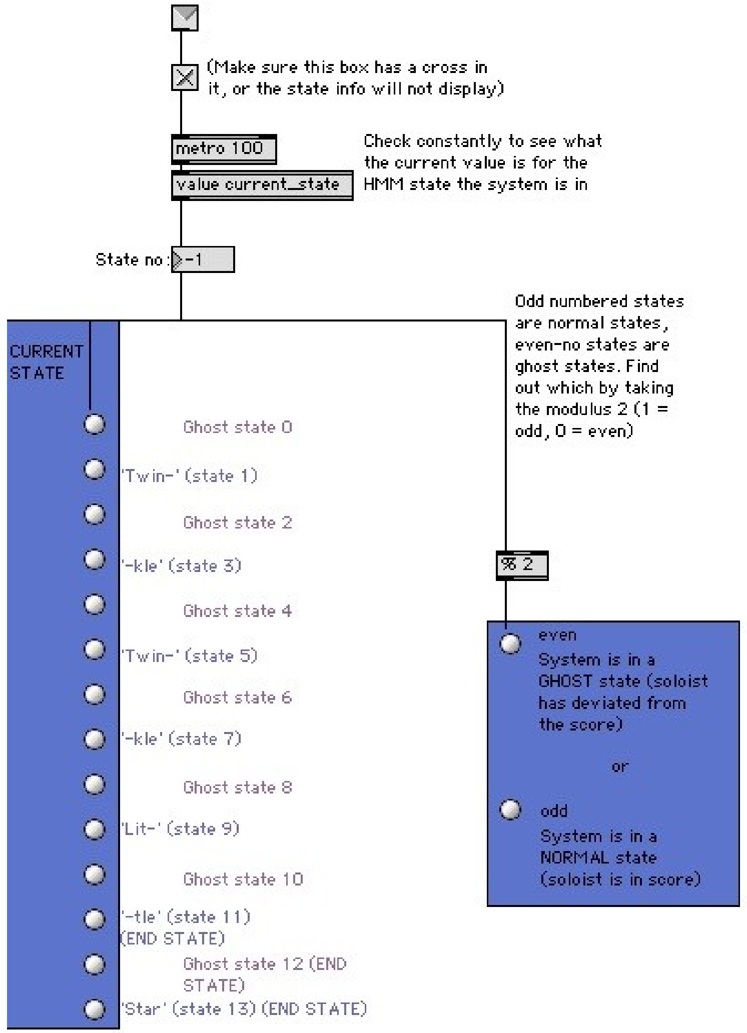
You can double click on this patcher to get a graphical display of the current HMM state

This is the main body of the program. This section repeats continuously with every new input from the soloist

p get_soloists_next_input

p determine_what_state_the_performer_is_in

p generate_accompaniment



(Make sure this box has a cross in it, or the state info will not display)

metro 100

value current_state

Check constantly to see what the current value is for the HMM state the system is in

State no : -1

Odd numbered states are normal states, even-no states are ghost states. Find out which by taking the modulus 2 (1 = odd, 0 = even)

% 2

CURRENT STATE

Ghost state 0

'Twin-' (state 1)

Ghost state 2

'-kle' (state 3)

Ghost state 4

'Twin-' (state 5)

Ghost state 6

'-kle' (state 7)

Ghost state 8

'Lit-' (state 9)

Ghost state 10

'-tle' (state 11) (END STATE)

Ghost state 12 (END STATE)

'Star' (state 13) (END STATE)

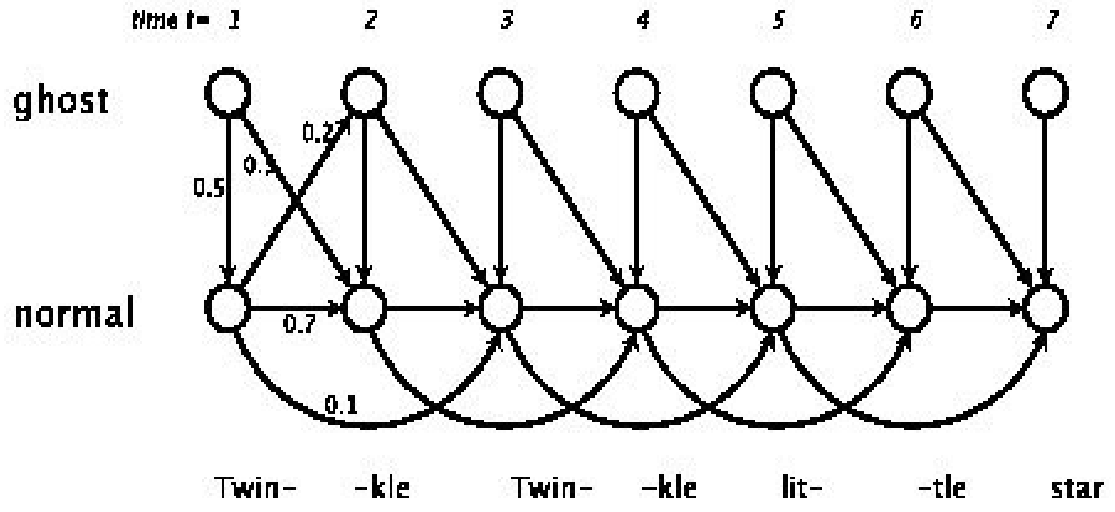
even
System is in a GHOST state (soloist has deviated from the score)

or

odd
System is in a NORMAL state (soloist is in score)

Details of how my Score Following system works

The system is programmed to provide accompaniment for the ‘Twinkle Twinkle Little Star’ melody (which is 7 notes long), and uses the HMM from earlier.



Details of how my Score Following system works

get_soloists_next_input: Extracts the new information from MIDI keyboard.

determine_what_state_the_performer_is_in: Runs a series of procedures:

1. Convert the actual pitch of the note into one of 12 possible observations (0 if a ‘C’ has been played, 1 if a ‘C sharp’ has been played, etc).
2. Adds this observation to the list of observations seen so far and extracts the three most recent observations.
3. Performs the Viterbi decoding algorithm with these three observations to decode which HMM state the soloist is in (i.e. where they are in the score).

generate_accompaniment: Looks up the HMM state in an attribute-value pair, to find which notes to play as accompaniment
E.g: <*1, [48, 52, 55]*> - ‘In state 1, play MIDI notes 48, 52, 55’ (C major chord)

Timetable of remaining work to be done

Week commencing 23/07/07.
Finish implementing the Viterbi algorithm in my Max/MSP system, and tweak HMM parameters so that the Score Follower works properly.
Then start work on a model with more complex accompaniment: .

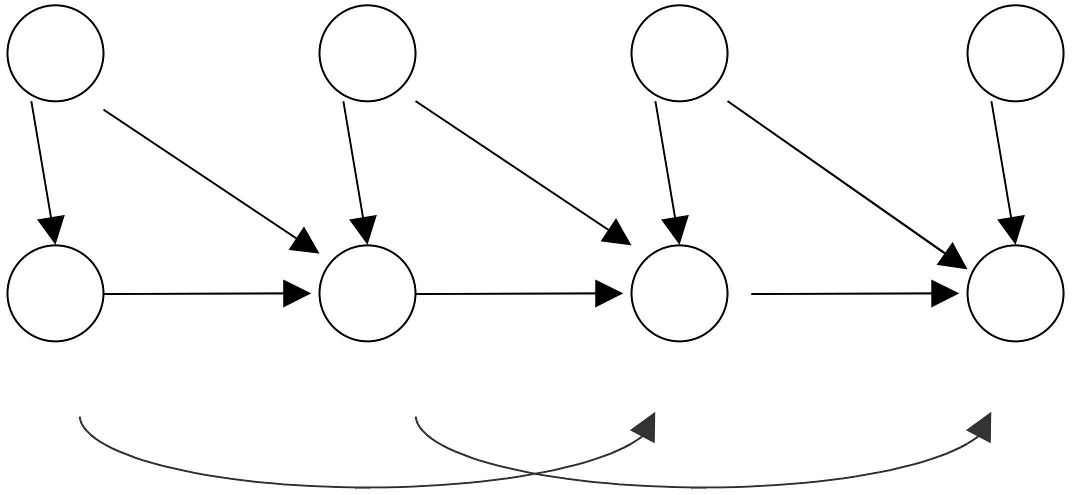
E.g: <*1, [[<u>0.5</u>, 48, 52, 55],[<u>0.5</u>, 55, 59, 62]]*> -
'In state 1, play MIDI notes 48, 52, 55' (C major chord) for <u>half a beat</u>,
then play MIDI notes 55, 59, 62 (G major chord) for <u>half a beat</u>

So <u>some timing information</u> will need to be taken from the observed events.

Week commencing 30/07/07.
Continue to work on this more complex model. If time, look at this problem:
Can my system automatically extract (learn?) the score and HMM parameters from a MIDI file in the right format? Currently HMM is programmed in by hand.

Conclusions

- Automated musical accompaniment, or score following, can be implemented using Hidden Markov Models (HMMs) to track what state a performer is in.
- In my system, for each note that the performer is expected to play, there are two corresponding states that the soloist could be in: the ‘normal state’ and the ‘ghost state’.



Conclusions

- Automated musical accompaniment, or score following, can be implemented using Hidden Markov Models (HMMs) to track what state a performer is in.

- In my system, for each note that the performer is expected to play, there are two corresponding states that the soloist could be in: the ‘normal state’ and the ‘ghost state’.

- My system determines which HMM state the soloist is currently in, by analysing what they have just played against a specified score. It then plays the appropriate accompaniment for that state.