

**Score Following:  
An Artificially Intelligent  
Musical Accompanist**

*Anna Jordanous*



Master of Science  
Artificial Intelligence  
School of Informatics  
University of Edinburgh  
2007

# Abstract

Score Following is the process by which a musician can be tracked through their performance of a piece, for the purpose of accompanying the musician with the appropriate notes. This tracking is done by following the progress of the musician through the score (written music) of the piece, using observations of the notes they are playing.

Artificially intelligent musical accompaniment is where a human musician is accompanied by a computer musician. The computer musician is able to produce musical accompaniment that relates musically to the human performance.

Hidden Markov Models (HMMs) are a stochastic modelling tool that can be used to represent real-world systems in a variety of domains.

This project discusses how HMMs can be used in the domain of Score Following and describes the construction and evaluation of a score following system that uses HMMs to implement score following. It explores the hypothesis that using an HMM to represent a musical score is an efficient and practical way to implement score following, and that in particular this method is suitable for providing real-time accompaniment to a human performer.

The score followers developed during this project are tested and compared against other score following systems and against human musicians. The resulting performances support the project hypothesis to a large extent.

# Acknowledgements

I would like to thank my supervisor Alan Smaill, for his guidance and help throughout this project.

Also, thanks to David Murray-Rust in the School of Informatics, and Kinnell Anderson, Michael Edwards and Peter Nelson in the Music Department at the University of Edinburgh, for their advice and practical help.

Christopher Raphael was kind enough to respond to some questions I had when starting this project and additionally gave me some advice based on his considerable experience in score following research.

Matt Baker and Michael Wood provided technical assistance which helped me a great deal in the early stages of development.

Michael Wood, Rosalin Cooper, Adam Apostoli and Mike Thorpe provided valuable feedback during the testing phase of this project.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

*(Anna Jordanous)*

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	What is score following . . . . .	1
1.1.1	Score following in real-life domains . . . . .	1
1.1.2	How an artificially intelligent score follower could be useful . . . . .	2
1.2	The use of Hidden Markov Models in score following . . . . .	3
1.3	Project Hypothesis . . . . .	4
1.4	Project Aims and Objectives . . . . .	4
1.5	What was achieved during this project . . . . .	4
1.6	Terms used in this document . . . . .	5
1.7	Outline of this document . . . . .	6
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	Dynamic programming and pattern matching . . . . .	8
2.2	Important developments in score following research during the 1980s and early 1990s . . . . .	9
2.2.1	Real-time scheduling . . . . .	9
2.2.2	Probability density functions . . . . .	9
2.3	Dynamic time warping . . . . .	9
2.4	Hidden Markov Models . . . . .	10
2.4.1	General theory of Hidden Markov Models . . . . .	11
2.4.2	Score following using Hidden Markov Models . . . . .	12
2.4.3	Implementation of score followers using Hidden Markov Models . . . . .	14
2.5	The current state of research in score following . . . . .	15
2.6	How this work fits into the research context . . . . .	15
<b>3</b>	<b>Design</b>	<b>16</b>
3.1	Requirements . . . . .	16
3.2	Practical constraints . . . . .	16
3.2.1	Time available for this project . . . . .	16
3.2.2	The type of musical input to be used . . . . .	17
3.2.3	Hardware resources available for this project . . . . .	17
3.3	The conceptual design of my score following system . . . . .	18
3.3.1	RECEIVE INPUT FROM PERFORMER . . . . .	18
3.3.2	ESTIMATE WHAT HMM STATE THE PERFORMER IS IN, USING THE VITERBI ALGORITHM . . . . .	19
3.3.3	PRODUCE THE APPROPRIATE ACCOMPANIMENT . . . . .	19

3.3.4	DISPLAY STATE TRACKER ON SCREEN FOR INFORMATION . . . . .	19
3.4	Major conceptual design decisions . . . . .	20
3.4.1	Choosing the best features for state representation . . . . .	20
3.4.2	What to use in the set of possible observations . . . . .	20
3.4.3	Restricting the soloist's playing to be monophonic . . . . .	21
3.4.4	Comparison of anticipative and reactive design strategies . . . . .	22
3.4.5	Adapting the accompaniment when the score follower has identified a large change in the soloist's current score location . . . . .	24
3.4.6	Encoding musical knowledge in the HMM probabilities . . . . .	25
3.4.7	The action to take when more than one state is judged equally probable by the HMM . . . . .	25
3.4.8	Learning the most accurate HMM probabilities through training . . . . .	26
3.4.9	Representing notes of different lengths using HMM states . . . . .	27
3.4.10	Tempo extraction (beat tracking) . . . . .	28
3.5	Major practical design decisions . . . . .	30
3.5.1	How MIDI files should be used in the score follower . . . . .	30
3.5.2	Output format for accompaniment . . . . .	31
3.5.3	The type of pieces used for my score follower . . . . .	32
3.5.4	Which development tools to use . . . . .	32
3.5.5	Existing HMM packages . . . . .	35
3.5.6	Synchronising the start of the soloist and the score follower . . . . .	36
3.5.7	Responding to the volume at which the soloist is playing . . . . .	37
<b>4</b>	<b>Implementation of the Score Followers</b>	<b>38</b>
4.1	Stages of development . . . . .	39
4.2	Major sections of the score follower system . . . . .	43
4.3	Problems encountered during implementation . . . . .	44
4.3.1	Conceptual/theoretically-related problems . . . . .	44
4.3.2	Technical problems . . . . .	46
4.4	Experimentation with my system during development . . . . .	47
4.4.1	Fitting an HMM to score following . . . . .	48
4.4.2	Number of observations used for Viterbi . . . . .	49
4.4.3	Setting a probability thresholds for the estimation of the next state . . . . .	50
4.4.4	Finding the most effective beat tracking parameters . . . . .	50
4.4.5	Performing the viterbi algorithm off-line, to generate pre-calculated probabilities for longer scores . . . . .	52
4.4.6	Reducing latency issues by only examining local states as potential next states . . . . .	53
<b>5</b>	<b>Testing and Evaluation of System Performance</b>	<b>55</b>
5.1	Objective testing . . . . .	56
5.1.1	Objective testing methodology . . . . .	56
5.1.2	Results of objective evaluation tests . . . . .	59
5.1.3	Analysis of objective evaluation test results . . . . .	61
5.2	Subjective testing . . . . .	64
5.2.1	Subjective testing methodology . . . . .	64

5.2.2	Observations arising from Tester 1 . . . . .	65
5.2.3	Observations arising from Tester 2 . . . . .	67
5.2.4	Observations arising from Tester 3 . . . . .	68
5.2.5	Observations arising from Tester 4 . . . . .	72
5.2.6	General conclusions from tester feedback . . . . .	72
<b>6</b>	<b>Discussion and Conclusions</b>	<b>76</b>
6.1	Capabilities of the score followers developed in this project . . . . .	76
6.2	Meeting the specified requirements . . . . .	77
6.2.1	Evaluation of project hypothesis . . . . .	78
6.3	What could be improved upon or done differently, in hindsight . . . . .	80
6.4	Suggestions for possible future work . . . . .	82
6.5	Concluding remarks: What has this work achieved . . . . .	84
	<b>Bibliography</b>	<b>85</b>
	<b>Appendices</b>	<b>88</b>
<b>A</b>	<b>Repertoire for my score follower</b>	<b>88</b>
<b>B</b>	<b>An example of the Hidden Markov Model probabilities: Twinkle Twinkle Little Star</b>	<b>93</b>
<b>C</b>	<b>Technical program detail</b>	<b>97</b>
<b>D</b>	<b>Test melodies</b>	<b>109</b>
<b>E</b>	<b>Glossary of terms used for musical ornamentations and embellishments</b>	<b>113</b>

# List of Figures

2.1	Two sequences that DTW treats as very similar . . . . .	10
2.2	Two sequences that DTW treats as very different . . . . .	10
2.3	Errors for states . . . . .	14
4.1	HMM for <i>Twinkle Twinkle Little Star</i> , with one mistake allowed at a time	39
4.2	HMM for <i>Twinkle Twinkle Little Star</i> , with handling of multiple concurrent mistakes . . . . .	41
4.3	Transitions representing the SKIP error (where a scored note is missed out) are only necessary for a small number of future <i>normal</i> states . .	48
5.1	<i>Results of objective evaluation tests</i> . . . . .	60
A.1	<i>Twinkle Twinkle Little Star</i> extract with simple monophonic bass line accompaniment . . . . .	89
A.2	<i>Twinkle Twinkle Little Star</i> extract with simple chordal accompaniment	89
A.3	<i>All I Ask of You</i> extract with simple chordal accompaniment . . . . .	90
A.4	<i>All I Ask of You</i> extract with more complex chordal accompaniment .	91
A.5	<i>Danse Macabre</i> extract . . . . .	92
C.1	Program structure by patcher (function) . . . . .	100
C.2	My score follower on startup . . . . .	101
C.3	Program initialisation patcher . . . . .	102
C.4	How the score follower processes new input from the soloist . . . . .	103
C.5	How the score follower determines what the next state should be (part 1)	104
C.6	How the score follower determines what the next state should be (part 2)	105
C.7	The viterbi algorithm as implemented in the score follower . . . . .	106
C.8	How the score follower generates accompaniment (part 1) . . . . .	107
C.9	How the score follower generates accompaniment (part 2) . . . . .	108
E.1	Acciaccatura . . . . .	113
E.2	Appoggiatura . . . . .	114
E.3	Grace notes . . . . .	114
E.4	Inverted Mordent . . . . .	114
E.5	Slide . . . . .	114
E.6	Trill . . . . .	114
E.7	Turn . . . . .	114



# List of Tables

5.1	Overall Summary Results . . . . .	59
B.1	state_transitions.txt: The probabilities associated with transitions from one state to another ( $A$ ) . . . . .	94
B.2	emission_probabilities.txt The probabilities of seeing each observation in a state ( $B$ ) . . . . .	95
B.3	initial_probs.txt: The probability of starting in each state ( $\pi$ ) . .	96

# Chapter 1

## Introduction

This project addresses what score following is and how it can be implemented using Hidden Markov Models. This chapter sets the context for this project and outlines what the project achieves.

### 1.1 What is score following

In a musical situation, a **score** is the written music that a musician reads when they play music. **Score following** is the process where a musician follows another musician's playing of a musical piece, by tracking their progress through the score of that piece.

#### 1.1.1 Score following in real-life domains

Real-life score following is best illustrated by an example.

Consider a flautist<sup>1</sup> who is performing a solo piece at a concert, with a piano player providing accompaniment. The piano accompanist listens to what the flautist is playing, to ensure their accompaniment matches the flautist. The flautist may occasionally not perform the piece exactly as it is written in the score. In these cases the piano player must adjust their accompaniment, to synchronise with the flautist.

There are several reasons why the flautist may not perform the piece exactly as written.

Changes may be added by mistake:

- A wrong note is played

---

<sup>1</sup>flute player

- Extra notes are added
- Scored notes are missed out
- The flautist loses their place in the music or starts playing from the wrong point in the score (in particular this is a concern if the flautist is playing from memory rather than having the music in front of them)
- The flautist's tempo<sup>2</sup> speeds up or slows down unintentionally

Also changes may be added deliberately, as the flautist adds their own interpretations to the music:

- The flautist adds embellishments such as trills<sup>3</sup>, to 'decorate' the notes
- The flautist's tempo speeds up or slows down deliberately, for musical effect
- The piece being played may have *rubato*<sup>4</sup> or free/improvised sections, where the flautist is free to vary the tempo and notes played according to their own choice.

### 1.1.2 How an artificially intelligent score follower could be useful

There are two main application areas where automated score following is useful: in electronic music and as an aid to human musicians.

#### 1.1.2.1 Electronic music

Many musical performances consist of a soloist playing a melody from a score, with the given accompaniment provided by another instrumentalist. In the case of some electronic music, the accompanying parts can be computer-generated<sup>5</sup>. The accompaniment supports the soloist and adds underlying harmonies.

#### 1.1.2.2 As an aid to human musicians

Musicians do not always have access to accompanists for practice purposes. Other common problems are that their accompanist may not be available when needed for

---

<sup>2</sup>speed at which the music is being played

<sup>3</sup>See Appendix E

<sup>4</sup>variable and flexible tempo

<sup>5</sup>For example Pierre Boulez: *Repons*, or Philippe Manoury: *Sonus ex machina*

performance, or that the accompanist does not have the technical ability necessary to provide adequate accompaniment.

A possible solution involves the accompaniment to be generated automatically by a computer or recording, as happens in some electronic music.

Many musicians practise playing over recorded or computer-generated accompaniment where the accompaniment is static, i.e. it is fixed and will not change from one playing to another. This means, though, that the musician adapts their performance to match the recording. It is more natural for the musician if the accompaniment adapts to fit the performer. Raphael (2001b) describes this as moving from “*music minus one*” to “*music plus one*”.

To dynamically synchronise the accompaniment with the performance by the musician, it would be necessary for the accompanist to track the performer in some way through the score of the piece as they play.

This may become complicated if the performer makes mistakes that deviate from the score: missing some notes out, misplaying others or adding extra notes. The accompaniment would need to cope with such mistakes that cause the performer to deviate from the written score.

In addition the performer should have the freedom to add musical embellishments that do not exist in the original score, as they would be able to with a human accompanist, without this disrupting the accompaniment.

It is very useful for a musician to have access to accompaniment during practice. The musician can learn how the accompaniment sounds and from this they can derive valuable assistance for future performance. As example, the musician would be aware of the underlying harmony provided by the accompaniment, and of any musical cues they could use when learning the timing of each section.

## **1.2 The use of Hidden Markov Models in score following**

Hidden Markov Models are a statistical modelling tool that can be used to represent real-world systems. In the last ten years, score following researchers have looked at using Hidden Markov Models to implement score following systems (Raphael, 1999; Cano et al., 1999; Orio and Dechelle, 2001; Orio et al., 2003).

## 1.3 Project Hypothesis

Using an Hidden Markov Model representation of the sequence of states in 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.

## 1.4 Project Aims and Objectives

This project investigates how an intelligent artificial music system can follow a human musician through the performance of a piece, and accompany the musician's performance as a human accompanist would, musically and in real-time.

The project considers a number of different approaches to score following, from the perspective of constructing a score follower, incorporating what has been learnt from previous research in the area.

Due to time restraints, the aim of this project was not to compare HMMs to other implementation approaches, but rather to test how an HMM can be utilised for score following, and evaluate how useful HMMs are for this application.

The primary objectives at the start of this project were:

1. Construct a Hidden Markov Model of the piece to be performed
2. Analyse input from the performer in conjunction with this Hidden Markov Model
3. Provide appropriate accompaniment to the performer that reacts to their interpretation of the piece and keeps in time with the performer.
4. Test system on a range of music to see how it adapts to different pieces.

## 1.5 What was achieved during this project

During this project, a number of score followers were developed in Max/MSP, using a Hidden Markov Model as the basis by which the soloist's progress was tracked through the score. As a byproduct of this project work, a Hidden Markov Model structure was implemented in Max/MSP such that the Viterbi algorithm could be carried out on a sequence of observations.

A number of enhancements were tried out, to enhance the basic score follower, such that beat tracking, complex accompaniment and relative score positioning were all incorporated to some extent in the score follower.

Three musical pieces of varying complexity and length were programmed into the different versions of the score followers. The performance of each score follower was evaluated subjectively by testers of varying musical ability and experience. Each score follower was also tested by objective criteria that was used to evaluate score followers at the Music Information Retrieval Evaluation eXchange conference in 2006(MIREX, 2006b); hence some general comparisons could be made between this project's score following systems and systems produced by existing score following research groups.

## 1.6 Terms used in this document

As well as the terms **score** and **score following** defined above, there are a number of terms used frequently throughout this document which shall be defined here, in order to avoid confusion:

- **Performance:** In the context of this project, a **performance** is defined specifically as the situation where a solo musician (**soloist**), such as a flute player or singer, performs a piece of music. The solo musician would be accompanied by another musician (**accompanist**) on an instrument such as piano. This may be in a concert or similar scenario, performing to an audience, but this condition is not mandatory. What is important is that the **soloist** is making an attempt to play through the piece in a linear fashion, from start to finish.
- **Performer/Soloist:** The solo musician who is performing the piece; what they play is the most important part of the performance for any audience that may be listening.
- **Accompanist:** The musician who is playing the **accompaniment**; supporting the **soloist's performance**.
- **Melody/Solo melody:** The music that is being played by the **soloist**.
- **Accompaniment:** The music which is played by an **accompanist**, during the **performance** of the **soloist**. **Accompaniment** can be thought of background music which is designed to enhance what the **soloist** is playing and support the **soloist's performance**.
- **Score follower:** A computer **accompanist** that follows the **solo melody** through the **score** as it is being played, to produce accompaniment relative to where the

**soloist** is in the **score**.

## 1.7 Outline of this document

This chapter introduces the reader to the project and gives an overview of its aims, the hypothesis that is being evaluated in this project and achievements.

Chapter 2 gives detail of significant prior research carried out in score following, describing how research in this field has developed and placing this project in the context of the current state of research.

Chapter 3 describes the fundamental decisions that were taken during the design of the score followers, to give the project reasonable scope and to decide which approach should be taken in various areas of the score follower (where more than one approach was possible).

Chapter 4 details the major stages of implementation of the score followers, including a summary of how I dealt with problems encountered and experimentation that was undertaken to decide how best to implement parts of the score followers.

Chapter 5 reports how the score followers performed in objective and subjective evaluation tests, and analyse the successes and limitations of the score followers produced during this research.

Chapter 6 discusses the performance and capabilities of this score following system as a whole, and presents the conclusions reached upon completion of this project.

An overview of the score following program produced in this project can be found in the appendices, along with more detailed implementation and testing information. The program is also available online<sup>6</sup> for download.

---

<sup>6</sup>Available at <http://homepages.inf.ed.ac.uk/s0676484>

# Chapter 2

## Background

Research into score following and computer/human musical interactivity was pioneered in 1984 (Vercoe, 1984; Dannenberg, 1984).

Originally dynamic programming algorithms were used to match patterns of notes expected against the sequence of notes actually played by the soloist.

Probabilistic methods were introduced to score following in the 1990s. Grubb and Dannenberg (1998) used probability density functions to locate the soloist's most probable position in the score.

A research group at the IRCAM institute<sup>1</sup> has developed much research into score following, using approaches derived from biological sequence alignment and from speech processing. Orio and Schwarz (2001) used dynamic time warping to analyse audio signals and align them to a musical score by detecting peaks in the music performed and fitting them to prominent events in the score.

Later research at IRCAM (Orio et al., 2003) used a Hidden Markov Model (HMM) to model the score and possible deviations where a wrong note is played, a note is missed out or an extra note is added. The Hidden Markov Model that they use has transitions modelling the probabilities for each event.

Raphael has carried out very similar work (Raphael, 1999). He too chose HMMs as the preferred method by which to model and track the score of the piece, although he has since started to incorporate Bayesian probabilities into his score followers as well (Raphael, 2001b).

What follows is a discussion of the important developments in score following research over the past three decades and a comment on how this project fits into the context of the current state of research.

---

<sup>1</sup><http://www.ircam.fr/>



## 2.1 Dynamic programming and pattern matching

A dynamic programming approach to score following was the first to be implemented, in the 1980s. The first automatic accompanists were created through research led by Barry Vercoe (Vercoe, 1984; Vercoe and Puckette, 1985) and also by Roger Dannenberg (Dannenberg, 1984; Bloch and Dannenberg, 1985).

The general methodology of Vercoe and Dannenberg's work is to compare what has been played by the soloist with what the score follower expected the soloist to play (i.e. the music that was written). In this way a recursive algorithm builds up the optimal path that the soloist would have taken through the score.

Vercoe (1984) emphasises the three-stage process behind score following:

1. Listen
2. Perform
3. Learn

The aim of score following, as Vercoe describes it, is

to recognize the computer's potential not as a simple amplifier of low-level switching or acoustic information (keyboards and live audio distortion), but as an intelligent and musically informed collaborator in live performance as human enquiry. (Vercoe, 1984) (p. 199)

In Vercoe and Puckette (1985) the work now includes the ability of the synthetic accompanist to learn the accompaniment and to learn the performer's likely paths through the piece (rather than merely reacting to the performer during a performance). This new aspect meant that this research team had to give serious consideration to the form the score model should take.

The score follower in Bloch and Dannenberg (1985) is constructed of two parts. The first is a *Matcher* segment, whose function is to match what the soloist is playing to a location in the score. It also has an *Accompanist* segment which takes information from the *Matcher* and produces the corresponding accompaniment.

This score follower deals with situations where it loses track of the soloist's location in the score in the following way: if a long period of time elapses where the *Matcher* has not reported a matching event to the *Accompanist*, the *Accompanist* assumes the *Matcher* is unable to locate the soloist in the score. Therefore it stops playing any accompaniment until a matching event is reported again by the *Matcher*.

This seems like a sensible approach to take; if the system is lost then any accompaniment it produces could well conflict with what the soloist is playing. In such situations it seems that it would be better to remain silent until the score follower has located the soloist in the score. This relates to the situation in real life where the accompanist does not know where in the score the soloist is, so stops playing until they have worked out the soloist's location again.

## **2.2 Important developments in score following research during the 1980s and early 1990s**

### **2.2.1 Real-time scheduling**

Dannenberg's score following work developed during the 1980s to include a strong usage of timeline-based scheduling (Dannenberg, 1989). A distinction is made between *real time* and *virtual time*. Actual physical time periods are referred to as *real time*, whereas the internal measure of time used by Dannenberg's score followers is referred to as *virtual time*. Dannenberg separates these two concepts as a way of measuring the soloist's relative progress through a score.

### **2.2.2 Probability density functions**

In 1989 Dannenberg and his student, Lorin Grubb, wrote about a new approach to score following using probability density functions (Grubb and Dannenberg, 1998).

From a given previous position in the score, the score follower continually estimates the distance from that previous position. The score follower then uses the most recent observations of pitch, spectral peaks, amplitude changes and so on, to locate where in the score the soloist has now reached.

Although this method was not adopted by other significant researchers, it is worthy of recognition as the first use of probabilistic methods in score following.

## **2.3 Dynamic time warping**

IRCAM presented two score following research projects at the 2001 International Computer Music Conference: one using dynamic time warping (Orio and Schwarz, 2001) and one using another technique for implementing score following, Hidden



Figure 2.1: Two sequences that DTW treats as very similar



Figure 2.2: Two sequences that DTW treats as very different

Markov Model based techniques (Orio and Dechelle, 2001). There is more detail about Hidden Markov Model based score following later in this document.

Dynamic Time Warping (DTW) finds the best alignment between two sequences by analysing similarities between the structure of the audio signals received and those expected, and by aligning note onset times. So Figure 2.1 shows two sequences which are treated as very similar, whereas Figure 2.2 shows two sequences which are considered to be very different from each other, even though this could just be due to a mistake where the performer has started playing the note at the wrong time:

Durbin et al. (1998) contrast the DTW approach with the Hidden Markov Model approach to sequence alignment. Orio and Schwarz's interpretation of this analysis is that the two techniques seem to be completely interchangeable except that DTW is more optimal on memory requirements for large files (Orio and Schwarz, 2001) (Section 2.3). Later work by IRCAM on score following, however, concentrates completely on the Hidden Markov Model approach (Orio et al., 2003; Schwarz et al., 2004; Cont et al., 2004), and there is little mention in later literature of DTW approaches being used further.

## 2.4 Hidden Markov Models

Automated musical accompaniment, or score following, can also be implemented using Hidden Markov Models (HMMs) to track what state a performer is in. This approach to score following has had some success to date.

### 2.4.1 General theory of Hidden Markov Models

Hidden Markov Models (HMM) are a stochastic modelling tool, popular in a variety of domains from speech processing (Rabiner, 1989) to biological sequence matching (Durbin et al., 1998). Real-world systems that produce some kind of observable signal can be modelled with HMMs. In particular this includes systems that operate non-deterministically: systems whose behaviour cannot be predicted exactly by using a set of algorithmic rules or formulae.

Probabilities are used in the HMM to represent the system's observable behaviour and to represent internal (hidden) facets of the system. The HMM can then be used to process these observable signals to explain the system's behaviour and make probability-based estimates about future behaviour.

As Rabiner describes (Rabiner, 1989), (p. 257),

The underlying assumption of the statistical model is that the signal [produced by the system at any given time] can be well characterized as a parametric random process, and that the parameters of the stochastic process can be estimated in a precise, well-defined manner.

So we assume there is some underlying pattern which can be picked out to model the observable outputs of a process (which may include some stochastic or random influence).

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 gain information about what state the system is currently in by examining recent outputs from the system ('observations'). The actual states themselves can not be observed, just the sequence of observations that result from the system passing through those states. The observed output can be interpreted as being "a probabilistic function of [the system being in] the state" (Rabiner, 1989) (p. 258).

The relationship between individual states and observations is not a functional relationship but a many-to-many relationship; one observation may be produced by many system states, and in turn there may be more than one possible observation should the system be in a given state. The fundamental difference between Hidden Markov Models and Markov chains (Durbin et al., 1998) is that you cannot gauge what state the sequence is in purely from the current observation in isolation. There is not a one-to-one correlation between states and observations in HMMs although there is in Markov chains.

Rabiner (1989) cites the example of a sequence of coin tosses, using either a biased or normal coin. A sequence of coin tosses generates a list of observations such as H T T H H H H T T H . With prior knowledge of the probabilities associated with starting with one of the two coins, the probability of changing from one coin to another and the probability of getting either a ‘head’ or a ‘tail’ with each coin, we can model this scenario using an HMM. We can use this HMM to make a probabilistic estimate of which coin is being tossed at any one time.

The components of a Hidden Markov Model are (Rabiner, 1989):

- $N$  = the number of states in the model
- $M$  = the number of distinct observations possible per state
- $A$  = the state transition probability distribution  $\{A_{ij}\}$  such that:

$$A_{jk} = P(\text{transition from state } s_j \text{ to } s_k)$$

$$\sum_{k=1}^N A_{jk} = 1$$

- $B$  = the probability distribution in each state  $j$ , that governs the probability of seeing each observation  $m$  when in that state, such that:

$$b_j(k) = P(v_k \text{ at } t \mid q_t = S_j)$$

$j$  is one of  $N$  states

$k$  is one of  $M$  observations possible from that state

- $\pi = \{\pi_i\}$  where  $\pi_i$  = probability of starting in state  $i$
- $V$  = the set of all observation symbols

### 2.4.2 Score following using Hidden Markov Models

A musical score is divided up into a sequence of musical events (for example where one note is considered as one modellable musical event)

The score follower is given a Hidden Markov Model that represents these musical events, and uses an algorithm such as the Viterbi algorithm to estimate what state the performer is most likely to be in at that time, i.e. which musical event in the score the performer is currently playing.

The aim is to find the most probable state sequence that could generate the sequence of observations produced by hearing the soloist's playing.

Later chapters of this thesis describe how HMMs were used to implement a number of score followers. In the score followers developed during this project, the Viterbi algorithm is used to find out which state the soloist is most likely to be in (given the sequence of observations of what notes the soloist has most recently played).

The Viterbi algorithm can be used to find the most probable path through a set of HMM states. Given a sequence of the most recent observations of the soloist's playing, the Viterbi algorithm:

finds the state sequence  $[q_1, \dots, q_t]$  that most likely generated the complete sequence of observations  $O_T$  (Rabiner, 1989)

Implemented in the traditional fashion, this algorithm finds the globally optimum path through the Hidden Markov Model states to the most probable current state, using the history of observations seen. However in score modelling we instead require a locally optimal path to the current point. This is because we are interested in accuracy locally at the expense of a more global accuracy (i.e. the correct accompaniment playing at the right time, even if the resulting path through the music overall is not the most probable path when the performance is viewed as a whole).

The Design chapter of this document describes how the Viterbi algorithm was adapted to be locally optimal rather than globally optimal.

In Rabiner (1989) and Pardo and Birmingham (2005), the equations used for finding the current state in a score following model are described in some depth. The equations that are implemented during the course of this project are a simplified version of these equations and are as follows:

$$\text{Given a sequence of observations } \{O = o_1, o_2, \dots, o_n\} \quad (2.1)$$

$$\text{and given an initial probability distribution } \sigma(s_j) \quad (2.2)$$

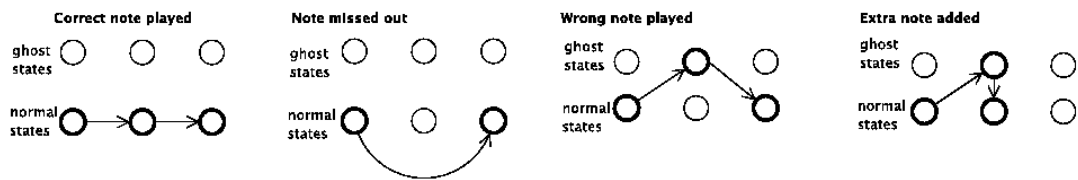
$$\text{The current location } l_i = \arg \max (\alpha(s_j, o_i)) \quad (2.3)$$

$$\alpha(s_j, o_i) = \phi(s_j, o_i) \sum_{s_k \in S} (\tau(s_k, s_j) \alpha(s_k, o_{i-1})) \quad (2.4)$$

$$\alpha(s_j, o_1) = \phi(s_j, o_1) \sigma(s_j) \quad (2.5)$$

$$\phi(s_j, e_j) = \varepsilon(s_j, e_j) \quad (2.6)$$

$$(2.7)$$

Figure 2.3: *Errors for states*

### 2.4.3 Implementation of score followers using Hidden Markov Models

The performance of HMM-based score followers has been evaluated favourably against string-matching/dynamic programming-based score followers (Pardo and Birmingham, 2005). HMM-based score followers significantly outperformed the dynamic programming score followers on the accompaniment of jazz melodies <sup>2</sup>.

The methodology described in Orio and Dechelle (2001) is that events in the performance (for example rests, notes, trills, chords, and so on) get modelled by Hidden Markov Model states. These events are modelled in parallel by both an *n-state* or *normal* state, and a *g-state* or *ghost* state.

Orio and Dechelle (2001) identify three classes of probable errors:

- **WRONG:** An incorrect note is played in place of the correct note.
- **SKIP:** A note in the score is missed out altogether.
- **EXTRA:** An extra, unscored note is added in the performance.

The Hidden Markov Model processes such errors by the soloist, as they happen, by taking a specific path through the *normal* and *ghost* states. The paths for each class of error are shown in Figure 2.3

Christopher Raphael has also constructed score followers using HMMs (Raphael, 1999). Raphael's choice of musical events represented by a HMM state are time-driven. Each observation is made up of a number of periodic samples taken from the incoming sound signal. These observations are used to analysed and locate the current state of the soloist.

<sup>2</sup>Generally the jazz melodies were of short to medium length, with chords rather than specified melodies as accompaniment, so the repertoire in this testing was slightly limited in scope

## 2.5 The current state of research in score following

The research team at IRCAM, Christopher Raphael and Roger Dannenberg are currently the prominent researchers in score following.

IRCAM are the publishers of the only commercial score follower that I can find evidence of. Their *suivi* score follower is available commercially<sup>3</sup>. However many other research efforts exist. Of these, the work by Christopher Raphael in particular has an impressive range of repertoire composed for it, with a number of demonstrations available online<sup>4</sup>.

Orio et al. (2003) discuss the unpredictability of performance. Currently when the IRCAM score follower is used in performance, the score follower needs close supervision by someone who knows the piece and the performer's common errors, in case the score follower makes mistakes during the piece.

Current problem areas in score following that are identified as prevalent in a 2003 review of score following research (Orio et al., 2003) include:

- Sources of mismatch between the performer and the computer
- Working with polyphonic input/ensemble groups of instruments
- Enabling cues (other than those derived directly from the musical performance)
- Reliability of the system in performance

## 2.6 How this work fits into the research context

My project would adopt the Hidden Markov model (HMM) approach, which has been successful for previous score following systems such as (Raphael, 1999; Orio and Dechelle, 2001; Orio et al., 2003). It would be interesting to take advantage of the new generative and analytical musical software that has become available, such as Max/MSP.

---

<sup>3</sup>At <http://forumnet.ircam.fr/357.html?&L=1>

<sup>4</sup>At [http://xavier.informatics.indiana.edu/~craphael/music\\_plus\\_one/index.html](http://xavier.informatics.indiana.edu/~craphael/music_plus_one/index.html)



# Chapter 3

## Design

In this chapter the design decisions taken during this project are discussed in depth. The requirements for the project are discussed and initial constraints on the project are specified, so that the scope of the project is clear to the reader. Once the exact requirements of the project has been clarified, the fundamental design decisions on conceptual issues and on practical issues are described, including comments on the alternative decisions that were considered and justification of the reasons why each design choice was made.

### 3.1 Requirements

My score followers require:

- A way of receiving musical input from the soloist
- A way of processing musical input from the soloist
- A way of implementing an HMM to analyse musical input from the soloist
- A way of generating musical accompaniment in real-time

### 3.2 Practical constraints

The scope of this project was initially bound by the following practical constraints:

#### 3.2.1 Time available for this project

The length of time allocated to this project was twelve weeks in total.

### 3.2.2 The type of musical input to be used

Orio and Dechelle (2001) discuss two approaches to score following:

- **note** approach: It is assumed that the performer's error is the cause of error, as the musical input method (e.g. a MIDI file) is assumed to be accurate.
- **signal** approach: Interpretation of the incoming musical signal is considered to be the source of errors, and any performance errors are disregarded.

At a very early stage in the project, it was decided to focus this project on the note approach. This was because including signal processing in my work would increase the amount of implementation time a good deal and would take the focus away from the simulation of an artificially intelligent accompanist, moving the focus more towards solving technical music processing problems.

MIDI files lend themselves well to being manipulated computationally to extract information about note pitches and durations. So MIDI is a particularly useful file format for musical information processing because the format of the music is described in such detail numerically in these messages, and can be extracted for calculation efforts more simply than it could be for an audio signal encoded in other common musical file formats such as Wave files and MP3. However there is often some sacrifice to be made in the quality of sound produced, and a common criticism of MIDI music is that the resulting audio sounds artificial and unrealistic.

Nevertheless, using MIDI rather than signal processing will significantly simplify note extraction from musical input. My focus will be on the interpretation of what is played, in an artificially intelligent manner, rather than the acoustics problems addressed in analysing exactly what has been played in the first place. Therefore using a MIDI representation of notes is preferable to using a signal representation.

### 3.2.3 Hardware resources available for this project

The setup needed in the development and testing phases of this project includes a workstation that is capable of sending and receiving MIDI messages, and a way of communicating between this workstation and a MIDI-enabled keyboard.

Two separate workstations were set up, so that the project was not reliant on just one workstation being available. This meant that when one workstation was unavailable or not functioning properly, there was an alternative workstation to use.

The Music department at the University of Edinburgh provided a Macintosh computer attached to a Yamaha Clavinova which functioned as a MIDI keyboard. An alternative hardware setup available during the project was a Windows PC attached to a MIDI-enabled Casio keyboard.

### 3.3 The conceptual design of my score following system

My score following system utilises the *normal* state/*ghost* state Hidden Markov Model structure described in Orio et al. (2003), with two states per musical event: a *normal* state (should the musical event be performed as expected) and a *ghost* state (should the soloist deviate from the score at this point).

Each state  $j$  is associated with an accompaniment  $x$ , such that if the soloist is in that state  $j$ , the accompaniment  $x$  will be played. This accompaniment produced is either the notes that should sound at that specified score location, should the soloist be in a *normal* state, or silence, should the soloist have reached a *ghost* state (as inspired by Bloch and Dannenberg (1985)).

The general algorithm that my score follower will use is as follows:

1. RECEIVE INPUT FROM PERFORMER
2. ESTIMATE WHAT HMM STATE THE PERFORMER IS IN, USING THE VITERBI ALGORITHM
3. PRODUCE THE APPROPRIATE ACCOMPANIMENT
4. DISPLAY STATE TRACKER ON SCREEN FOR INFORMATION

In more detail:

#### 3.3.1 RECEIVE INPUT FROM PERFORMER

For this the system needs to store: the note pitch (C, C#, etc) and the volume that the note is being played at by the soloist (so that the accompaniment is played at a volume relative to that of the performer, to match their musical interpretation).

### 3.3.2 ESTIMATE WHAT HMM STATE THE PERFORMER IS IN, USING THE VITERBI ALGORITHM

Given that we have a Hidden Markov Model  $\lambda$ : this part of the score follower uses the note pitch information obtained during RECEIVE INPUT FROM PERFORMER, along with previous observations, to carry out the Viterbi algorithm to find the most probable current state in  $\lambda$ .

At this point the system needs to store the number of the state which has been calculated as being the most probable current state (to generate the appropriate accompaniment).

### 3.3.3 PRODUCE THE APPROPRIATE ACCOMPANIMENT

This section uses a *coll* (Max/MSP's array structure) which has been pre-programmed with the details of what accompaniment to output, given the system is in a particular state  $j$ . Here the system uses the current state number, as calculated in the above VITERBI function, to look up the appropriate accompaniment for that state. It then produces MIDI messages that will play this accompaniment (or in the case of the current state being a *ghost* state, MIDI messages are produced such that the accompaniment falls silent).

### 3.3.4 DISPLAY STATE TRACKER ON SCREEN FOR INFORMATION

Feedback is a fundamental part of any interactive program. Given the amount of time available for development, and the extra load on processing speed that graphics display places in Max/MSP<sup>1</sup>, the feedback that the system gives to the user has been kept simple for the score followers in this project.

The score followers display the current HMM state. They also display information on the current score location in terms of the current bar and beat of that bar that the soloist is adjudged to have arrived at.

---

<sup>1</sup>If processing power is given to image processing, this would be at the expense of computational speed or music production, both of which are of higher priority in the score follower than graphical display

## 3.4 Major conceptual design decisions

### 3.4.1 Choosing the best features for state representation

Many musical features could be potentially be represented by an state, for example:

- A single note
- A single beat
- A fraction of a beat
- A phase of the *ADSR* note envelope (one of *attack*, *decay*, *sustain*, *release*)
- A rest
- A sequence of notes
- A cadence
- A phrase
- A MIDI note-on/note-off event

It was necessary to consider how best the score could be modelled: what events are most useful for the accompanist to know about and recognise. A list of significant events may include key changes or tempo changes, modulations and cadences, or specific sequences of notes. This would be variable depending on what piece is being played by the score follower.

In initial experimentation, it was enough purely to model a simple tune for which each note played by the soloist represented a new event. Hence each MIDI note-on message would be suitable as a new observation. For more complex melodies considered later, though, it was necessary to change this so that the musical events being modelled were beats or fractions of beats. This is discussed further in the Implementation chapter.

### 3.4.2 What to use in the set of possible observations

HMM observations or emissions are, in this domain, the notes played by the soloist.

If the score follower had been built to consider the absolute pitch of the incoming notes from the soloist, then this would mean there are 128 possible observations in each

state (as there are 128 MIDI notes). Even if this is restricted to a smaller keyboard, say one with five octaves, the score follower would still have 60 possible observations per state. Computationally this would be expensive.

Instead my score follower ignores octave differences between notes and merely consider 12 possible observations, the 12 notes in the western musical chromatic scale:

{C, C#, D, Eb, E, F, F#, G, G#, A, Bb, B}

So, for example, Middle C, MIDI note no. 60, would be treated as the same note as the C an octave above, MIDI note no. 72. This will mean that the score follower will perform considerably less calculations per state, with only a small trade-off in accuracy of observations for most pieces<sup>2</sup>.

### 3.4.3 Restricting the soloist's playing to be monophonic

If receiving input from a soloist via a keyboard, there are two possibilities for the incoming information.

The first possibility is that the input can be *polyphonic*. This means that the soloist would play more than one note at the same time, for example if they were playing a chord, they would press down more than one note on the keyboard at the same time. This would mean that every individual note in the chord would have to be processed against the score. It would give more information as to the soloist's current location, but would add complexity in the form of error analysis. As there are more notes to play, there are potentially more errors. It would be necessary to add an extra functionality to the score follower that compared how well the actual chord that was played could match to chords in the score (such as that described in Bloch and Dannenberg (1985)).

The alternative is to restrict the soloist so that they are only allowed to play *monophonic* melodies. In other words, they can only press one note on the keyboard at any one time. This would simplify the score follower to a particular domain of performance, i.e. simulating instruments that are only capable of performing monophonically, such as a flute or trumpet<sup>3</sup> Therefore this is a reasonable simplification to make

<sup>2</sup>The only type of piece where this compromise will cause problems are those where patterns of notes are repeated at different octaves, however it is anticipated that in such circumstances the score follower would have enough information historically to help in locating the soloist's correct current location through these passages

<sup>3</sup>The sound waves produced when such an instrument is played is actually constructed of subtones of many notes (known as *harmonics*) combined together to sound as one note overall. However the effect produced is that of hearing one note and a human accompanist would interpret what they are hearing as one note. The extra complications introduced by these acoustic concerns are far beyond the scope of this project.

for this project.

In actual fact, even if a soloist plays a chord on a MIDI keyboard, this is generally interpreted as a string of notes played monophonically, with a very small time-gap between each one (i.e. a few milliseconds). Sound processing systems require a level of extra processing to recognise such input as a chord. However this extra processing is quite simple to implement. So if in future work, my score following system was adapted to accept polyphonic input as well as monophonic input, then this would not require major changes to the way my system processes input from the soloist. However the part of my system that locates the soloist in the score would need to be updated to deal with chord matching such as described above.

#### **3.4.4 Comparison of anticipative and reactive design strategies**

Anticipative design strategies involve making some kind of predictions of what is about to be played by the soloist. Reactive design strategies, on the other hand, purely respond to the soloist as they play each musical event.

In considering both design strategies, a number of questions were considered:

- Should the accompanying note(s) start to play a fraction before the next expected note, or exactly when the next note is expected, or not until the soloist actually plays the next note?
- Should the score follower play the accompaniment associated with the next expected location in the score until it has worked out the actual next location the soloist has reached in the score? Or should the score follower play nothing until it has worked out where in the score the soloist has reached? Other alternatives are to play only the expected accompaniment then adjust accompaniment in time for the next input from the soloist.

If a human accompanist hears their soloist deviate slightly from the score, it takes time for the accompanist to relocate the soloist and adjust their playing from the expected accompaniment to the accompaniment matching the soloist.

It would be reasonable to have the computer accompanist only respond to a deviation on the next state after a deviation from the score was identified: replicating the slight delay that a human accompanist would also have. This is on the assumption that the states are modelled such that they are close enough together in timing for the delay not to be too noticeable.

In this project, though, it was decided that the score follower should make the change in accompaniment as soon as it has noticed there has been a deviation from the scored music, rather than delaying any changes in accompaniment. This way the score follower is being as accurate as its capabilities allow. Also, the score follower is not continuing to play an accompaniment that it has decided is incorrect, but is amending the accompaniment as soon as it has realised the change is necessary. In the same way, a human accompanist would change their accompaniment as soon as they detected the need to do so.

An interesting point is made in Dannenberg (1989) about the anticipation that an accompanist must perform. Latency issues may arise if the accompanist does not make the accompaniment sound at the same time that the soloist's sound is produced. In real-life a human accompanist prepares to play the expected next accompaniment before it happens, for example positioning their fingers in readiness to play the expected next part of the accompaniment. Dannenberg describes how there is potentially some latency in his score follower system (while the system is processing the information received by the soloist and locating them in the score). Because of this latency, there is a need to build a corresponding amount of anticipation into the playback of the accompaniment. The score follower predicts what the next accompaniment will be, early enough to have produced this accompaniment at the time when they expect the soloist to play their next note.

Given the advances in computer performance between 1989 and the present, this issue has reduced somewhat in importance. Computers can now process information considerably quicker than in 1989. But this does not mean that this issue has been eradicated altogether. In fact the improved computational resources can now be taken advantage of, so that a greater volume of information is considered when attempting to locate the soloist in the score (such as a longer history of what the soloist has played). The side effect of this is that latency issues may result. So it was decided that some element of prediction of the next accompaniment should be incorporated into my score follower, such that the score follower produces some accompaniment at the point when the soloist plays their next note (although the score follower may then adapt their accompaniment according to what the soloist has played).



### 3.4.5 Adapting the accompaniment when the score follower has identified a large change in the soloist's current score location

Dannenberg (1989) discusses what strategies to follow to keep the accompaniment “musical” (p. 257). As an example of this, he mentions one heuristic that the score follower should follow: if the accompaniment detects that it is behind the performer in a moderate amount or less, then the most musically pleasing way to re-synchronise with the performer is, according to Dannenberg, by playing the accompaniment line more quickly until it has matched with the performer again.

This is as opposed to ceasing to play the current accompaniment and passing over the accompaniment that lies in between that point and the point that the performer has reached, then starting to play the accompaniment again<sup>4</sup>.

Whilst this approach would mean that a smoother accompaniment would result overall, it is not necessarily the approach that would be taken by a human accompanist in a real-life situation. In my experience, if a significant gap (e.g. more than a few beats) developed in between the performer and the accompanist, the accompanist would be more likely to skip the intervening bars of the accompaniment and relocate to where the performer was in the score, rather than take Dannenberg's approach. This would mean that the accompaniment would match the performer, rather than potentially causing musical discordance<sup>5</sup>. Also, from the perspective of designing a score follower, it would be simpler to implement a jump in accompaniment location rather than to implement a speeding up and slowing down in the accompanist's play, especially if this is to be done in a musically acceptable way. So for these reasons I believe that there is little benefit to be had in following the more complicated approach that Dannenberg advocates (Dannenberg, 1989), therefore the simpler approach will be taken here.

---

<sup>4</sup>Dannenberg concedes that if the performer and the soloist are a large distance away from each other in the score, then the accompanist could skip parts of the accompaniment that are identified as less necessary for a musical accompaniment, although he does not go into any detail about how this judgement of musical importance could be performed

<sup>5</sup>If the gap between the two players' score positions was not so great, though, then Dannenberg's approach seems worthwhile

### 3.4.6 Encoding musical knowledge in the HMM probabilities

It would be helpful to have some means of gauging the local distance from the previously located position in the score to the estimated new location, in order to reposition the score follower. The larger the distance from the expected current position to the actual current position, the less likely in general it is that the performer has moved to that score location. Exceptions to this can be found in cases where the performer has missed out a bar or a sequence of notes, or where the performer has misinterpreted score markings such as repeat signs or Codas (that move the performer around in the score in ways other than purely sequentially, bar by bar).

Hidden Markov Models offer a convenient way of encoding this, by setting the probabilities associated with state transitions such that transitions between states near to each other should in general be more probable than transitions between more distant states. This does not include the exceptional situations such as mentioned in the paragraph above, where the probability for such state transitions may be increased slightly.

One aspect that deserves consideration is what the probabilistic weights in the Hidden Markov Model actually mean in terms of the operation of the score follower. The HMM probabilities influence actions taken by the score follower, by increasing or decreasing the probability associated with different paths through the HMM states. However this should not be confused with a deterministic prescription of one single possible path through the HMM states to the soloist's current location; in actual fact, more than one path through the HMM may be equally likely, given what the soloist has played. The score follower is estimating the soloist's most likely current location in the score, rather than knowing this location with full certainty. So the HMM probabilities are most effective if set to help to guide the score follower to consider more musically likely paths (for example, moving through the score smoothly and from left to right), placing less emphasis on musically less likely paths (such as ones which jump about in the score with little linearity).

### 3.4.7 The action to take when more than one state is judged equally probable by the HMM

It may be the case that for a given sequence of observations considered, the HMM finds that more than one state is equally likely to be the current state. Therefore the Viterbi algorithm will not return one state as its result, but instead a set of possible states, which are equally probable given what the soloist has just played.

In this situation, the score follower will know which state it has judged the soloist to be in, prior to the most recent input from what the soloist has played. It can use this information to choose the next state from the set of possible next states.

It was decided that the state that should be chosen by the score follower in this situation is the state which satisfies both of the following two conditions:

1. It is positioned to the right of the current state in the score (hence the state which occurs most immediately after the current state, as we read musical scores from left to right).
2. It is the closest state to the current state, of those positioned in the score in a location occurring later than the current state.

Should there be no states positioned to the right of the current state, then the score follower should choose the state which is closest to the current state, even if it is positioned in the score before the current state.

### **3.4.8 Learning the most accurate HMM probabilities through training**

To incorporate some element of learning through experience would be an interesting element to add to my score follower.

If we were to choose to train the HMM, this would involve getting the maximum probability of being in the correct *normal* state or *ghost* state, given a sequence of observations.

Training algorithms for HMMs exist and have been well documented (Rabiner, 1989; Durbin et al., 1998). Orio and Dechelle (2001) and Cont et al. (2004) describe how to train an HMM in the context of score following.

There are a number of issues when considering the training of HMMs:

- Who would train my score follower if it is being used as a practical application?

The training needs to be completed before performance. Soloists would be unlikely to want to spend time training up a score follower to the correct level. This is unless the practice sessions were of benefit to the soloist as well, in the same way that practice with a new human accompanist is beneficial so that the two players can become more familiar with each other's playing style and with the new music. So training would add to the preparation workload for the system.

- Should my score follower be trained for a particular performer's style of playing? Or a particular piece?

Training the HMM can cause it to model a given score more accurately, and target it to perform optimally for a particular style of playing or a specified piece. However as with all training techniques there is the risk of overtraining, which can overfit the model to a given performer, a given set of training scores or to a subset of common mistakes.

Musicians with different musical backgrounds may have playing styles that are quite distinct from each other. One example which becomes apparent later in the project is that there is more than one interpretation of the staccato style of playing, where notes are played for a shorter length duration than marked. Also, different performances would be heard for a piece which has rubato or improvised sections, or in one which has much ornamentation.

- Would training make that much difference, if the HMM settings were allocated well initially?

IRCAM have previously reported that training an HMM score follower was found to be less useful than expected (Smaill, 2007), although this may be due to the skills developed by the researchers in fitting the HMM parameters to score following in the first place, rather than a genuine lack of need of training. Schwarz reports in 2004 that some probabilities are "quite simple in nature for our case [score following], such that the PDF parameters can be set by hand" (Schwarz et al., 2004)

Considering the issues raised in this discussion, the decision made for this project was to not implement training of the HMM unless there is time at the end of the project. The extra work involved in implementing training may not have enough benefit to warrant the time spent on it, in relation to other tasks that could be carried out in that time.

### **3.4.9 Representing notes of different lengths using HMM states**

Initially, the score follower utilised a simple tune for which each note in the tune was of the same length. Hence it was an obvious choice to model each note as an individual HMM state.

Once the pieces become more complex however, it is no longer realistic to model each note as a new state, and instead the more pertinent aspect to model as a state is each beat, or a fraction of each beat. For such cases, it was necessary to consider how the timing information within the score should be modelled (in addition to how the notes should be modelled).

The two obvious ways to model a note that is held for longer than one state (i.e. notes that extend over a beat or more) are:

1. Allow states with self-transitions, so the HMM stays in a given state while a note is being held and only moves out of that state when the note is released.
2. Have a finite number of states representing each note that is longer than one state, proportional to the length of the note (for example if each state represents one beat and a note is three beats long, represent it as three sequential states).

The more successful option here seems to be the second (Raphael, 2007). This approach gives the score follower more flexibility to vary the accompaniment and also gives some information as to the expected length of the note. Therefore this was the approach used for encoding notes of different lengths into my HMM.

### **3.4.10 Tempo extraction (beat tracking)**

The ideology behind this project is that the soloist musician should be able to play their melody with their own interpretation, and that the accompanist should be able to keep with the soloist while they are playing the melody.

A natural extension of this is that the soloist should be able to play the melody at the tempo of their choice, perhaps varying the tempo for musical reasons or by mistake.

Beat tracking is the process of working out what tempo or speed a piece of music is being played at, by analysing the music that is being heard. It is a burgeoning research field in its own right, with much current research effort spent on improving beat-tracking capabilities (Dixon, 2001; MIREX, 2006a).

The implementation of beat tracking in this project is very simple in comparison to the latest state-of-the-art beat-tracking methods, but worked fairly effectively nonetheless.

My score follower measures the time in between notes played by the soloist. If the soloist is currently judged to be in a *ghost* state (i.e. they have deviated from the score), then the last input is not considered as valid for use in updating the tempo. If

the soloist is currently judged to be in a *normal* state (i.e. they can be found on the score), then the score follower works out how long the previous note should have been and compares this with the actual length of the last note.

The current tempo is based on an average of the recent (valid) tempo observations. The largest and smallest tempo observations are ignored and a mean is taken of the remaining tempo observations, to generate an estimate of the current tempo.

At first my concern was that the score follower was being given too much information from my own musical knowledge, rather than letting it demonstrate its own musical capabilities. Information about note lengths is however easily extracted from MIDI files. It would be a relatively straightforward task to write a program that could extract this information, as well as HMM probabilities and note information directly from MIDI files, in the format required by the score follower. However this is more an algorithmic/computational problem than an AI problem, so attention was instead refocused on the tempo extraction problem.

The issue with using the soloist's input to calculate their current tempo is that the soloist does not necessarily play on every beat, nor can they be relied upon to always play correctly and in perfect time<sup>6</sup>.

So the following tactics were implemented:

- If the soloist has entered a *ghost* state, then this means they have made a mistake, therefore the note they are currently playing should not be considered as very reliable evidence for tracking their tempo
- If the soloist is currently in a score location where they do not play a new note, then by definition there will be no new information from the soloist as to their personal tempo
- The only input that can guide the score follower in gauging the soloist's tempo is that where the soloist is expected to play a new note, and does in fact play that note. In these cases, the score follower can record the duration of such notes<sup>7</sup>, and compare the duration to what it expected the duration of that note to be (i.e. the current tempo, multiplied by the number of beats that note was to occupy).

The score follower keeps track of discrepancies between the expected note duration and its actual duration, and adjusts its own metronome tempo if necessary.

---

<sup>6</sup>If all performers could be relied upon to only play what is written in the score, then there would be no motivation for projects such as this

<sup>7</sup>Calculated as the time period between this note and the next note, so that if a note is not held for its full length - a common mistake - then this does not have a major influence on the tempo extraction

One particularly important decision made early on in this phase of development was to represent the tempo as msec per beat/state, rather than the more traditional tempo measure of number of beats per minute. This made calculations much simpler and did not affect the clarity of the system too much, as details of the tempo measurements were mostly used internally by the score follower rather than given out as feedback to the soloist.

## 3.5 Major practical design decisions

### 3.5.1 How MIDI files should be used in the score follower

The MIDI musical file format encodes notes in a way which can be interpreted by the score follower as an accurate source musical input. MIDI files transmit musical information in the form of MIDI instructions or messages, which can be one or more bytes long, depending on the complexity of the information being processed.

MIDI messages encode information such as:

- New note events,
- The cessation of notes being played,
- The channel on which a note should be played,
- The tempo at which notes should be played (therefore the speed at which the MIDI file should be processed)
- The instrumentation for a given note,
- Timing information for sequencing,
- ... etc.

What MIDI files do not include are musical markings such as bar markings and repeat signs. In a similar way, human musicians do not hear such musical markings. In general MIDI files encode what human musicians hear, i.e. what notes occur at what times.

A MIDI keyboard generates MIDI code, but does not necessarily have any built-in sound generation. This means that there is an issue that should be addressed in how sound is produced in my score following system. Either a MIDI keyboard that also

produces sound could be used, or a computer or an external synthesiser could generate the actual sound output that is the accompaniment (and perhaps also the solo line, if necessary).

To improve efficiency of the score follower, there would need to be a MIDI processing hardware setup that allows direct connections between the computer, MIDI keyboard and sound generation units (indeed the sound generation units may not be separate from the computer and/or MIDI keyboard). It would also be very helpful to use a programming/control environment that can facilitate direct communication in MIDI messages.

### 3.5.2 Output format for accompaniment

The output produced by the accompanist part of the score follower will be a stream of MIDI data. This could be either:

- A sequence of musical MIDI events, each triggered by a match between the soloist's playing and the score
- A continuous playback of a MIDI file with temporal "signposts" to guide the accompanist on which part of the accompaniment to play, such that the score follower is sensitive to tempo changes and location changes throughout the playback

So the output associated with each state would either be in the form of MIDI messages or score location data, respectively.

My decision was to take the first option, as this was a more direct way of producing the accompaniment. This option requires more effort as it is necessary to code the MIDI accompaniment directly into the score follower, rather than using a MIDI file for the accompaniment. Hence the score follower would need a little more preparation for a new piece rather than just telling it the location of the MIDI file that holds the accompaniment. However in real-time accompaniment the overall priority for the accompanist must be to produce the accompaniment sound with as little processing delay as possible; hence if the score follower produces MIDI messages as a direct output this would produce sound more quickly than if there was an extra level of processing to do before sound could be generated. Timing the accompaniment to be accurate in real-time, and avoiding any unnecessary sources of latency, is a primary concern in this project.



### 3.5.3 The type of pieces used for my score follower

Specific score-following repertoires have begun to develop, that take advantage of the lack of physical restrictions faced by an artificial accompanist in comparison to a human accompanist (such as notes reachable by a human hand on a piano, or the speed at which a human hand can play a sequence of notes). However this type of piece is not necessarily what my score follower should be able to play. Instead this project focused on a more traditional performer/accompaniment repertoire. This decision is due to the emphasis placed in this project on producing a score follower that behaves as a human accompanist would do, rather than investigating how an artificial accompanist may possibly outperform a human accompanist.

### 3.5.4 Which development tools to use

Several software packages have recently been developed which can generate music and reduce incoming audio to a computationally analysable form. Examples of these are JMusic<sup>8</sup> JSyn<sup>9</sup>, the MIDI toolbox for Matlab<sup>10</sup> and Max and its associated programs, for example Max/MSP<sup>11</sup>, jMax<sup>12</sup>, PureData<sup>13</sup>.

- Max/MSP, jMax and PureData

Max is a graphical development environment designed for implementing interactive musical performances between human and computerised performers. It has specific functions incorporated in its core program to allow interactive communication through MIDI messages and sound input/output. MSP is an addition to Max, that focuses on audio signal processing.

Max/MSP is a commercial package, incorporating Max and MSP, that is widely used for interactive music and is available to me through the Music Department at Edinburgh University.

There is a free version of Max called jMax, implemented in Java. Another freely available program that is based on very similar principles to Max/MSP is Pure-Data (developed by Miller Puckette, who has also been part of the Max development team at IRCAM).

---

<sup>8</sup><http://jmusic.ci.qut.edu.au/>

<sup>9</sup><http://www.softsynth.com/jsyn/>

<sup>10</sup><http://www.jyu.fi/musica/miditoolbox/>

<sup>11</sup><http://www.cycling74.com/products/maxmsp/>

<sup>12</sup><http://freesoftware.ircam.fr/>

<sup>13</sup><http://crca.ucsd.edu/~msp/software.html>

Of the three variants, PureData is in general more focused on signal processing (in fact the signal processing functionality of Max/MSP is based on PureData). jMax is slightly more unwieldy to use than Max/MSP. As its status as a commercially available program befits, Max/MSP is heavily documented (Zicarelli et al., 2006) and there are many sources of information on its use, as well as much sharing of functions written in Max/MSP. So as access to Max/MSP was available for this project, Max/MSP would be the preferred option here from these three packages.

- **MIDI Toolbox and Matlab** The MIDI toolbox for Matlab is created and distributed by the University of Jyväskylä, Finland. It takes in MIDI input and produces MIDI output. There are a large number of analytical tools available in the toolbox, where the raw data inside the midi file can be analysed and manipulated.

It has functions for “analysing and visualising” (and playing) MIDI files.

The toolbox has:

- Simple manipulation and filtering functions
- Cognitively inspired analytic techniques which enable context dependent musical analysis, e.g. metre finding segmentation.
- A new extension: **MIR toolbox**. This toolbox extracts musical features from an audio file.

The MIDI toolbox processes MIDI files by converting them to an  $N \times 7$  note matrix (NMAT). In this way it is very powerful at performing computational processing on MIDI files.

- **JMusic, JSync and Java**

JMusic is an audio package written in Java which allows the user to input, manipulate and output various types of audio files.

A similar package is JSyn: a Java interface that allows development of real-time interactive musical applications. It has classes specifically designed for functions such as synchronising audio streams.

- Writing a program in a conventional programming language (other than Java)

The other option considered was the use of C, Prolog or a similar programming language. Although this would be a realistic option, the more specialised tools considered above have greater capabilities for interactive music processing and manipulation, and would simplify implementation considerably compared to a language such as C or Prolog

Max/MSP is designed for real-time, dynamic, interactive music processing. Its strength is in how it enables interaction between performer and computer.

Max/MSP is less well set-up for implementing the finer points of the theory of HMM, as coding is done in Max/MSP at a higher level than a programming language such as Java or C++. Hence processing precise mathematical equations and implementing rigid control structures will be more challenging in this environment.

The MIDI toolbox and Max handle MIDI files more simply than the Java packages. From preliminary investigations, JSyn works better than JMusic at manipulating different events of the score and at handling interactivity between user and computer. The primary advantage of both of the Java packages over the MIDI toolbox is that it is simpler to customise a user-friendly interface to a Java system.

Unfortunately the MIDI toolbox had to be discounted as an option after consultation with the MIDI toolbox authors (Toivainen, 2007). The MIDI toolbox cannot be used to process data in real-time, through any way that they or I know of. Hence it will not be suitable for implementing a system that by necessity must process real-time incoming information (i.e. the soloist's playing). Therefore it could not be considered further as an option.

Several people were consulted for advice in making this decision who had experience in real-time music processing. By far the most recommended option was Max/MSP. Although I was not originally familiar with how to use Max/MSP, there were extensive tutorials and documentation of its capabilities (Zicarelli et al., 2006).

As acknowledged earlier, it is more complex to use Max/MSP, compared to a conventional programming language, for program control structures and data structures<sup>14</sup>. However the Max/MSP documentation (Zicarelli et al., 2006) demonstrated a wide variety of complex tasks which had been implemented in Max/MSP as examples of Max/MSP's capabilities.

Upon initial consideration of the project's practical demands, one that was immediately challenging for me was setting up a processing environment that could handle

---

<sup>14</sup>I am writing from the perspective of having a background in programming and computational calculation

and process incoming and outgoing sound information. Max/MSP is designed with this as a primary focus, and offers a simple and effective way of handling this task.

Therefore Max/MSP was chosen as the best tool with which to encode my score follower.

### 3.5.5 Existing HMM packages

As HMMs were to be used to implement my score follower, it would have saved time and implementation effort if a readily-available HMM package could be used to encode the HMM used in the score follower. The following HMM packages were considered:

- *HTK*: Hidden Markov Model Tool-Kit (University of Cambridge) <sup>15</sup>

This tool-kit is widely used in HMM applications. It is written in C but it is possible to export C code into Max/MSP. However I encountered a great deal of difficulty in performing this exporting of the C code into Max/MSP. Exporting C code to Max/MSP (this is referred to as writing a Max *external*) is an advanced facility of Max/MSP which requires certain extra software and I did not have the advanced knowledge in Max/MSP or access to the extra software necessary to use the HTK as a Max external.

- *UMDHMM*: HMM implementation (Tapas Kanungo) <sup>16</sup>

As this is another HMM implementation written in C, it was originally considered as an option but again similar practical difficulties were experienced when attempting to use this package in Max/MSP.

- *Hidden Markov Model Toolbox* for Matlab (Kevin Murphy) <sup>17</sup>

Had the MIDI toolbox for Matlab been capable of processing input data in real-time, this HMM toolbox would have been very helpful. Unfortunately, as mentioned earlier, this was not the case, so the HMM toolbox for Matlab was not a viable option for this project.

- *dishmm*: Discrete Hidden Markov Model (Paul Kolesnik) <sup>18</sup>

This is a Max external, written in C and imported by the author into a Max/MSP usable form. Originally this looked a very promising tool to use and I gained

---

<sup>15</sup><http://htk.eng.cam.ac.uk/>

<sup>16</sup><http://www.kanungo.com/software/software.html>

<sup>17</sup><http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html>

<sup>18</sup><http://www.music.mcgill.ca/~pkoles/download.html>

much experience in using HMMs in Max/MSP by experimenting with this HMM package and its examples. However during development, fundamental problems were encountered while using this package in my score follower. My experiences with using this package are described in greater detail below.

- *hmmm 0.15*: Hidden Markov Model implementation for Max/MSP (Yon Jason Visell) <sup>19</sup>

This is an Max/MSP patcher (a Max/MSP program) that implements a basic Hidden Markov Model. As it does not include any algorithm to find the best path through the HMM for a given set of observations, its sole use to me was as an example of how it is possible to implement Hidden Markov Models in Max/MSP itself, rather than as a C program imported into Max/MSP. It provided some inspiration for the belief that such an implementation was possible, however this patcher was not used to any further extent when implementing my own Max/MSP Hidden Markov model.

After some consideration of the above options, the Kolesnik HMM implementation (*dishmm*) emerged as my only realistic option for a readily-available HMM package.

### 3.5.6 Synchronising the start of the soloist and the score follower

In performances involving human musicians, the soloist and their human accompanist both start to play at the same time together, by giving each other the other musical and gestural cues such as counting a bar into the start of the piece or following a conducting gesture. (This also serves the purpose of setting the initial tempo that is shared by the soloist and the accompanist.)

For an artificial accompanist, an option would have been to simulate such a counting-in process by input from the soloist, or alternatively the score follower could generate a count-in through the MIDI file to bring in the soloist<sup>20</sup>.

As the score follower is designed to be reactive to the soloist's input, however, this consideration was deemed unnecessary, as the system will react to the soloist's input. While this may mean a small delay in starting to accompany the soloist, this is a short enough delay that it causes no discernable problems in performance.

<sup>19</sup><http://www.ph.utexas.edu/~yon/soft.html>

<sup>20</sup>It is beyond the scope of this project to incorporate any gestural recognition that could facilitate the score follower being brought in by conducting gestures.

### 3.5.7 Responding to the volume at which the soloist is playing

When the computer accompaniment produces music, it will need to know what volume to play at. There are three possible options here:

- The accompaniment is played at a pre-set volume throughout, regardless of the volume the soloist is playing at
- Volume markings are included in the accompaniment that the score follower is programmed with, such that it is told what volume to play at what time.
- The score follower plays its accompaniment at a volume relative to that of the soloist

My preferred option was the last of the three: to match the volume of the accompaniment to the volume that the soloist is playing. It is very rare that the accompaniment music should be heard at a louder volume than the soloist's melody. This is because the nature of a solo melody is that it is the primary part of the music being heard, therefore should be the most prominent sound for any listeners.

The score follower being developed in this project should therefore not play more loudly than the soloist at any point. Hence it should interpret the volume of the soloist and play no louder than this level of volume.

---

At this point, all major design decisions have been made, both practical and conceptual. The next chapter describes the actual implementation of the score follower system.

# Chapter 4

## Implementation of the Score Followers

This chapter describes the major stages of actual development. It includes a discussion of any decisions needed during development. Also documented here are the main problems encountered during the development process and how these problems were dealt with.

My score follower is written in Max/MSP (a real-time music interaction programming environment). The score follower runs on a Macintosh<sup>1</sup> Quadra 650 16Mb/6Gb(ext) computer that is attached to a MIDI keyboard: a keyboard which sends and receives data to/from the computer in the musical format MIDI (numeric messages about the note pitch, volume etc). In my studio the MIDI input device was a Yamaha Clavinova CLP.

My score follower was programmed with three different pieces:

1. First seven notes of *Twinkle Twinkle Little Star* (Simple melody), first with a simple monophonic<sup>2</sup> bass line and then with a polyphonic<sup>3</sup> or chordal accompaniment \*
2. An extract from *All I Ask of You* by Andrew Lloyd-Webber, first with a simple chordal accompaniment and then with a more complex accompaniment for which the accompaniment may move from one note to another while the soloist remains in a single state \*
3. An extract from *Danse Macabre* by Camille Saint-Saens

---

<sup>1</sup>My program can run in Windows or Macintosh OS, however all testing was performed on a Macintosh computer

<sup>2</sup>A maximum of one note sounding at any one time

<sup>3</sup>Potentially more than one note can be sounding at the same time

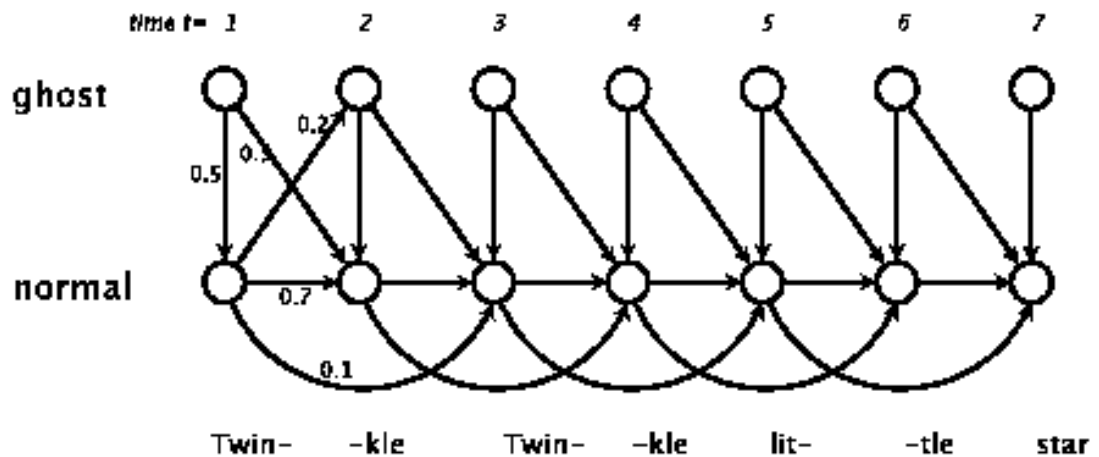


Figure 4.1: *HMM structure for Twinkle Twinkle Little Star, with one mistake allowed at a time*

All accompaniments for the pieces marked \* were composed by myself, so that the accompaniments fitted the requirements specified here. Scores for each arrangement of these pieces are included in the appendices section of this document.

## 4.1 Stages of development

The score followers in this project were developed in an incremental fashion, starting from basic versions and gradually adding more complex functionality.

The various stages of development during this project were:

- Score follower that can recognise a simple short melody. This version allows for recovery from one deviation from the score at a time, and produces a monophonic accompaniment (one note sounding at a time).

This score follower was the foundation of all later versions. It uses a Hidden Markov Model as depicted in Figure 4.1. Each note in the melody (*Twinkle Twinkle Little Star extract*) is represented by a *normal* state and corresponding *ghost* state. Some exemplar probabilities are included in Figure 4.1 (although the majority of probabilities are omitted from this diagram, for overall clarity).

- As for the previous score follower, with chordal accompaniment added (so more than one note is played at the same time by the accompanist).



This was implemented by altering the data structure used to store the accompaniment, and developing the part of the score follower that generated the accompaniment, so that both were able to cope with multi-note input.

- As for the previous score follower, with added functionality: the score follower plays the accompaniment that is expected to occur next, being played prior to the actual score location being estimated

This version implements some anticipation of what is to be played next by the soloist. To work out what accompaniment it expects to play next, the system takes the current state and moves onto the next state sequentially of that type. So if it has calculated that the soloist is located at *normal* state 3, the system expects the next state to be *normal* state 5. (*Normal* states are numbered with odd numbers, *ghost* states with even numbers.)

If the soloist is currently considered to be in a *ghost* state, for example *ghost* state 8, then the current accompaniment would be silence (as no accompaniment is produced when located at a *ghost* state). Hence the simplest action here would be to move to the next *ghost* state, for example *ghost* state 10. As the score follower does not yet know how or when the soloist will return to the score, in this scenario the preferable action is to keep the accompaniment silenced until the soloist can be identified as on the score again<sup>4</sup>.

- As for previous score follower, but allowing more than one deviation from the score to be made concurrently

To enable this functionality, the structure of the HMM was changed. Figure 4.2 shows part of the new HMM structure: it shows all the state transitions that are possible from the first *normal* state, with associated probabilities<sup>5</sup>.

- As for previous score follower, with simple beat-tracking implemented to estimate the soloist's current speed

Beat tracking was implemented, as described above in the Design chapter. At this point the score follower is still fairly basic; it reacts to each input from

---

<sup>4</sup>Here the same effect could have been achieved by staying in the same *ghost* state, or indeed any *ghost* state, but as this choice influences nothing except the first few msec of producing accompaniment, I chose the option that was simplest to implement

<sup>5</sup>By this point, the probabilities had been developed during experimentation so have been refined somewhat

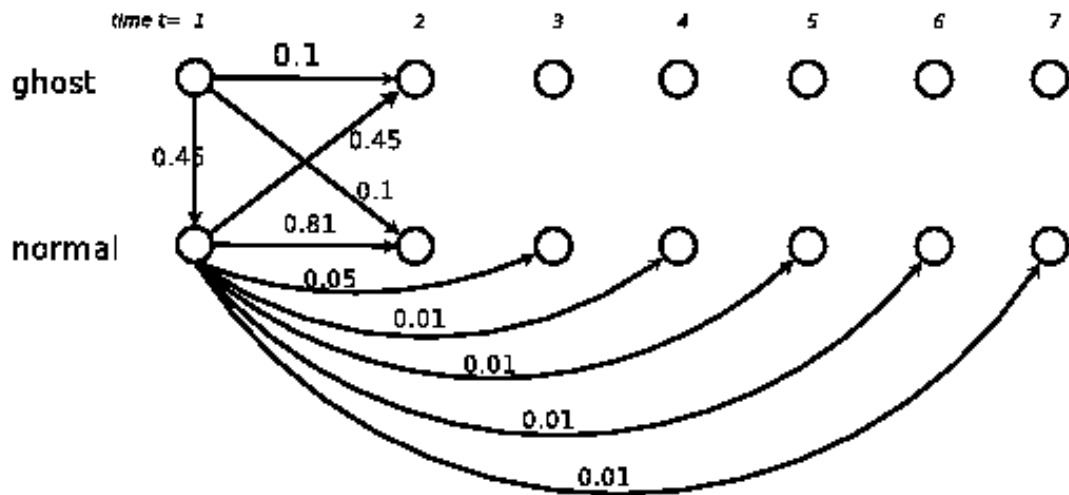


Figure 4.2: *HMM structure for Twinkle Twinkle Little Star, with handling of multiple concurrent mistakes*

the soloist, rather than using any notion of tempo to control the timing of the accompaniment that it plays, or to anticipate the soloist's next musical event.

- Smoothing of accompaniment, so that held notes in the accompaniment can be played smoothly as intended, rather than being played as individual new notes on each state transition.

The heuristic used for this is:

If new note is currently sounding, do not replace it. If new note is not currently sounding, make it sound. Any currently-sounding notes that are not included in the set of new notes to be played should be turned off

- Score follower that recognises a substantially longer and more complex melody  
The new melody (*Danse Macabre extract*) incorporates notes of different lengths, repetition of note sequences in different locations in the score, and staccato playing.

This was essentially the same as the most advanced of the previous score followers (but without beat-tracking implemented at this stage).

The only amendments necessary to the main structure of the previous score follower were the inclusion of the new HMM information and secondly, a slight

change in the interpretation of what was being played by the soloist. Previously the melody being interpreted by the score follower only had notes of equal length, and no rests or silences by the soloist, to be interpreted. The new melody has notes of different lengths. It also has staccato notes: the effect of staccato is that the note is played more shortly, such that there is extra silence added after the note is played, at the expense of playing the note for its full duration.

Therefore the HMM structure was amended to represent the score slightly differently. Instead of representing each **note** by a *normal* state and *ghost* state, the music score was now represented by modelling each **beat** (in actual fact each half-beat) by a *normal* state and *ghost* state. This is because some notes were now longer than others and so their duration spanned across more than one state. Hence differing note lengths were modelled using a finite collection of states such that the number of states is relative to the length of the note, as advocated in Christopher Raphael's work (Raphael, 1999, 2001b, 2007).

Also the score follower was adapted to recognise when the soloist was not playing a note at a given time, and an extra emission was added to represent that the soloist was silent at that point.

- As for previous score follower, with simple beat-tracking implemented to estimate and follow the soloist's current speed

Beat-tracking was implemented as before.

- Give the score follower a new melody to recognise

The new melody (*All I Ask of You extract*) was chosen because it was slightly simpler and shorter than for *Danse Macabre*, but notationally much longer and more complex than for the *Twinkle Twinkle Little Star* extracts.

The main objective in this stage of development was to see if the score follower could recognise a new melody, without needing to make fundamental changes to the score follower program itself. This was achieved; the only changes to the score follower system were to give it the new HMM information for the new melody.

- As for previous score follower but with more complex accompaniment that has notes moving in the accompaniment while the soloist is still static in one state

Here it was necessary to further develop the data structures and the part of the score follower that generated the accompaniment, so that they could include and use information about the length of time each note in the accompaniment should be held for (in terms of the number of states, or the fraction of a state, the note should be sounding for). As a result the score follower could be told what the more complex accompaniment was and could play this more complex accompaniment.

## 4.2 Major sections of the score follower system

**get\_soloists\_next\_input:** Extracts the new information from the soloist, that comes in from the 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# has been played, etc) In later development a thirteenth possible observation was added: 12 if nothing is being played, i.e. the soloist is silent.
2. Adds this observation to the list of observations seen so far and extracts the three most recent observations.
3. Performs the Viterbi 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,  $\langle [48, 52, 55] \rangle$  - 'In state 1, play MIDI notes 48, 52, 55' (these MIDI numbers correspond to a C Major chord)

N.B. I found during development that it was necessary to keep the functionality of each section completely separate, so that for example the generate\_accompaniment patcher was the only patcher that could affect the playing of the accompaniment. This encapsulation approach eradicated many small errors where one section would interfere with the correct operation of another section.

## 4.3 Problems encountered during implementation

### 4.3.1 Conceptual/theoretically-related problems

#### 4.3.1.1 My implementation of an HMM in Max/MSP

As discussed later in this section, no suitable HMM packages were found for use in Max/MSP. Indeed I was advised to program as much of the inner workings of the system as possible myself, for better control over system performance (Raphael, 2007).

Given the graphical and high-level nature of Max/MSP, a programming task such as this is more complicated than if implementing an HMM in a language such as C or Matlab<sup>6</sup>. Due to the time available for development, it was decided that only a partial implementation of the HMM architecture should be necessary. So the model structure, parameters and probabilities were all implemented, as was (a slightly amended version of) the Viterbi algorithm, but learning functionality was not incorporated into my HMM.

Once the Hidden Markov Model structures and Viterbi algorithm had been implemented successfully, some adjustments were made to the score follower's Viterbi algorithm, to improve its efficiency. Originally, when working out the probability of a particular state being the soloist's current state, there were many redundant calculations being performed. This was because in an equation of the form  $LHS * RHS$ , if LHS was equal to 0 then there would be no need to calculate the RHS, as by definition, anything multiplied by 0 is also 0. However my original implementation of the Viterbi algorithm worked out values for both LHS and RHS in this situation. So to reduce the computational effort, the algorithm was amended such that before working out a RHS, it would first check to see if the LHS of the product was 0. If this was the case then it would not need to work out RHS, but instead return  $LHS * RHS = 0$ .

#### 4.3.1.2 Using a locally optimised version of the Viterbi algorithm

The Viterbi algorithm (Rabiner, 1989; Pardo and Birmingham, 2005) is a recursive algorithm. The base case for this algorithm relies on the assumption that the first observation being considered relates to being in an **initial state**  $a_0$ .

---

<sup>6</sup>It is possible to write a C program and import it into Max/MSP (this is referred to as writing a Max external), however my knowledge of Max/MSP and C is not advanced enough for this to have been an option for me

$$\alpha(s_j, o_1) = \phi(s_j, o_1)\sigma(s_j) \quad (4.1)$$

$$\sigma(s_j) = \text{the probability of starting in state } s_j \quad (4.2)$$

(Pardo and Birmingham, 2005) (p. 3) If using only a subset of the observations rather than all the previous observations, as is the case here, then the Viterbi algorithm requires some amendment. The base case of this algorithm was altered slightly to calculate a probability for the first state of this sequence that did not require this first state to be an initial state. This was done by using the  $\phi$  function to replace the use of the initial probability table.

$$\alpha(s_j, o_1) = \phi(s_j, o_1)\phi(s_j, o_1) \quad (4.3)$$

Hence  $\phi(s_j, o_1)$  replaces  $\sigma(s_j)$ . So the Viterbi algorithm can now be used on a locally optimal basis, rather than needing to be based on the whole sequence of observations so far.

#### 4.3.1.3 Linear movement through the score

For more complex pieces (particularly *Danse Macabre*, which had a number of similar sequences repeated through the score), the states identified by the score follower did not always form a linear sequence, even when the soloist played in a linear fashion. Such linearity could be derived to some degree with experimentation on setting the HMM state transition probabilities.

The system was enhanced further after feedback from testers. Some extra weight was added to the probability of the next expected state that had been calculated by the Viterbi algorithm. This had promising results and helped the score follower to move through the score more smoothly.

This amendment meant that the HMM settings were being overridden to a certain degree; however with more accurate HMM probabilities (and possibly with training) I believe the same effect could be achieved. When assessing the effectiveness of the HMMs used in score followers in this project, though, it was necessary to bear in mind this amendment to the HMM probabilities.

#### 4.3.1.4 How to represent a rest or silence from the soloist

My original set of possible observations was the set of integers from 0 to 11, representing the twelve possible pitches in an octave ( $\{C, C\#, D, E\flat, E, F, F\#, G, G\#, A, B\flat, B\}$ ). As mentioned above, when modelling more complex pieces that had rests in the soloist's melody or staccato (short) notes, it was necessary to include a thirteenth possible observation that represented silence or the lack of note input from the soloist.

### 4.3.2 Technical problems

#### 4.3.2.1 Silencing notes that are currently sounding

A common problem with MIDI is that if a note has been made to sound using a MIDI note-on message, then this note will continue to sound until a corresponding MIDI note-off message is received for that note.

This caused a problem when needing to silence a previous accompaniment note or chord, in order to replace it with the next note or chord. This problem was solved by sending all accompaniment through a Max/MSP object called *flush*. Using this object, it is possible to turn off all currently sounding notes for which *flush* has not yet received a note-off message, by sending a 'bang' message to the *flush* object. This 'bang' message causes all such notes in the *flush* object to cease sounding, to make way for the new accompaniment to be heard.

#### 4.3.2.2 Using an external HMM package in my score follower

My original plan, as described in the previous chapter, was to use the HMM implementation for Max/MSP written by Paul Kolesnik. Initially this HMM package worked well, in the early stages of implementation of a score follower. However problems soon emerged.

The Viterbi implementation in Kolesnik's HMM package was designed to find the optimal path through the HMM only after all observations had been received. As it was necessary for the Viterbi algorithm to operate at a more local level during the performance of the piece rather than at just the end of the piece, I attempted to change the source code slightly to allow this to be possible. On re-compilation, however, I was unable to use my new code as I did not have all the necessary source code files made available to me<sup>7</sup>, so could not complete the compilation of the new code. Therefore I

---

<sup>7</sup>Despite some email correspondence with Paul Kolesnik about my use of his code

could not use the Viterbi algorithm as originally planned.

If this had been the only problem encountered, I could have worked around this, as the implementation of the HMM was otherwise well written and sound in operation. However I also found that the HMM implementation did not scale up a great enough extent. I was not able to use it for an HMM with 14 states and 12 observations. I was unable to work out why this was the case, but could find no reason that I could trace in the code other than perhaps it was too large for the HMM structures to handle (however the settings for the HMM structures were in the source code files which I did not have access to).

Having emailed several prominent researchers in score following using HMMs, as to how they implemented their HMM practically, I received one response from Christopher Raphael. His recommendation was to implement the HMM structures and functionality myself. As an HMM implementation suitable for my needs could not be found, I did in fact code a partial HMM implementation myself (as described earlier in this chapter). Although this meant a significant extra implementation effort and extra complication in the development stage of this project, this approach also meant I had a more thorough overall understanding of the inner workings of the HMM and could exert more control over the various aspects of the HMM implementation, to better suit the needs of the score followers.

#### **4.3.2.3 Terminating the accompaniment gracefully at the end of the piece**

The last chord of the accompaniment did not automatically stop playing at the end of the piece as there was no incoming accompaniment notes with which to replace the last accompaniment chord. To solve this problem, refinements were made to the part of the program that processed the soloist's input. As a result it was able to monitor whether or not the soloist had appeared to stop playing and also could monitor if an end state had been reached. The accompaniment was silenced if one or both of these conditions were satisfied.

## **4.4 Experimentation with my system during development**

Although many design decisions were made prior to implementation, a number of decisions were taken during the development of the project, about how various parts



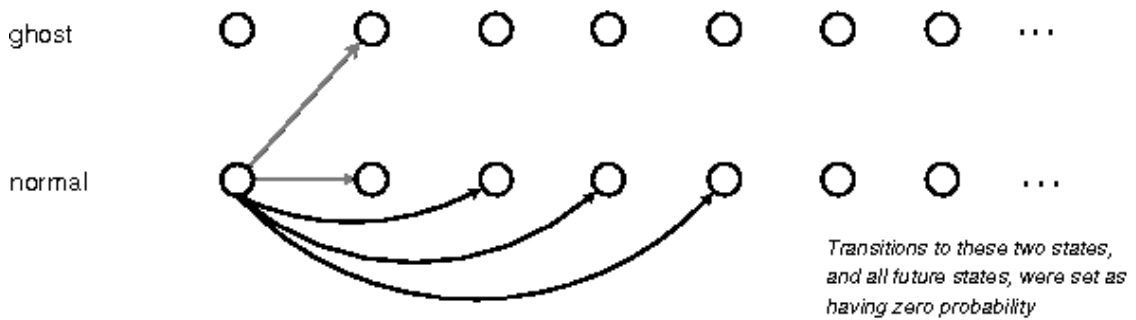


Figure 4.3: Transitions representing the SKIP error (where a scored note is missed out) are only necessary for a small number of future *normal* states

of the score follower would work best. These implementation decisions were based on the results of experimentation carried out during the development phase of this project.

What follows is a brief discussion of the main experimentation that was carried out during development, and the results that were obtained.

#### 4.4.1 Fitting an HMM to score following

My score follower needed to know probabilities associated with starting in each state (`initial_prob.txt`), moving from one state to another (`state_transitions.txt`) and the probability of seeing each possible observation from each state (`emission_probabilities.txt`).

Although training of these probabilities was not explicitly included in my implementation, much experimentation was performed with different values for these probabilities. This experimentation could be said to serve as a form of manual training of the HMM.

The main findings from this experimentation were that:

- A smoother performance was achieved when the transition between sequential *normal* states was allocated a much higher probability than the transition between *normal* states and *ghost* states. This was found to encourage the score follower to take a more linear path through the score of the music.
- Transitions representing that a note had been missed out or SKIPPED<sup>8</sup> were only necessary for a small number of future *normal* states. Figure 4.3 shows an example of the resulting HMM structure.

<sup>8</sup>This type of error is discussed in previous chapters and in Orio et al. (2003)

It would be interesting to investigate whether, with some implementation of training of the HMM, these settings could be improved further.

Experiments were also made with the structure of the HMM, as to how the states modelled the musical score. Each piece of the three pieces was modelled slightly differently.

- The *Twinkle Twinkle Little Star* extract was modelled using a state to represent each new note of the melody.
- The *All I Ask of You* extract was modelled using a state to represent each beat of the melody
- The *Danse Macabre* extract was modelled using a state to represent each half-beat of the melody

It was more effective to use beats rather than notes as the musical events to be represented by states, because more complex melodies could be implemented. (The HMM for *Twinkle Twinkle Little Star* relied upon each note of the melody being the same length.)

Using a full beat as a musical event was more musically intuitive for myself, when constructing the HMM. However the use of a half-beat as a musical event allowed the score follower to track the position of the soloist more precisely.

#### 4.4.2 Number of observations used for Viterbi

I experimented with using different sizes of histories of observations (the number of notes most recently played by the soloist) to locate the soloist's current position in the score. The score followers were run with a history of the last 2, the last 3 and then the last 4 observations being used by the Viterbi algorithm.

My findings were as expected: the more observations that the Viterbi algorithm used, the more accurately the score follower was able to estimate the soloist's position in the score. This was verified when measuring how accurate the score followers were during later testing (see the Objective Testing discussion in the later chapter on Testing).

Another significant finding was that the smaller the number of states, the larger the number of observations that could be presented to the Viterbi algorithm without

serious latency issues emerging<sup>9</sup>. For melodies that were modelled with a large number of states, the score followers quickly developed latency issues if using the Viterbi algorithm on a larger number of observations (although this was overcome to a certain degree by calculating the probabilities beforehand, as described below).

Later experiments with restricting the scope of states considered as potential next state helped to alleviate this latency somewhat. These experiments are discussed later in this chapter.

#### **4.4.3 Setting a probability thresholds for the estimation of the next state**

I experimented with setting a probability value as a threshold value. The probability of a possible new state being the current state had to exceed this threshold in order to be considered as valid for the actual current score location.

This did not have any adverse effect on the score follower's performance, but it was later found that the optimum value for this probability varied according to the piece, because of differing numbers of states being involved in the multiplicative calculations. Therefore it was simplest to disable the enforcement of this condition for later versions of the score followers that used more states. Disabling this did not lead to any noticeable reduction in performance.

#### **4.4.4 Finding the most effective beat tracking parameters**

The beat tracking, or estimation of tempo from what has been played by the soloist, was experimented with in order to find a set of parameters for the beat tracking which gave the most accurate measure of the soloist's speed.

Some of the explorations made in this area included:

- Only considering tempo observations once there have been three or more tempo observations

Therefore the first three notes are not used initially for tempo alignment. The soloist should have dictated the tempo to the system before starting to play (although there are default values set for the tempo of each piece in case this does not happen). It is worth remembering that if the soloist had not given their tempo

---

<sup>9</sup>By latency I refer to the speed with which the accompaniment was played in response to each note played by the soloist

to the accompanist in a real-life situation, then it would take a human accompanist a few notes to gauge the soloist's tempo as well.

- Taking the last eight tempo observations and finding the average of these observations

I experimented with the number of tempo observations that should be considered and found that eight was a good number to use. This meant that extreme values that distorted the average would eventually be “forgotten” by the score follower, but that the score follower still had an adequate history of tempo observations from which to estimate the soloist's current tempo<sup>10</sup>.

- Disregarding the most extreme large and small values in the list of tempo observations, so if there are any inaccurately timed notes then they do not have a disruptive influence over the tempo calculations

I experimented with removing the largest and smallest elements from the list of tempo observations (so it was ignoring two elements in total). This worked more accurately than if all elements from the list of tempo observations were considered.

In the next experiment the top and bottom two elements were removed from the list (so the score follower was ignoring four elements in total). This was too unresponsive to changes in tempo and did not adapt its estimated tempo quickly enough to match the soloist's tempo. So the final decision was to only disregard the largest and smallest tempo observations when estimating the soloist's current tempo.

- Changing the system slightly so that the internal metronome is restarted on a new beat, when new input comes in from the soloist.

If the soloist is slightly off-beat for one note, then this mistake will be corrected over time.

If the soloist is speeding up, then the current beat is started earlier than it should start, which is intuitively correct: if the soloist's tempo speeds up then their beats occur earlier than expected.

---

<sup>10</sup>This suggests the possibility of being able to use HMMs to implement beat tracking, such that the tempo observations could be used as a guide to the soloist's underlying tempo. However this is much out of the scope of this project and will not be considered further in this document

If the soloist is gradually slowing down, then the expected metronome click comes before the actual metronome click. Initially I had concerns that this would cause a problem where the score follower would start the next beat before the soloist does, so would start to work out the next state that the soloist has (in theory) changed to, *before* the soloist has actually changed state. However this proved to be less of a problem than anticipated. This was because any calculations of a new state were overridden when the soloist does change state.

#### 4.4.5 Performing the viterbi algorithm off-line, to generate pre-calculated probabilities for longer scores

Originally, latency issues had a large impact on the performance of the score follower for the *Danse Macabre* extract. Because of the large number of states in the HMM for this piece (201 states), the score follower could not keep track of the soloist to any recognisable degree, due to the amount of time that Viterbi algorithm computations were taking. This was not so much of a problem when only using the two most recent observations with which to track the soloist, but was significantly affecting performance when using a history of three observations or more.

For each new note played by the soloist, and using a history of three observations, the score follower is carrying out  $N^3$  calculations. For an observation sequence of  $M$  musical events, with a history of  $X$  observations, the score follower will have to carry out  $MxN^X$  calculations. This is a high computational load and the score follower could not cope with this computational burden and still produce real-time accompaniment.

To reduce the computational load, one version of the score follower used a table of probabilities that had been calculated off-line, prior to the soloist starting to play. During runtime, it would then use the most recent observations from the soloist as a key to look up what the most probable next state would be.

This approach involved a large calculation effort when constructing the table of probabilities ( $M^X * N^X$  calculations total where  $N$  = no of states,  $M$  = no of possible observations, and a history of  $X$  observations is used)<sup>11</sup>. However during the time the system is online and running, the time saving is considerable. Instead of carrying out

---

<sup>11</sup>There is likely to be a more efficient way of calculating this table, but I chose to use a simple and direct way by taking each combination of observations in turn. On my system setup this took 24 minutes to run for a 201 state/13 emissions HMM with a history of three observations. I estimated that using a history of four observations would take approximately five hours to produce a table of probabilities for this setup. Unfortunately, certain time restraints I had on access to resources meant that I was not able to produce and test this probabilities table

intensive calculations for each observation (as outlined above), it is only necessary to look up a single value in a table for each musical event played by the soloist. So implementing this meant that the score follower a piece with a larger HMM structure (in this case the *Danse Macabre* extract) could be tested on a live basis.

#### 4.4.6 Reducing latency issues by only examining local states as potential next states

In the research literature, many HMM-based score followers have been produced which can cope with longer pieces (Raphael, 2001b; Orio et al., 2003; Pardo and Birmingham, 2005), without resorting to offline calculation of probabilities.

To reduce the computational burden, the options were to either change the way the soloist's position in the score was estimated or to reduce the number of calculations being performed. The latter option was taken in this project, in order to reuse the basic structure of the previous score followers.

Instead of considering all states in the HMM as possibilities for the soloist's current location in the score, the score follower now only considered a *window* of states, centred around the state that had been estimated as the previous state. The default window size was large enough to consider approximately a bar and a half worth of states.

This experimentation was developed further, by expanding the number of states considered by the score follower as potential next states, if the system is not able to estimate the current location of the soloist successfully, and is "lost". The score follower was deemed to be "lost" if it has estimated the current state to be the same as one of the previous two states; in other words if it has become stuck in a particular set of states.

At some point the score follower would find the soloist's position in the score again, so would have chosen a suitable state such that it was no longer considered to be "lost". Upon reaching this point, the number of local states being considered would be reduced back to the default window size.

Now my score follower could perform the Viterbi algorithm online. The score follower was more robust than expected, in terms of how well it performed when the soloist's position in the score moved larger distances in the score than usual. However this robustness could be improved in the future, given further work.

---

At this stage in the project, a number of versions of a score follower have now been fully implemented. This chapter has discussed the major implementation stages that occurred during this stage of the project, describing the main decisions taken during implementation and how several problems were dealt with during this time.

# Chapter 5

## Testing and Evaluation of System Performance

The performances of the score followers produced during this project were evaluated both objectively and subjectively. The system was tested against measurable criteria originally constructed in 2006 by score following experts to test the latest research efforts (Cont and Schwarz, 2006).

As well as this testing, the score followers were tested and judged by musicians of varying musical ability and experience, so that they could give their opinions on the quality of accompaniment provided by the score followers.

Several versions of the score followers were tested. As the objective testing was carried out at a later time than for the subjective testing, the *Danse Macabre* score followers had been updated slightly. Two versions were tested during objective testing that carried out a localised rather than global search for the next state, in order to be able to perform the Viterbi algorithm online (while the soloist was playing) rather than having to calculate probabilities beforehand. All other versions were tested both subjectively and objectively.

The score followers were tested in a thorough and structured manner and the results from each testing stage were evaluated. The different score followers in this project were compared against each other, against other score followers in the research domain, and against a human accompanist.



## 5.1 Objective testing

### 5.1.1 Objective testing methodology

My score followers were tested using criteria that was used to test score following systems in the 2006 Music Information Retrieval Evaluation eXchange (MIREX) conference, and which will be used again in the 2007 MIREX conference.

This testing criteria is the result of much discussion between experts in score following (Raphael et al., 2006). On the MIREX conference website <sup>1</sup> there are results published from the evaluation of two separate score following systems in the 2006 conference (MIREX, 2006b). So this criteria has been devised with some careful consideration. It is also possible to perform some general comparison between my score follower and other score following systems (although this will be limited as my score follower will have been tested using different pieces and therefore different challenges to the score follower, and will also have been tested under different conditions).

The objective testing criteria from MIREX (Cont and Schwarz, 2006) is as follows:

- **Event Count:** The number of musical events included in the played melody (i.e. the number of musical events for which the score follower has to estimate a state)
- **Number of Notes Missed:** Scored notes that the score follower does not recognise at all, or which are recognised but with an offset of greater than 2000 milliseconds. This is tracked by seeing which states the score follower goes into during accompaniment, and when each state is entered
- **False Positive (FP):** Scored notes that the score follower only recognises after a delay of greater than 2000 milliseconds (this is also included in the above statistic **Number of Notes Missed**)
- **Average Offset:** The mean of the recorded Offset measurements between the note onset (note being played by the soloist) and the accompaniment being played
- **Standard Deviation Offset:** The standard deviation of the above Offset measurements

---

<sup>1</sup>[http://www.music-ir.org/mirexwiki/index.php/Main\\_Page](http://www.music-ir.org/mirexwiki/index.php/Main_Page)

- **Average Latency:** The mean of the recorded Latency measurements between the detection time of the note being played by the soloist and the time the system has processed the audio so that it is ready to be matched to the score <sup>2</sup>
- **Missed Note %:** The percentage of missed notes ( $\frac{\text{Number of Notes Missed}}{\text{Event Count}}$ ). The inclusion of missed notes in this criteria is to convey how accurate the score follower is at tracking the soloist's exact position in the score.
- **False Positive %:** The percentage of False Positive notes ( $\frac{\text{Number of False Positives}}{\text{Event Count}}$ )

Additionally there are two overall measures with which to compare my work overall with each score follower submitted at MIREX 2006:

- **Total precision:** The percentage of correctly detected notes overall (i.e. all score followers' results added together)
- **Piecewise precision:** The mean of the percentage of correctly detected score notes *by each score follower*

As well as this, I included a subjective measure of how well I judged my score follower to have performed during the test:

- $\frac{5}{5}$ : Flawless accompaniment, indistinguishable from or better than the accompaniment that an expert human accompanist would play
- $\frac{4}{5}$ : Very good accompaniment, with almost no errors
- $\frac{3}{5}$ : Good accompaniment, with some flaws but generally accurate and musical
- $\frac{2}{5}$ : Some accompaniment performed accurately but with many errors and unmusical moments
- $\frac{1}{5}$ : Poor accompaniment with very few moments where the accompaniment was played accurately
- $\frac{0}{5}$ : Where the accompaniment played bears no resemblance whatsoever to what should have been played

---

<sup>2</sup>The definition of this in (Cont and Schwarz, 2006) is slightly misleading: "Difference between detection time and the time the system sees the audio" but my interpretation of the latency measure is as described in the main text

All the measurements described above were used as objective measurements to evaluate my score followers.

The score followers tested were:

1. *Twinkle Twinkle Little Star*, with beat tracking enabled such that the soloist's tempo is extracted - but as this score follower is purely reactive, it only plays accompaniment when it receives a soloist's input. So the beat tracking is enabled but does not affect the playing of the accompaniment.
2. *All I Ask of You*, with a set metronome tempo for the tester to follow.
3. *All I Ask of You*, with beat tracking enabled so that the score follower should follow the tester's tempo.
4. *Danse Macabre*, with beat tracking enabled, using the probabilities calculated offline by the Viterbi algorithm
5. *Danse Macabre*, with beat tracking enabled, performing the Viterbi algorithm online and using a history of the last three observations from the soloist
6. *Danse Macabre*, with beat tracking enabled, performing the Viterbi algorithm online and using a history of the last four observations from the soloist

Five tests were carried out on each score follower. For each test, the score follower was presented with a specified melody from the soloist. During each test, the score follower's performance was judged against the objective criteria outlined above. The five tests were:

1. Play the melody as scored, with no mistakes, tempo changes or embellishments
2. Play the melody with selected errors added
3. Play the melody with selected embellishments added
4. Play the melody as scored but with selected tempo adjustments made
5. Play the melody, making all the deviations from the score from tests 2, 3 and 4

Details of the specific alterations made to each melody are included in Appendix D.

Where there is more than one version of the score follower for a particular piece: the five evaluation versions of the melody were kept the same for the testing of each

Score Following System Authors	Total Precision	Piecewise Precision
Arshia Cont and Diemo Schwarz (MIREX 2006)	82.90%	90.06%
Miller Puckette (MIREX 2006)	29.75%	69.74 %
Anna Jordanous (this project)	60.89%	54.04%

Table 5.1: Overall Summary Results

score follower for that piece, so that one version could easily be compared against the others.

It was necessary to make some adjustments to my score followers so that evaluation information could be collected. These adjustments made no difference to the actual workings of the program but had the sole function of depositing information at various stages of the program's runtime.

Where possible, the MIDI input was provided automatically via another Max/MSP patcher<sup>3</sup>, so that inconsistencies in a human's playing would not adversely affect the results too much. Instead the new Max/MSP patcher played the prescribed test melody at a specified tempo (which could be altered at any point during play).

Latency measures were taken using a patcher from the online Max/MSP documentation. This patcher, *cpuclock*, took measurements of the CPU clocktime elapsing between receiving MIDI input from the soloist and playing first the expected accompaniment and then the calculated accompaniment.

### 5.1.2 Results of objective evaluation tests

- **Results table:** See Figure 5.1
- **Overall Summary Results** as for MIREX (2006b): See Table 5.1.

<sup>3</sup>In Max/MSP, the term *patcher* is used to refer to individual programs and/or functions

	Piece	Event Count	No. Notes missed	False Pos	Ave offset	Std offset	Ave Latency	Missed Note %	False Pos %	Rating /5
1	Twinkle	7	0	0	20.95	3.99	0.07	0.00%	0.00%	5
2	Twinkle	7	0	0	12.81	8.04	0.08	0.00%	0.00%	5
3	Twinkle	7	1	0	52.08	70.91	0.08	14.29%	0.00%	3
4	Twinkle	7	0	0	95.03	149.54	0.11	0.00%	0.00%	5
5	Twinkle	7	0	0	131.14	180.75	0.07	0.00%	0.00%	3
1	All I ask v.1	40	1	0	532.15	1028.59	0.07	2.50%	0.00%	5
2	All I ask v.1	44	24	11	487	984.42	0.09	54.55%	25.00%	4
3	All I ask v.1	40	28	26	504.97	1033.88	0.06	70.00%	65.00%	3
4	All I ask v.1	n/a								
5	All I ask v.1	44	38	27	504.47	1034.02	0.47	86.36%	61.36%	3
1	All I ask v. 2	40	3	0	505.01	490.95	247.76	7.50%	0.00%	3
2	All I ask v. 2	44	11	2	433.99	354	239.58	25.00%	4.55%	2
3	All I ask v. 2	40	6	0	790.79	555.08	212.33	15.00%	0.00%	2
4	All I ask v. 2	40	13	1	473.63	615.65	158.3	32.50%	2.50%	3
5	All I ask v. 2	44	16	2	840.37	547.21	237.38	36.36%	4.55%	2
1	All I ask v.3	40	1	0	542.82	896.06	0.18	2.50%	0.00%	4
2	All I ask v.3	44	23	11	499.74	860.74	0.16	52.27%	25.00%	3
3	All I ask v.3	40	18	2	354.13	691.49	0.75	45.00%	5.00%	2
4	All I ask v.3	n/a								
5	All I ask v.3	44	33	13	371.43	681.79	0.86	75.00%	29.55%	2
1	All I ask v.4	40	3	0	452.11	458.27	221.5	7.50%	0.00%	2
2	All I ask v.4	44	15	4	374.66	312.95	224.47	34.09%	9.09%	2
3	All I ask v.4	40	10	0	417.98	365.59	303.54	25.00%	0.00%	2
4	All I ask v.4	40	15	2	379.09	443.77	184.66	37.50%	5.00%	3
5	All I ask v.4	44	20	4	358.38	392.61	215.13	45.45%	9.09%	2
1	Danse Macabre v.1	100	95 *	11	269.65	518.02	204.53	95.00%	11.00%	2
2	Danse Macabre v.1	113	108 *	15	283.17	524.4	301.38	95.58%	13.27%	2
3	Danse Macabre v.1	100	95 *	11	319.76	539.61	209.86	95.00%	11.00%	2
4	Danse Macabre v.1	100	95 *	13	271.83	547.69	182.42	95.00%	13.00%	3
5	Danse Macabre v.1	113	108 *	13	263.93	552.29	265.39	95.58%	11.50%	1
1	Danse Macabre v.2	100	53	4	242.62	53.04	86.97	53.00%	4.00%	2
2	Danse Macabre v.2	113	98	3	250.86	81.96	98.63	86.73%	2.65%	1
3	Danse Macabre v.2	100	40	6	267.67	95.34	102.2	40.00%	6.00%	2
4	Danse Macabre v.2	100	64	4	240.08	62.09	81.34	64.00%	4.00%	3
5	Danse Macabre v.2	113	77	3	254.45	44.72	153.47	68.14%	2.65%	1
1	Danse Macabre v.3	100	4	0	912.41	283.54	62.4	4.00%	0.00%	3
2	Danse Macabre v.3	113	39	1	961.57	369.91	58.79	34.51%	0.88%	2
3	Danse Macabre v.3	100	87	5	962.88	366.25	76.67	87.00%	5.00%	0
4	Danse Macabre v.3	100	79	17	969.97	312.31	95.08	79.00%	17.00%	2
5	Danse Macabre v.3	113	105	0	982.47	386.88	101.94	92.92%	0.00%	0

\* The missed note counts for this score follower are estimated, due to the high performance inaccuracy

Figure 5.1: Results of objective evaluation tests

### 5.1.3 Analysis of objective evaluation test results

It is pleasing to see in table 5.1 that despite the differing development times concerned, the score followers developed in this project compared favourably overall in performance to the two score followers analysed at MIREX 2006 (submissions by Cont and Schwarz<sup>4</sup> and by Puckette<sup>5</sup>). The weaker result on the piecewise precision is affected by the poor performances overall from the *Danse Macabre* score followers.

These comparisons, however, can only be made at a very general level, if at all. It is difficult to justify comparing this project's score followers with the MIREX score followers, as the score followers were tested on different pieces.

The MIREX 2006 test repertoire included a Boulez flute piece, a Bach violin sonata, Mozart clarinet concerto and Mozart vocal piece. These pieces were of considerably longer duration than my test repertoire, with on average 2239 events per piece, as opposed to my average of 49 events per piece. It is interesting, though, to note that there is a similar degree of variance in the success of the score followers tested in MIREX 2006 (MIREX, 2006b) as there is in the results shown for this project (in Figure 5.1). This indicates that there is a degree of variance in the accuracy of the MIREX 2006 score followers, depending on what piece is being played. This was also true for the different pieces that my score follower was accompanying.

In interpreting the results, the **Average latency** column shows how long the system took to receive the input from the soloist and process it ready for using in the HMM (for example by storing it in internal variables). The **Average offset** column shows a measure of the time it takes to estimate the most likely state and produce the corresponding accompaniment.

Recording the number of **False Positive** notes in addition to the **Number of Notes Missed overall for that melody** was useful; a high percentage of False Positive notes relative to Missed Notes overall indicates situations where the score follower has not located the exact position of the soloist in the score, but where it is following the soloist through the score at roughly the right position. Although this is unlikely to produce correct sounding accompaniment, it is better to know that the system is tracking the soloist to some degree, rather than having completely lost track of where the soloist is in the score.

---

<sup>4</sup>An HMM-based note/signal score follower, described at [http://www.music-ir.org/evaluation/MIREX/2006\\_abstracts/SF\\_cont.pdf](http://www.music-ir.org/evaluation/MIREX/2006_abstracts/SF_cont.pdf)

<sup>5</sup>A dynamic programming-based note score follower based on Dannenberg (1984), described at [http://www.music-ir.org/evaluation/MIREX/2006\\_abstracts/SF\\_puckette.pdf](http://www.music-ir.org/evaluation/MIREX/2006_abstracts/SF_puckette.pdf)

As expected, when the score followers incorporated some form of beat-tracking, the latency associated with receiving and processing the soloist's playing was higher than for the simpler score followers. This is because there was an extra layer of processing involved for each input from the soloist (checking to see if the tempo needed to be updated).

In general, the score followers that were rated highly on the accompaniment they produced had low scores in the **Missed Note %** and **False Positive %** columns. This is an obvious conclusion to make: if the score follower finds the correct location of the soloist, then it will know exactly what accompaniment to play.

A less obvious addition to this conclusion can also be made. If the score follower performs badly in the percentage of notes missed, but the percentage of notes identified after some delay (False Positive notes) is higher than average, then the score follower was usually rated as performing well. For example in Test 2 of the first version of *All I Ask of You*, the performance of the score follower was rated at  $\frac{4}{5}$ . This is despite over 50% of the notes played by the soloist not being matched to the soloist's location in the score. Unlike, for example, Test 1 on the second version of *Danse Macabre*, which was similarly poor at finding the soloist's location, but rated at only  $\frac{2}{5}$ , 25% of the notes missed were identified after some delay (were False Positive notes). This meant that the score follower for Test 2 of *All I Ask of You*, version 1, had some idea of where the soloist was located in the score, so as a consequence was able to produce an accompaniment that flowed more smoothly and musically, compared to the other performance.

As expected, the simpler melodies performed much better in general than the *Danse Macabre* score followers. In particular the score followers for *Twinkle Twinkle Little Star* and *All I Ask of You* performed the accompaniment better than anticipated during Test 5. This was the test where all the errors from the previous tests were combined into one playing. For this test there were many bars of the derived melody which were almost unrecognisable from the original tune. On many occasions, in evaluating the results, there was some ambiguity in exactly which state the soloist was in at a number of points, with a number of equally plausible options to explain the series of deviations that had been made from the score. A human accompanist would have had to apply some skill and concentration when accompanying a soloist who was making this number of deviations from the score. So the attempts made to accompany the soloist in Tests 5 (particularly for version 1 of *All I Ask of You*) were a positive result of testing.

Some of the results for the *Danse Macabre* score followers are particularly poor. Poorer performances came from the score follower that considered all states as possible candidates for the next state, rather than a local selection. This score follower very rarely found the correct next state and was also poor at estimating the next state to be one that was

Generally, if there is a smaller number of potential next states from which to choose from, then the score follower will be likely to make a better attempt at locating the soloist. As there are fewer options to consider, numerically there is a higher probability of estimating the next state correctly or nearly correctly, even if no HMM probabilities were to be considered.

The second and third versions of *Danse Macabre* score follower rely on the soloist playing through the score in a roughly linear fashion, with no large jumps. There is some increasing of the number of states considered, should the score follower not be able to locate the soloist in the score on first parse. However it is much easier for these versions of the score follower to lose track of the soloist, should the soloist make large jumps in the score. Tests 2 and 5 included a jump of two bars backwards by the soloist, and one bar forward. Both these tests were handled relatively well by the version two score follower. The third version of the score follower coped well with these score relocations in Test 2, however the delays in note processing that was involved in Test 5 caused much interference in the score follower's performance, such that no useful conclusions could be drawn from this test.

The *Danse Macabre* score follower that used a history of four observations for the Viterbi algorithm (version 3) gave a very accurate performance in the first test (where the solo melody was performed correctly). It was also reasonably accurate in the second test (where selected errors were included during performance of the solo melody). This shows the greater level of accuracy that can be achieved if more information from the soloist is considered.

A criticism of this particular score follower, though, is that latency measurements associated with the more detailed calculations were considerably higher and this is reflected in the poorer ratings overall that the third version received for quality of accompaniment. In general, the higher the Average Offset or Average Latency recorded, the less musically and accurately the score follower was judged to have performed.

There were very large figures for the Average Offset for version 3 of the *Danse Macabre* score follower. This was reflected in the performance, where the accompanist lagged behind the soloist, especially in Tests 3 and 5. However the overall accuracy



measurements for some of the tests on this version were considerably higher than expected, given how the accompaniment was performed. The score follower was finding the next states, but not quickly enough to perform the accompaniment well. Without the objective testing, this would probably not have been noticed. As the primary objective of a score follower must be to produce musically accurate and well-performed accompaniment, though, this delay in the system should be a main consideration in any further work <sup>6</sup>.

## 5.2 Subjective testing

### 5.2.1 Subjective testing methodology

In addition to testing the score followers against objective measurable criteria, the score followers that were developed in this project were evaluated by human musicians of different levels of musical competence and experience.

The overall aim of a competent score follower should be to provide musical and accurate accompaniment. The quality of an accompanist's performance in general is judged by how well it fits and enhances the playing of the soloist whom they are accompanying. In fact the very nature of a good accompanist is that the listener is not aware of their playing except as an enhancement to the soloist, whose playing should be attracting all the listener's attention.

Testers were presented with 5 versions of the score follower to test, in order of increasing complexity of the score follower and the piece:

1. *Twinkle Twinkle Little Star, with beat tracking enabled such that the soloist's tempo is extracted - but as this score follower is purely reactive, it only plays accompaniment when it receives a soloist's input. So the beat tracking is enabled but does not affect the playing of the accompaniment.*
2. *All I Ask of You, with a set metronome tempo for the tester to follow.*
3. *All I Ask of You, with beat tracking enabled so that the score follower should follow the testers's tempo.*
4. *Danse Macabre, with a set metronome tempo*

---

<sup>6</sup>Perhaps by using different algorithms to calculate the most probable path, for example the Forward-Backward algorithm (Rabiner, 1989)

### 5. *Danse Macabre, with beat tracking enabled*

For the more complicated pieces, I gave the testers a couple of minutes to practice the solo melodies before adding the accompaniment. This meant that they could pay more attention to the performance of the accompaniment rather than concentrating purely on playing the right note, but that they were still inclined to make the occasional mistake themselves, especially for the *Danse Macabre* extract.

For each score follower, the testers were asked first to play the melody as correctly as they could, then to play it with different variations of mistakes, embellishments and tempo changes. I deliberately did not specify any errors or embellishments that the testers should make, so as to avoid influencing them in their choice of what to play. This meant that some of the testers tried errors or embellishments that I had not considered trying, which was useful to me in evaluating the score follower.

The testers gave their opinion on how well they perceive the system does at accompanying them, and how well it recovers from errors and embellishments that the testers add.

## 5.2.2 Observations arising from Tester 1

Tester number one is a bass guitar and double bass player of medium to high musical ability. He is experienced in accompanying other musicians. In particular he is used to playing jazz music, where the accompanist must always be aware of any improvisation or free playing that the soloist might want to include.

### 1. *Twinkle Twinkle Little Star*

The score follower performed well in accompanying the tester. He commented that it kept better track of his exact position in the melody than he would have expected.

When the tester tried polyphonic input (playing the melody using both hands, in different octaves at the same time), he felt that it still accompanied him relatively well, although there was a noticeable consistent error in the parts of the melody that include repeated notes. This is understandable as although we hear polyphonic input as a number of notes occurring exactly at the same time, programs such as the score follower would treat the polyphonic input as a sequence of notes received one after another in very quick succession. Unless the score follower was specifically designed to treat such sequences of notes as chords

(as discussed in Bloch and Dannenberg (1985) and in the Design chapter of this document), then it would interpret each note in the chord as a new event, rather than treating each chord as one event.

A point made by this tester was that if he played the melody and then repeated back to the beginning, the score follower did not actually recognise that he had repeated back to the beginning, for a couple of notes. He said that if the soloist was to go back to the start of the piece and repeat it again, then the score follower should be able to recognise the repetition. I had not thought of this possibility when setting the HMM probabilities as I had originally designed the score follower to move through the melody once only per run. This possibility would however be enabled quite easily, by increasing the probability of moving from the final *normal* state (in this case state 13) back to the first *normal* state (state 1).

## 2. *All I Ask of You, with a set metronome tempo*

The tester noticed that the score follower was more likely to lose track of his score position at certain points, but that it finds his score position again fairly well.

He commented that in general, the score follower performed better as an accompanist when he made errors, rather than when he played the music perfectly. This would probably be due to the HMM probabilities and therefore correctable, given a little more time spent in setting the probabilities more accurately, especially at the points that the tester highlighted the score follower had some difficulties with.

## 3. *All I Ask of You, with beat tracking enabled*

The score follower was judged to have made a passable attempt at accompanying the tester in this test.

An interesting effect that I observed while the tester was playing this was that the tester adjusted their playing to fit with the accompaniment, and attempted to give the score follower musical cues to assist the accompanist in producing a musical performance. For example he emphasised the first beat of the bar by playing it with a more pronounced manner. Unfortunately I had not implemented any way in which the score follower could use such cues, but such musical knowledge could be added to the system in the future.

4. *Danse Macabre, with a set metronome tempo, and*

5. *Danse Macabre, with beat tracking enabled*

Unfortunately neither version of the *Danse Macabre* score follower performed well in accompanying this tester.

They made a considerably less musical attempt at accompanying this tester than they had done during my playing, in development. The main difference between the tester's interpretation of the solo line and my own was that the tester played the many staccato notes in the piece for a shorter duration than I had tended to do during development, so there was a larger proportion of silence in the piece compared to when I played it. Also, he played the tune with a very careful observation of a correct tempo, even when playing the occasional wrong note.

6. *Further comments*

### 5.2.3 Observations arising from Tester 2

Tester number two is a classical woodwind player and pianist of medium musical ability. Unlike tester number one, she identifies more with being the soloist who is accompanied rather than the accompanist.

1. *Twinkle Twinkle Little Star*

The tester was impressed with the way in which the system accompanied her for this extract. She commented that "it was almost as if the system was being perceptive", in working out where she was in the score and producing the right accompaniment.

2. *All I Ask of You, with a set metronome tempo*

Again the tester was impressed with the accompaniment that the score follower produced, particularly noting the slide of notes in the last bar of the more complex accompaniment, and how the last note of the accompaniment was being played even when she tried to confuse the system by playing the last couple of bars incorrectly.

The tester also mentioned that this score follower responded well to embellishments she made to the tune, accompanying her as she would have expected a human accompanist to do.

3. *All I Ask of You, with beat tracking enabled*

This score follower in general kept with the solo performance line well, the tester commented. However there were obvious points where she felt the accompaniment was not catching up to her change in tempo quickly enough. She was happy with an experiment she did where she made a rapid change in tempo for a few notes then returned to her original tempo. As she would have expected a human accompanist to do, the score follower ignored the few notes played at the different tempo and instead continued to play at her original tempo, but made some attempt at synchronising to the few notes played more rapidly during that short period of time.

4. *Danse Macabre, with a set metronome tempo, and*

5. *Danse Macabre, with beat tracking enabled*

Unlike the previous tester, the score follower was able to keep playing some accompaniment for the tester, although it did not perform as well as for the other tunes and the tester was less impressed by this version.

This tester's interpretation of the staccato notes was more similar to my interpretation, in that she intentionally played staccato notes for half their normal duration rather than making each staccato note equally short, and as short as possible.

Her performance of the correct version of *Danse Macabre* solo line was different to the first tester's, in that when she made a mistake, she generally paused, and played that section again, correcting her mistake, then continuing, rather than carrying on and ignoring the mistake. This occasionally confused the score follower but in general it coped fairly well with this approach.

6. *Further comments*

### 5.2.4 Observations arising from Tester 3

Tester number three is a classical vocalist and pianist of high musical ability, who specialises in Baroque and Classical period music. He is also a Music Technology student with a particular interest in Acoustics.

This tester is equally knowledgeable about being the accompanist and being the

soloist in a real-life performance environment. He is also particularly influenced towards adding embellishments to notes, in a Baroque style<sup>7</sup>.

### 1. *Twinkle Twinkle Little Star*

One point made by this tester was that when he added a trill<sup>8</sup>, the accompanist should play one held chord underneath the note for the duration of the trill. Instead, it tried to follow the trill notes as individual notes of the melody, quickly repeated, so the accompaniment was disjointed and uneven. This is because this version of the score follower uses no information on the length of notes, but instead treats each new incoming note as a new state. The tester found that this was less of a problem in the later versions, although the trills still caused difficulties for the score followers which had beat tracking enabled.

The tester tried playing the melody backwards and the score follower responded by roughly tracking where he was but by playing the accompaniment in a forward direction where there was a choice. He was happy with this response as he felt this is what a human accompanist would do until it had worked out what the soloist was doing. He felt that the score follower should eventually be able to spot this pattern, however. If any learning had been implemented in my score follower, then I would agree that this pattern should be detectable by the learning part of the score follower.

A significant question from the third tester was about the purpose of this score follower. His question was whether the score follower was designed to be a “practice model” or a “performance model”. The difference here is that with a practice model, the score follower would more freely allow (and in fact expect) different sections to be repeated, varying levels of accuracy and more variance in tempo, and so on. This is how the third tester had interpreted the purpose of this project. The score followers had actually been designed as a performance model, though, so that the score follower was expecting a single run through of the piece, with an emphasis on getting from one end of the piece to another. The distinction between the two models is subtle but important enough to have some bearing on the probabilities that would be associated with each part of the HMM.

---

<sup>7</sup>His playing style shows a particular tendency to decorate notes following a set of prescribed rules, and he has a high degree of expectation about the style in which a musically competent accompanist should react to these note decorations

<sup>8</sup>a succession of quickly repeated notes that alternate the written note with the note directly above, for decoration of the written note

## 2. *All I Ask of You, with a set metronome tempo*

In this version, the tester was more satisfied with how the accompanist responded to the trills and other embellishments he added. This score follower worked more as expected by the tester because it ignored any input played by the soloist except that which occurred on each beat (when information is taken for the new state). Though it is not ideal that the system ignores any information from the soloist, in this particular situation it is actually a useful function of the score follower. The only embellishment that the score follower did not respond well to was appoggiaturas<sup>9</sup>. These were treated as incorrect notes. The tester rightly pointed out that the accompaniment should not become silent underneath these notes, but should perhaps carry on playing the accompaniment as expected for a few states while it tries to match the soloist back to a location on the score.

Again, a noticeable effect was that the tester adjusted to the accompaniment, so again I observed how the tester attempted to synchronise with the accompanist and to assist the accompanist in producing a musical performance. The tester commented that playing music is a co-operative process. The computer player has to be a proficient player already and cannot just react to the human player. The human player will instinctively react to the computer's playing.

The tester tried omitting playing some bars, singing them instead then starting to play again. He commented that missing a few notes out did not confuse the score follower, however missing a whole bar did, and suggested that the score follower should continue to play the accompaniment while it waited for the soloist to start playing, even if just for a bar or two. Speaking from the point of view of a vocal performer, he said it was a fairly common occurrence that singers may miss a bar or two during performance for various reasons. The accompanist would be expected to continue playing while the singer readied themselves to continue their performance again.

The tester noticed the increase in complexity of the accompaniment compared to the previous score follower, and was impressed by the smoothness of the accompaniment in the sections where it followed his performance closely, and the manner in which it recovered from errors.

A last comment on this version was that the metronome sound which had been

---

<sup>9</sup> grace notes or additional notes, which are played before the actual written note, such that the grace note is played on the beat and the actual written note is played shortly after the beat

added to help synchronise the soloist and accompanist should be turned off, as it became irritating after a while: a fair comment.

3. *All I Ask of You, with beat tracking enabled*

The tester was critical of the speed in which this score follower adapted to his playing, saying it should adjust much more rapidly to his playing.

Also he stated his belief that the score follower should only recognise correctly played notes for tempo adjustments, because the correctness of the notes is a sign that you are playing correctly at that point. The score follower does indeed work this way, and it was interesting to hear the tester advocate this without any mention of its inner workings from myself.

4. *Danse Macabre, with a set metronome tempo, and*

5. *Danse Macabre, with beat tracking enabled*

Unfortunately neither version of *Danse Macabre* worked during this test session. The tester summed up the success of this score follower by saying “I can play it when there’s no accompaniment!”

One useful observation did arise from this test session, however. The tester consistently misread a bar in the first section of the music, each time he played it. Even though this was sight-reading for the tester rather than a performance, this observation showed that even very gifted musicians can make mistakes without realising. Hence the need for a responsive accompanist is important even at a high level of performance.

6. *Further comments*

The tester made some general remarks about the user interface. He mentioned that if estimated tempos were to be displayed, then they should be displayed in the more musically conventional beats per minute format, rather than in msec per beat. He thought the interface could be more user-friendly and that it could include a display of the score on screen so that the performer did not need a separate copy of the music. This copy of the score could incorporate a marker of where the score follower thought the soloist currently was in the score, and mentioned that he thought a package called *jitter* could be used to facilitate this in Max/MSP.



### 5.2.5 Observations arising from Tester 4

Tester number four has lower musical ability than the other testers. He was only able to test the first score follower with the simple melody extract from *Twinkle Twinkle Little Star*, and not the other two pieces.

Though he was not able to test the system fully, his use of the system provided particular insights into how the score follower worked that were not forthcoming from the other testers, due to the higher error rates that occurred in his playing.

1. *Twinkle Twinkle Little Star* What was noticeable about this tester's playing, in contrast to the other testers was his uncertain tempo, a higher number of wrong notes played, and a tendency to go back and repeat sequences of a few notes, to get them right. Therefore the majority of his deviations from score were to play extra notes, or to play wrong notes.

In this case, monitoring how the score follower tracked the soloist's tempo was not very worthwhile, as the tester's tempo was too inconsistent to be measured accurately as one rate.

It was interesting to see how the *Twinkle Twinkle Little Star* score follower could cope with the repeated jumps that the tester made between different parts of the melody. Although the score follower had not been specifically programmed to be able to track backwards through a melody or to cope with sections being repeated a number of times with varying accuracy, the tester reported that the system was (in general) able to accompany him in a way that matched what accompaniment he was expecting to hear. From watching the states selected by the score follower while the tester was playing, I would agree with that conclusion.

### 5.2.6 General conclusions from tester feedback

Much of the feedback from my testers hints at the wider conclusion that it may in fact be worth implementing some form of learning of the HMM probabilities, despite the initial impression gained during preliminary research.

The score followers performed best when played by the tester with the most similarity to my own playing (the second player). When exposed to different styles of playing, the score followers needed a few adjustments in order to perform correctly, in some cases. These adjustments are detailed later in this section.

If several musicians were asked to train the HMM, the score follower is more likely to be exposed to different situations as a result rather than if just one musician trained it. This would be beneficial even if the training were to take place just for a certain initial period.

An unforeseen but fascinating result of the testers' experimentation with my score follower system was the emerging of the co-operative nature of this domain in real-life, and the importance of feedback and communication between two musicians. Roger Dannenberg has commented on a similar finding whilst testing his score follower in an ensemble situation (Dannenberg, 2000) (p. 3):

Early on, Lorin [Grubb] and I were playing trios with the computer, making intentional errors to test the system. We found that if we deliberately diverged so as to be playing in two different places, the computer could not decide who to follow. Even if one of us played normally and the other made an abrupt departure from the normal tempo, the computer would *not* always follow the "normal" player. In a moment of inspiration, we realized that the computer did not consider itself to be a member of the ensemble. We changed that, and then the computer performed much more reasonably. Here is why this worked: When the computer became a first-class member of the ensemble and one of us diverged, there were still two members playing together normally, e.g. Lorin and the computer. The computer, hearing two members performing together, would ignore the third.

While the emphasis found in previous research, and in this project, has been on the artificial accompanist following the soloist, I believe that a design with more focus on co-operation between soloist and accompanist would be worth further investigation, although as of the time of writing, little reference to this approach can be found in the literature beyond Dannenberg's work.

Reflecting on why the score followers for *Danse Macabre* failed in some tests and performed slightly better in others, the difference is likely to be due to the playing style of the performer. Problems with different ways of playing notes have been reported in previous score following research (for example Orio and Schwarz (2001) reported problems with legato). So I was anticipating that there might be some difference in interpretation between legato melodies such as that for *All I Ask of You* and melodies with much staccato, such as *Danse Macabre*. The staccato played notes in the *Danse Macabre* extract, when played in a very short style, did not result in accurate state matching by the score follower. The score follower performed better when the staccato notes were played for a slightly longer duration (as Tester 2 did).

Different testers had different interpretations of playing staccato and I had not re-

alised the effect that this would have had on the performance of the score follower. Again this highlights the usefulness of having several musicians' influence on the development of the musicality of the score follower (as is the case in real life; a human musician will usually benefit from a number of different influences rather than learning with just one teacher).

A change that was made as a result of this poor performance in testing was to change how the score follower dealt with incoming input from the soloist. Previously if the soloist had stopped playing a note, the input from the soloist was changed to be the emission '12', representing silence from the soloist. For the *Danse Macabre* score followers, this was removed from the score follower so that it no longer recognised when the soloist was not playing. This meant that the differences in staccato playing were less of an issue and hence the score follower performed better overall in accompanying this piece. (I was only able to get the first tester to re-test this version, but he reported an improvement in performance).

A notable side-effect was that the system could no longer use the observation of silence to signify the end of the piece, so the emission probabilities for the score follower had to be altered for the end states before the score follower could adequately locate the end of the piece again.

Although this change removed much of the problems with staccato notes, it would mean that the score follower could not be used for a piece with rests (silences) in the solo melody. So future work could include a search for a better way to cope with rests, or with staccato notes.

Another change made after testing was the addition of a localised search for the next state, rather than a global search. This is described in more detail in the Implementation chapter. This new version was unfortunately not ready in time for the testers to try.

As a final observation, my score followers in general performed better with musicians of lower rather than higher ability. They responded better to inconsistent tempos and errors, rather than correct playing and decorative embellishments. This is probably partly due to a slight bias in the way I have set the HMM probabilities, where I have approached this more from the point of view of recovering from errors rather than dealing with note decorations and embellishments. It is pleasing, however, to see that most of the score followers generally performed well in responding to tester errors of different types, and coped with note embellishments to a certain degree.

---

The score followers in this project were tested with varying levels of success being reported during subjective and objective testing. The overall results are that the score followers achieved some notable successes in accompanying the soloists, particularly for simpler or shorter melodies, and that the . Nevertheless, there is much potential for further work on setting the HMM probabilities to improve the musicality and accuracy of accompaniment, especially for more complex melodies.

# Chapter 6

## Discussion and Conclusions

During this project, a number of score followers with varying levels of functionality have been designed, implemented and tested. This chapter reflects on the capabilities of the score followers produced during this project and in particular whether the score followers have met the requirements specified at the start of the project. Comments are made on how the implementation of the score followers could have been improved, from the perspective afforded by hindsight. Suggestions are also made for future work that could be carried out to develop the score followers with more advanced functionality.

The project hypothesis, testing the suitability of using Hidden Markov Models for score following, can now be evaluated in the light of the evidence gathered during development and testing of the above score followers.

The thesis ends with a concluding summary of what has been achieved during this twelve-week research project.

### **6.1 Capabilities of the score followers developed in this project**

My score follower uses an HMM to follow a musical soloist through the score of a piece, and produce musically acceptable accompaniment, even if the soloist's performance is occasionally inaccurate or embellished. Evidence for this can be found in the previous chapters of this document.

Currently versions exist that are programmed to recognise extracts from the traditional melody of *Twinkle Twinkle Little Star* and from Andrew Lloyd-Webber's *All I*

*Ask of You*. The last version of the *Danse Macabre* score follower makes a reasonable attempt at tracing the progress of the soloist through the course of the extract from *Danse Macabre*, although latency issues prevent it from producing a musically acceptable accompaniment.

It matches the performer's interpretation in terms of the volume the soloist is playing at, and the various score followers make a reasonable attempt at gauging the tempo at which the soloist is playing at. They are good at detecting changes in tempo although further work is required to detect the magnitude of the change in tempo more accurately. This aspect of the score follower is related to the ability to track the performer accurately through the piece, so as the HMM probabilities are more accurately set, this aspect of the score follower works more competently.

Each score follower is able to produce chordal accompaniment, such that more than one note is sounding at a time. The accompaniment moves in a smooth fashion with legato (slurred together) notes played where necessary. In more advanced versions, the score follower can cope with producing accompaniment that moves independently of the soloist: so for a given state that the soloist is in, the accompanist can produce accompaniment that changes during the time the soloist stays in that state. This allows for much more complex accompaniment to be produceable and also gives more flexibility in what can be chosen as representative by each HMM state.

As discussed in the previous chapter, the systems evaluate to be comparable in performance to other score followers evaluated during MIREX (2006b), although they do not perform as well as the best system presented in this conference, by Arshia Cont and Diemo Schwarz at IRCAM.

The standard of accompaniment produced by my score following systems for the two simpler pieces was judged by human musicians to be comparable to that of human accompanists, although as pieces became more complex the score followers struggled to maintain this standard of performance.

## 6.2 Meeting the specified requirements

It was specified earlier in this document that the score follower would require:

- A way of receiving musical input from the soloist
- A way of processing musical input from the soloist
- A way of implementing an HMM to analyse musical input from the soloist

- A way of generating musical accompaniment in real-time

The resulting score followers do indeed satisfy all of these requirements.

### 6.2.1 Evaluation of project hypothesis

To evaluate my hypothesis that:

Using an HMM representation of the sequence of states in 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

I considered the subjective evaluation from human testers and the statistics achieved based on how well the system does at score following. I also considered my practical experiences in implementing the score followers.

From the results obtained and the feedback from other testers, I conclude that the HMM-based score followers can provide real-time accompaniment to a human performer. The performance was better in some cases than others, although the HMM probabilities and exact implementations of the score could probably be set more accurately in the cases where the score follower did not produce such a good implementation.

The use of an HMM considerably simplified the implementation of the score following part of my systems. I did not have to give strong consideration to how the score follower tracked the soloist through the score, beyond implementing the HMM and the associated probabilities. So the performance of the resulting score followers is pleasing, and I feel that the HMM representation of the domain was very suitable, as my hypothesis states. With further experimentation as to the most appropriate settings for the HMM probabilities, and perhaps implementation of some automatic training for individual pieces or individual performance styles, better results should be possible for the score followers that did not perform so well.

It is necessary to acknowledge here that the score follower performed more accurately for HMMs with a large number of states when it had received some assistance in addition to the estimated probabilities returned by the Viterbi algorithm. As described in the Implementation chapter, my score follower gave more weight to the probability associated with the state located next to the current state in the score, in order to encourage the score follower to work in a more linear and smooth fashion. This meant that the HMM probabilities were overridden to some extent. However as discussed

previously, I believe that this is a simplification of the effect that more accurate HMM probabilities would have, had there been more time to experiment with probabilities or to implement training of the HMM probabilities.

One test carried out by testers that had not been anticipated was to see how the score follower would deal with the soloist moving in the wrong direction in the score<sup>1</sup>. To play a musical score from right to left is a rare mistake in my musical experience; indeed it is slightly counter-intuitive for Western musicians as we are used to reading from left to right, whether it be music or written text. It is more common, though, to repeat small sequences of notes during practice, or to move backwards in the score by jumping to a previous location by mistake, or by performing a repeat section. This was included in the controlled testing that was carried out.

State transitions that dealt with backwards movements in the score were not explicitly included in this project (being guided in this respect by the work in Orio et al. (2003)). The HMM structure coped well with this situation during testing, though. Here the score follower was able to estimate the current location correctly in most cases, by analysing the soloist's recent playing. This was a benefit derived from the use of an HMM that had not originally been anticipated.

A part of my hypothesis that needs further investigation is in the efficiency of my score followers. The more complex score followers in this project demonstrate how latency issues can severely disrupt the performance of the accompaniment by the score follower. Careful consideration needs to be made as to how to overcome the large calculation effort involved in larger scale score models (perhaps by using an alternative to the Viterbi algorithm or by adjusting it further with more efficiency optimisations). I note here that the concern with efficiency is not with the general use of an HMM structure, but specifically with the extraction of information from the HMM by calculations with the HMM probabilities.

So the findings of this research project are that the project hypothesis has been proven to some extent, and that an HMM is a good way to implement score following, but that the HMM probabilities need to be carefully chosen. Also there are concerns about the efficiency of using the Viterbi algorithm to track the soloist through the score.

Raphael (2001a) now prefers to use Bayesian networks as the statistical tool by which he implements score following, although IRCAM's *suivi* (a commercial score follower for Max/MSP) is still based on their work in using HMMs for score following

---

<sup>1</sup>My testers were deliberately not restricted as to what errors or embellishments they were asked to include; instead they were asked merely to experiment with the system as they saw fit, using their musical knowledge and imagination



(Orio and Dechelle, 2001; Orio et al., 2003).

### 6.3 What could be improved upon or done differently, in hindsight

- Locating the soloist's current state exactly, for more complex, long or repetitive melodies.

*Danse Macabre* was selected specifically to test my piece as a more challenging solo melody to track the soloist through. This is because it incorporates much repetition of note sequences, and some stylistic variation from the other scored pieces *Twinkle Twinkle Little Star* and *All I Ask of You*.

Finding the right state in *Danse Macabre* when it is played correctly is a challenge for earlier versions of the score follower, as is dealing with mistakes and other situations where the HMM *ghost* states are used. More success was achieved in later versions of the *Danse Macabre* score follower, where a greater degree of accuracy was reported during objective testing (as is discussed in more detail in the Testing Chapter). These more advanced score followers restricted the states considered by the score follower as potential next states to a set of states, local to the current position. These were better at tracking the soloist through the score when the soloist played in a generally linear fashion through the piece. Some extra work here would enable the score followers to deal better with situations where the soloist moved to a location far away from the previous location. Such situations would include the scenario where two pages in the music were turned over at the same time and hence a page was missed out, or if the soloist skipped to a different location in the score by mistake.

- Latency issues and efficiency of calculation

A more efficient way of finding the current state is needed, to implement real-time accompaniment more successfully.

Pardo and Birmingham (2005) found that implementing the Forwards-Backwards algorithm gave a similar level of performance to the Viterbi algorithm, and in some cases the Viterbi algorithm outperformed the Forwards-Backwards algorithm.

Orio and Dechelle (2001) describe how decoding is a quicker and more error-free method than Viterbi, with a similar computational cost. I found, though, that the computational cost for Viterbi was high for larger scores with more states. This was due to the high computational cost<sup>2</sup> associated with the calculations that were necessary for the Viterbi algorithm, rather than my choice of state model. Nevertheless there may be a decoding technique that is more computationally efficient technique, and research into this would be worthwhile.

- Ornamentation and Embellishments

Although the score followers in this project were reasonably capable of detecting and recovering from errors in the soloist's playing, in general they did not perform accompaniment particularly well underneath ornamented notes and added embellishments. This was especially noticeable on the *Twinkle Twinkle Little Star* score follower, but was one of the improvements noticed in the more advanced score followers such as those for *All I Ask of You*.

It is to be acknowledged that in my design of the HMM probabilities and during development, I concentrated more on error handling rather than dealing with musical embellishments. The way in which musical embellishments are interpreted by a score follower is worthy of consideration separate from error-handling, but at the time of development this had not been seen as necessary. With more experimentation and testing in this area, the probabilities in the HMM could be adapted to better cope with musical embellishments. The state representation could also be altered to recognise trills, turns and other embellishments as a single musical event rather than as a sequence of many musical events.

- Measuring the current tempo when the HMM is not finding the soloist's current location correctly

These score followers used a simple implementation of beat tracking. In testing, the beat tracking appeared to work quite well for the simple melodies, however there was a problem with more complex melodies. This was because of the reliance on the state to be located correctly in order to gauge note lengths, and therefore the expected distance between observations. An alternative implementation of beat tracking that could have been tried was to include the use of the previous tempo to work out roughly how many beats had passed between two

---

<sup>2</sup>Of the order of  $N^M$ , where N is the number of states in the HMM and M is the number of possible observations

note inputs, as opposed to relying on the HMM to have estimated the next state correctly in all occasions.

## 6.4 Suggestions for possible future work

The following extensions could be feasibly added to the score following systems created in this project:

- Automatically extracting (learning) of the score and HMM structure from a MIDI file. (The score and HMM structure are currently programmed in by hand.)

It would be simple to add this as an extension to the current score followers, as this information is encoded in separate text files which the score followers read in when the program is started. All that would be needed is a way of generating the content of these text automatically, which could be done in Max/MSP or alternatively in another program such as Matlab.

During the development phase, some initial experimentation with this was carried out in Matlab, using the MIDI toolbox for Matlab (described in Chapter 2). Some text files were generated for the HMM structure from MIDI files although they contained a number of errors, due to errors in my Matlab code. With a little further work, though, this should definitely be possible to implement fully.

- Update the system to be able to receive musical input from instruments rather than just from a MIDI keyboard (for example by playing a flute into a microphone)

The processing of an audio musical signal is part of the MSP part of the Max/MSP program, so again it is reasonable to consider adding this to the score followers developed in this project.

This project deliberately did not address how to implement this signal processing, but instead considered purely MIDI input, for reasons of project scope given the time constraints.

A number of research efforts have attempted to implement signal processing (for example Raphael (2001b); Orio et al. (2003); Orio and Schwarz (2001)), but it is acknowledged in a summary of score following research (Orio et al., 2003) that this adds an extra degree of complexity to the implementation of the

score follower. The details published in the literature would, though, be a useful starting guide if adding this functionality to my score followers.

- See if machine learning techniques could help the score follower to learn a particular performer's common embellishments and/or mistakes (by training the HMM)

HMMs can be trained such that the HMM probabilities are refined, using algorithms such as the **Baum-Welch** algorithm (Durbin et al., 1998; Rabiner, 1989). With some additional implementation effort, HMM training can be added to the score followers developed in this project, in a similar manner to how the HMM structures and use of Viterbi algorithm were implemented.

Although applying traditional HMM training methods to the domain of score following has been considered to be less useful than anticipated (Schwarz et al., 2004; Smaill, 2007), there has been recent research at IRCAM into new HMM training methods specifically for the domain of score following (Cont et al., 2004). It would be interesting to see HMM training could improve my score followers. Effective training could improve the accuracy of the probabilities for the HMM and, as a consequence, also improve the quality of performance for this project's score follower.

- Add knowledge to the score follower that allows it to respond to musical cues and feedback from the soloist, to co-operate with the soloist and achieve a joint performance

This arose from observations of how my testers were attempting to interact with the score followers and give them musical cues. It would be possible to extend my system to interpret such musical cues, but would require a concerted research effort into how and when musicians use such cues.

- Extend the project further to be helpful for teaching purposes by adding a "tutor" that gives feedback to the performer on how they deviated from the score (to learn from mistakes)

For example the student could be accompanied by the system during a practice performance. The system could be adapted to produce feedback to the student after a practice performance. This feedback would tell them where their performance deviated from the ideal model of the score (e.g. "At point S1 in the score the tempo of your performance sped up", "At points S2, S3 and S4, the

notes were not held for their full note value”, etc). This would help the student learn from mistakes that they may not otherwise have noticed. If the system is developed to the point that it can be trained for a specific performer, then this error recognition would help identify common mistakes made.

This should be relatively feasible and could be implemented by a feedback mechanism that is triggered whenever the HMM enters a *ghost* state. The system would record the fact that this *ghost* state had been entered and would record what the next state transition was (which would help describe the type of error that was made).

## 6.5 Concluding remarks: What has this work achieved

This project has examined the effectiveness of Hidden Markov Models for score following and concluded that they are a useful tool with which to implement score following systems.

During the lifetime of this project, HMM-based score followers have been developed in the interactive real-time music processing environment Max/MSP. These score followers incorporate various enhancements such as beat tracking, the handling of longer scores and the ability to produce complex accompaniment that changes whilst the soloist remains in a particular state.

During development, a Hidden Markov Model structure was partially implemented in Max/MSP to model the scores and to carry out the Viterbi algorithm.

My score followers are able to determine which HMM state the soloist is currently in, by analysing what the soloist has just played against a specified score. They can then play the appropriate accompaniment for that state.

Performances by each score follower have been evaluated subjectively by testers of varying musical ability and experience, and also by the objective criteria that was used to evaluate score followers at the Music Information Retrieval Evaluation eXchange conference of 2006. Overall the score followers have been able to produce real-time accompaniment to a human soloist, playing one of three different pieces, of varying complexity. In most cases the accompaniment was musically appropriate throughout the performance of the piece, even when the soloist performer deviated from the score by making errors or adding embellishments to the music performed.

# Bibliography

- Bloch, J. and Dannenberg, R. B. (1985) Real-Time Accompaniment of Polyphonic Keyboard Performance. *Proceedings of the 1985 International Computer Music Conference*, pp. 279–290.
- Cano, P., Loscos, A. and Bonada, J. (1999) Score Performance Matching using HMMs. *Proceedings of the 1999 International Computer Music Conference*.
- Cont, A. and Schwarz, D. (2006) Score Following Proposal. [http://www.music-ir.org/mirex2006/index.php/Score\\_Following\\_Proposal](http://www.music-ir.org/mirex2006/index.php/Score_Following_Proposal).
- Cont, A., Schwarz, D. and Schnell, N. (2004) Training IRCAMs Score Follower. *AAAI Fall Symposium on Style and Meaning in Art, Language and Music*.
- Dannenberg, R. B. (1984) An On-line Algorithm for Real-time Accompaniment. *Proceedings of the 1984 International Computer Music Conference*.
- Dannenberg, R. B. (1989) Real-time Scheduling and Computer Accompaniment. *MIT Press Series in System Development Foundation Benchmark*, pp. 225–261.
- Dannenberg, R. B. (2000) Artificial Intelligence, Machine Learning, and Music Understanding. *Proceedings of the 2000 Brazilian Symposium on Computer Music: Arquivos do Simpósio Brasileiro de Computação Musical (SBCM)*.
- Dixon, S. (2001) Automatic Extraction of Tempo and Beat From Expressive Performances. *Journal of New Music Research*, vol. 30(1):pp. 39–58.
- Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998) *Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press.
- Grubb, L. and Dannenberg, R. B. (1998) Enhanced Vocal Performance Tracking Using Multiple Information Sources. *Proceedings of the 1998 International Computer Music Conference*, pp. 37–44.
- MIREX (2006a) Audio Beat Tracking. [http://www.music-ir.org/mirex2006/index.php/Audio\\_Beat\\_Tracking](http://www.music-ir.org/mirex2006/index.php/Audio_Beat_Tracking).
- MIREX (2006b) Score Following Results. [http://www.music-ir.org/mirex2006/index.php/Score\\_Following\\_Results](http://www.music-ir.org/mirex2006/index.php/Score_Following_Results).
- Orio, N. and Dechelle, F. (2001) Score Following Using Spectral Analysis and Hidden Markov Models. *Proceedings of the 2001 International Computer Music Conference*.

- Orio, N., Lemouton, S., Schwarz, D. and Schnell, N. (2003) Score Following: State of the Art and New Developments. *New Interfaces for Musical Expression, Montreal*.
- Orio, N. and Schwarz, D. (2001) Alignment of Monophonic and Polyphonic Music to a Score. *Proceedings of the 2001 International Computer Music Conference*.
- Pardo, B. and Birmingham, W. (2005) Modeling Form for On-line Following of Musical Performances. *Proceedings of the Twentieth National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania*.
- Rabiner, L. R. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, vol. 77(2):pp. 257–286.
- Raphael, C. (1999) Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21(4):pp. 360–370.
- Raphael, C. (2001a) A Bayesian Network for Real-Time Musical Accompaniment. *Neural Information Processing Systems*.
- Raphael, C. (2001b) Music Plus One: A System for Flexible and Expressive Musical Accompaniment. *In Proceedings of the 2001 International Computer Music Conference*.
- Raphael, C. (2007) Personal email correspondence.
- Raphael, C., Cont, A., Schwarz, D., Litterst, G., West, K., Van Schuetterhoef, A., Good, M. and Downie, S. (2006) Score Following (Discussion). [http://www.music-ir.org/mirex2006/index.php?title=Score\\_Following](http://www.music-ir.org/mirex2006/index.php?title=Score_Following).
- Schwarz, D., Orio, N. and Schnell, N. (2004) Robust Polyphonic Midi Score Following with Hidden Markov Models. *Proceedings of the 2004 International Computer Music Conference*.
- Smaill, A. (2007) Personal communication regarding an IRCAM presentation on score following, attended circa 2002.
- Toivainen, P. (2007) Personal email correspondence.
- Vercoe, B. L. (1984) The Synthetic Performer in the Context of Live Performance. *Proceedings of the 1984 International Computer Music Conference*.
- Vercoe, B. L. and Puckette, M. S. (1985) Synthetic Rehearsal: Training the Synthetic Performer. *Proceedings of the 1985 International Computer Music Conference*.
- Zicarelli, D., Taylor, G., Bernstein, J., Schabtach, A., Dudas, R. and DuBois, R. L. (2006) Max/MSP v4.6 Tutorials and Documentation. Available from <http://www.cycling74.com/download/maxmsp463doc.zip>.

# Appendices

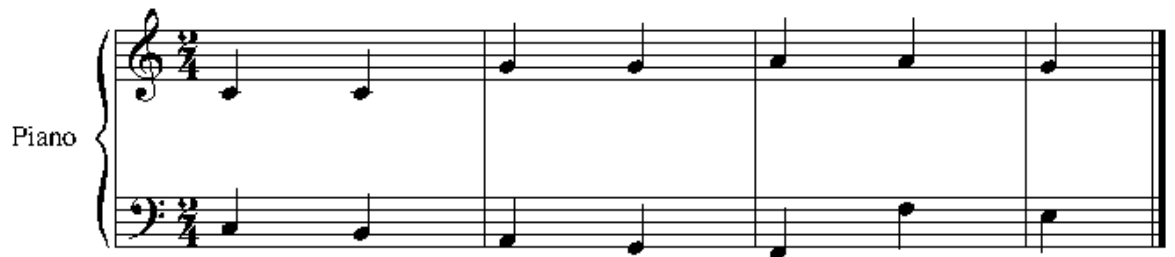


# Appendix A

## Repertoire for my score follower

1. *Twinkle Twinkle Little Star* extract with simple monophonic bass line accompaniment
2. *Twinkle Twinkle Little Star* extract with simple chordal accompaniment
3. *All I Ask of You* extract with simple chordal accompaniment
4. *All I Ask of You* extract with more complex chordal accompaniment
5. *Danse Macabre* extract

## Twinkle Twinkle Little Star



Piano

The musical score for 'Twinkle Twinkle Little Star' is presented in a grand staff format. The treble clef staff contains the melody, and the bass clef staff contains a simple monophonic bass line accompaniment. The key signature is one flat (B-flat), and the time signature is 5/4. The melody consists of a sequence of eighth notes: G4, A4, B4, A4, G4, F4, E4, D4. The bass line consists of a sequence of quarter notes: G3, F3, E3, D3, C3, B2, A2, G2.

Figure A.1: *Twinkle Twinkle Little Star* extract with simple monophonic bass line accompaniment

## Twinkle Twinkle Little Star



Piano

The musical score for 'Twinkle Twinkle Little Star' is presented in a grand staff format. The treble clef staff contains the melody, and the bass clef staff contains a simple chordal accompaniment. The key signature is one flat (B-flat), and the time signature is 5/4. The melody is identical to Figure A.1. The chordal accompaniment consists of a sequence of chords: G3, F3, E3, D3, C3, B2, A2, G2.

Figure A.2: *Twinkle Twinkle Little Star* extract with simple chordal accompaniment

# All I ask of you

Andrew Lloyd-Webber

Piano

The image displays a piano accompaniment extract for the song "All I Ask of You" by Andrew Lloyd-Webber. The score is written for piano and consists of two systems of music. The first system is labeled "Piano" on the left. Both systems feature a treble clef staff with a key signature of two flats (B-flat and E-flat) and a common time signature (C). The melody in the treble clef is simple and melodic, while the bass clef provides a simple chordal accompaniment. The first system contains four measures, and the second system contains five measures, ending with a double bar line.

Figure A.3: *All I Ask of You* extract with simple chordal accompaniment

# All I ask of you

Andrew Lloyd-Webber

Piano

The image displays a piano accompaniment for the song 'All I Ask of You' by Andrew Lloyd-Webber. It consists of three systems of music, each with a grand staff (treble and bass clefs). The first system is labeled 'Piano'. The music is in a key signature of two flats (B-flat and E-flat) and a common time signature (C). The melody in the treble clef is simple and lyrical, while the bass clef provides a more complex chordal accompaniment with various textures, including block chords and moving lines. The second system continues the piece with similar complexity. The third system shows the piece concluding with a final chord in the bass clef and a whole rest in the treble clef.

Figure A.4: *All I Ask of You* extract with more complex chordal accompaniment

# Danse macabre

Saint Saens

The image displays three systems of musical notation for the Recorder and Piano parts of the 'Danse Macabre' by Saint-Saëns. The music is in 3/4 time and B-flat major.

- System 1:** Recorder part starts with a forte (*f*) dynamic. The piano part is mostly silent, with some bass line activity.
- System 2:** Recorder part starts with a piano (*p*) dynamic. The piano part features a series of chords in the right hand and a steady bass line.
- System 3:** Recorder part continues with a piano (*p*) dynamic. The piano part continues with chords and a bass line.

Figure A.5: *Danse Macabre* extract

## Appendix B

# An example of the Hidden Markov Model probabilities: Twinkle Twinkle Little Star

N.B.  $N$ ,  $M$ ,  $A$ ,  $B$ ,  $\pi$  and  $V$  refer to the abbreviations used in Rabiner (1989) for each part of the HMM settings described in Chapter 2, such that  $\lambda = \{N, M, A, B, \pi, V\}$

$N$  = no of states = 15

$M$  = no of possible observations = 12

$A$  = state transition probabilities = (see B.1)

$B$  = observation probabilities = (see B.2)

$\pi$  = initial probabilities = (see B.3)

$V$  = possible observations =  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.0	0.45	0.1	0.45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.1	0.81	0.0	0.05	0.0	0.01	0.0	0.01	0.0	0.01	0.0	0.01	0.0
2	0.0	0.0	0.0	0.45	0.1	0.45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.1	0.82	0.0	0.05	0.0	0.01	0.0	0.01	0.0	0.01	0.0
4	0.0	0.0	0.0	0.0	0.0	0.45	0.1	0.45	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.83	0.0	0.05	0.0	0.01	0.0	0.01	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.45	0.1	0.45	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.84	0.0	0.05	0.0	0.01	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.45	0.1	0.45	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.85	0.0	0.05	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.45	0.1	0.45	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.9
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Table B.1: state\_transitions.txt: The probabilities associated with transitions from one state to another (A)

Emission:	0	1	2	3	4	5	6	7	8	9	10	11
0	0.0	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.09
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.09
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.0	0.091	0.091	0.091	0.09
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
6	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.0	0.091	0.091	0.091	0.09
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
8	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.0	0.091	0.09
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
10	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.0	0.091	0.09
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
12	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.0	0.091	0.091	0.091	0.09
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
14	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0834	0.0834	0.0834	0.0834

Table B.2: emission\_probabilities.txt The probabilities of seeing each observation in a state ( $B$ )



State	Probability
0	0.1
1	0.75
2	0.0
3	0.05
4	0.0
5	0.04
6	0.0
7	0.03
8	0.0
9	0.02
10	0.0
11	0.007
12	0.0
13	0.003
14	0.0

Table B.3: initial\_probs.txt: The probability of starting in each state ( $\pi$ )

# Appendix C

## Technical program detail

My score follower system is written in Max/MSP. This is an interactive music-processing environment that runs in real-time.

Rather than program code, in Max/MSP the program is carried out by passing information throughout the program, to various *patcher objects*. These patcher objects are analogous to procedures or functions in more conventional programming languages.

This appendix shows screenshots from the main score follower program windows. These screenshots express the workings of the score follower program as a code listing would do for a Java program or similar.

The screenshots included below show how the fundamental parts of the program have been programmed.

To interpret how each patcher in the program works: where there is a choice of two or more information flows to choose, by default Max/MSP passes on the right-most information first, then works right-to-left. It operates a depth-first strategy, meaning that all information flow possible is carried out in one particular branch before the program moves onto the next branch.

In these screenshots, each window is an individual patcher. Each patcher is pictured with its sub-patchers (patchers that are called during the operation of that patcher).

In general, information flows from the *inlet* at the top of the patcher, through the *patch cords* (black lines connecting each box in the patcher). Each box that the information reaches does some processing on that information.

At the bottom of the patcher is usually an *outlet* that passes information onto the next patcher (unless this patcher does not send out information to another patcher).

The other main ways of getting information in and out of these patchers are:

- *notein* and *noteout* objects pick up MIDI note messages from the MIDI keyboard

and send MIDI note messages to the keyboard, respectively.

- send and receive objects exchange information between each other. They have global scope, across the program.
- value objects are the Max/MSP variable. Values are available globally. Information stored in a value is retrieved by sending a 'bang' message to a value object.
- coll objects are the Max/MSP array structure. As for values, coll data is available globally.

### Notes on how Max/MSP works

- Max/MSP documents are called Patches.
- A Patch is a collection of Objects that have Input/Output inlets/outlets.
- Functionally, a Patch can be thought of as performing a task or procedure
- There are a large number of Objects available with specific functions e.g. midi-out generates MIDI output.
- Objects pass messages to each other through their inlets/outlets. This is how information flows through the Patch.
- Messages can take the form of numbers, words (called Symbols in Max/MSP), a list of numbers e.g. 1 5 85 12, or a combination of numbers and words.
- One message which is particularly useful is the *bang* message. This tells the receiving object to immediately carry out the function that that object is designed to do. So the *bang* message is useful for, among other things, triggering off sound generation after having made the appropriate observation.
- Objects can take arguments (for some objects this is necessary for it to operate).
- A Patch can be used inside another Patch by calling it by name within that Patch.

Max/MSP does in fact have a very simple score follower object available for use. The **follow** object is able to process an incoming stream of notes and make some comparisons against a pre-programmed sequence of expected notes. However this object is

limited in its applications, as acknowledged in the Max/MSP documentation (Zicarelli et al., 2006) (Tutorial 35) and it is considered to perform poorly as score following scenarios (Raphael, 2007). Also, if I had used this object, I would not be able to test my hypothesis about the suitability of Hidden Markov Models for performing score following. So whilst I acknowledge that this object is available, I did not feel it was necessary to make use of it in this project.

The files that make up my score followers can be downloaded from <http://homepages.inf.ed.ac.uk/s0676484>.

Extensive Max/MSP documentation and tutorials (Zicarelli et al., 2006), as well as a trial version of the Max/MSP software (for Windows or Mac OS), can be downloaded from <http://www.cycling74.com>.

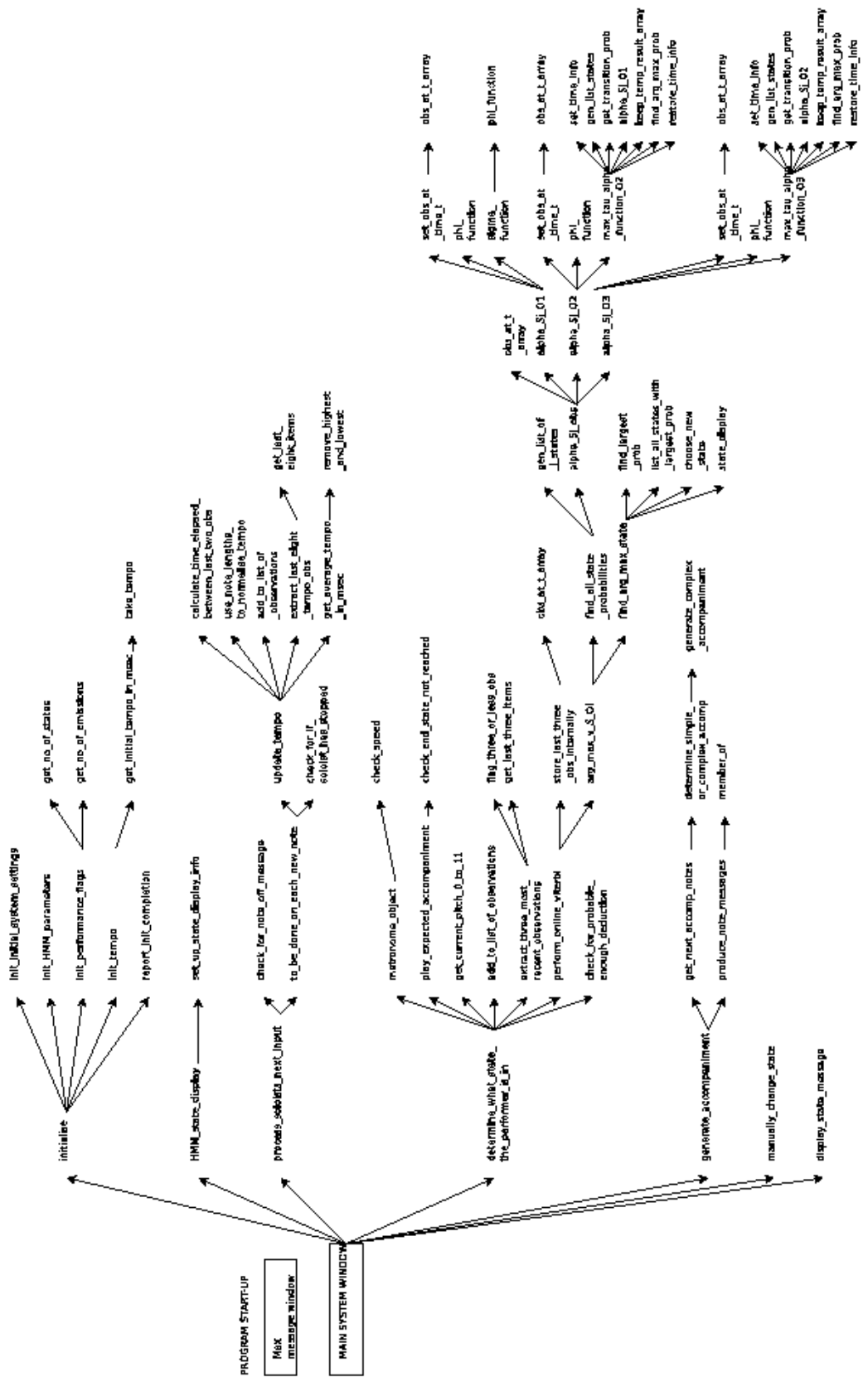




Figure C.2: My score follower on startup

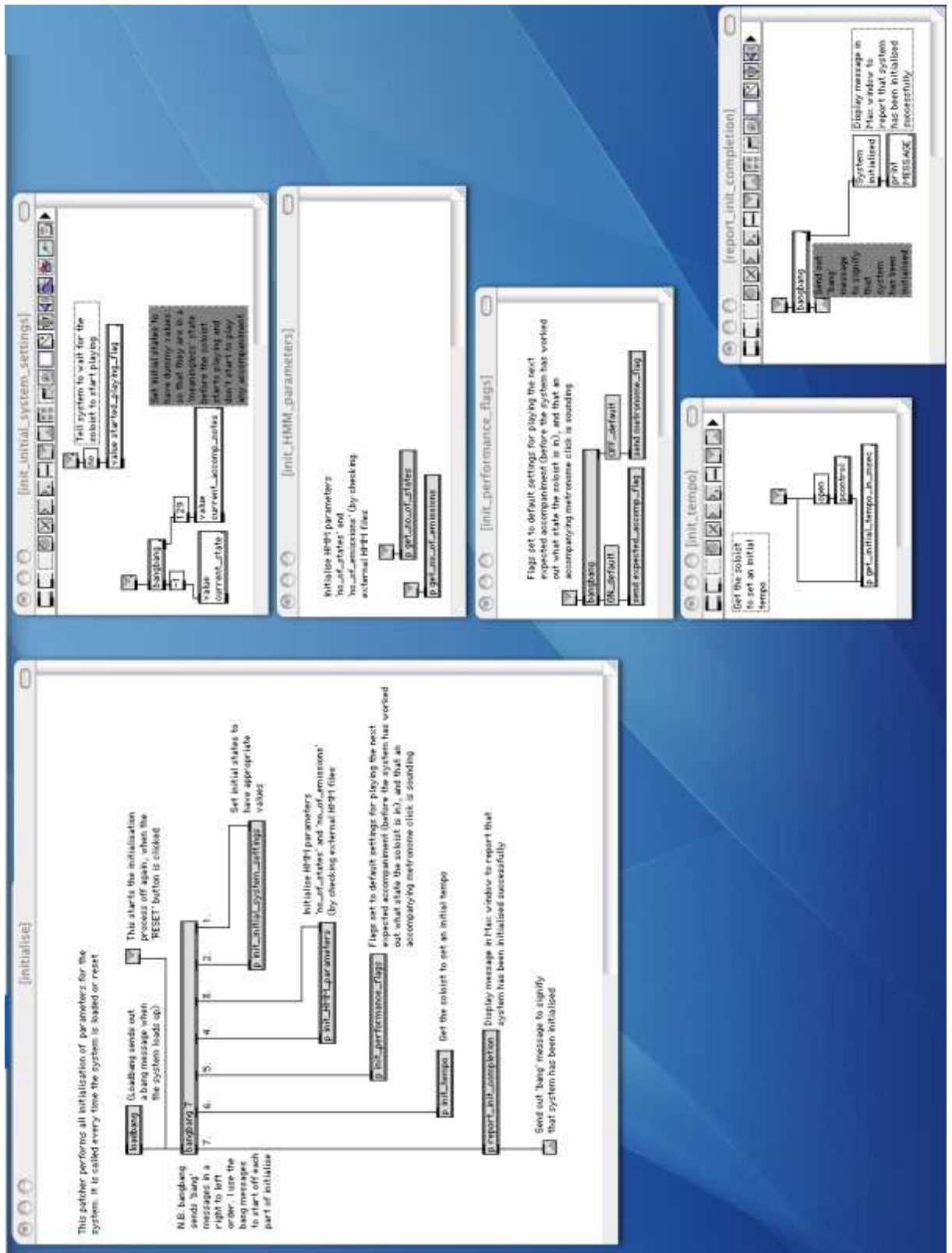


Figure C.3: Program initialisation patcher

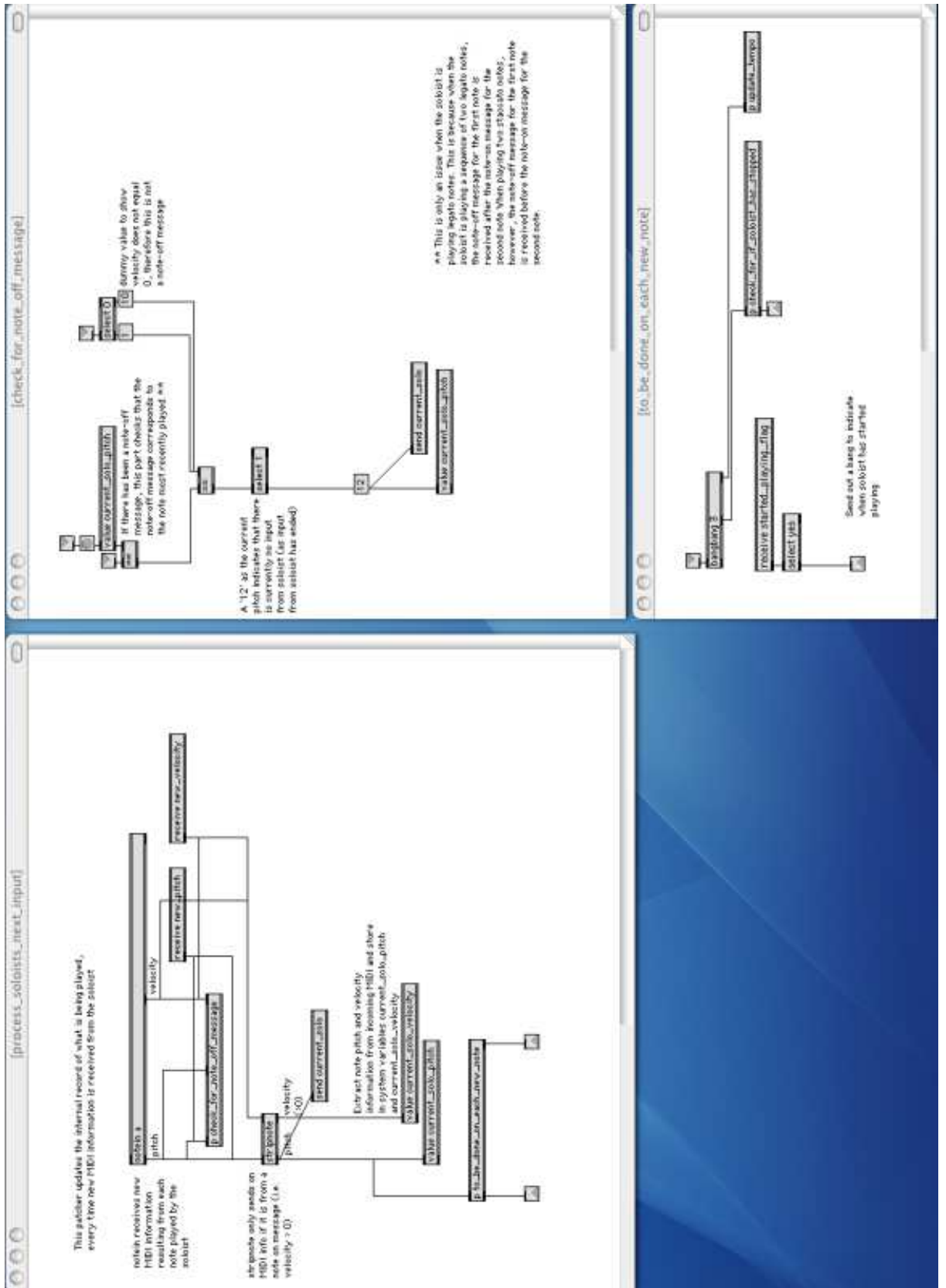


Figure C.4: How the score follower processes new input from the soloist







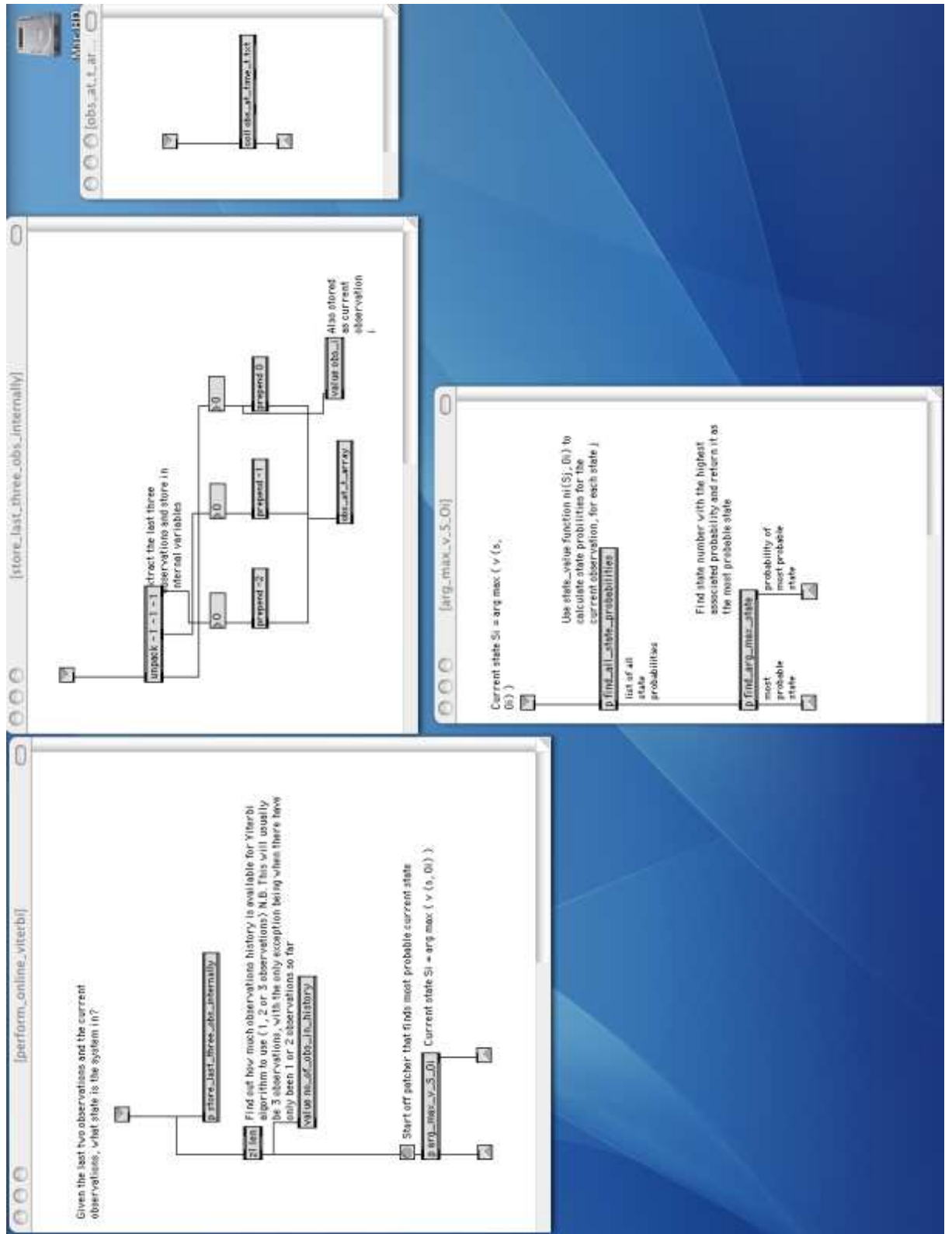


Figure C.7: The viterbi algorithm as implemented in the score follower



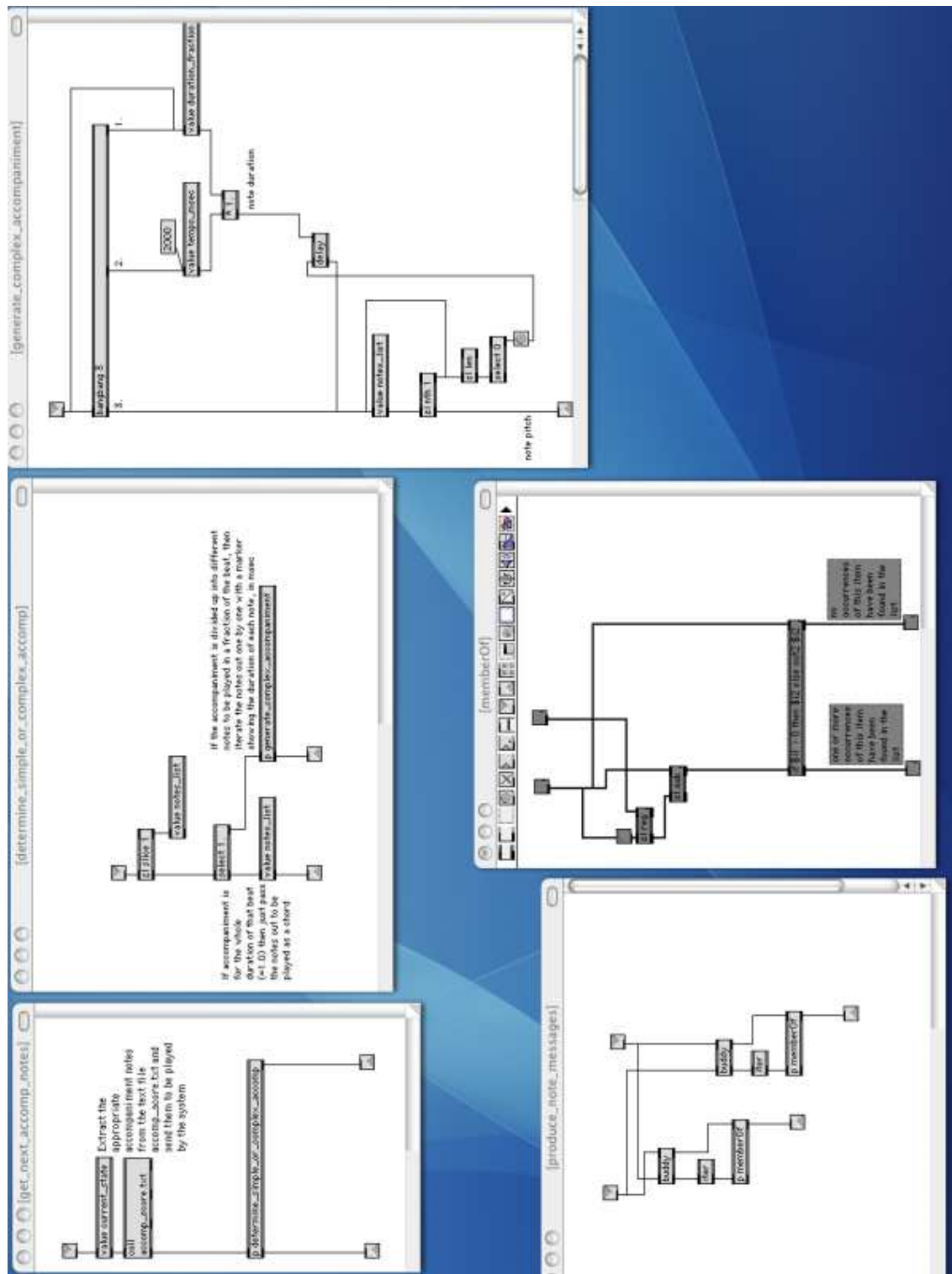


Figure C.9: How the score follower generates accompaniment (part 2)

# Appendix D

## Test melodies

During the objective testing process, the score followers were tested five times, with specific amendments being made to the melodies. Details of these amendments are as follows.

For *Twinkle Twinkle Little Star*, the following adaptations were made for testing purposes:

- Selected errors added to melody:
  - An extra note (E) added between the second C of bar 1 and the first G of bar 2
  - The wrong note played on bar 2 beat 2 (a G# instead of a G)
  - Skipping out the A at bar 3 beat 2
- Selected embellishments added to melody:
  - An *acciaccatura* (Definition E.1) rapidly played note added just before the scored note, played on the beat is added to the C at bar 1 beat 2
  - A *trill* (Definition E.6) is added to the G at bar 2 beat 2
  - A *turn* (Definition E.7) is added between the second A of bar 3 and the G in bar 4
- Selected tempo adjustments made whilst playing the melody:
  - The system was told that the soloist would be playing at 500 milliseconds per beat (ms/beat), but the soloist actually played at 600 ms/beat.
  - At the start of bar 3 the soloist's tempo changes suddenly to 400 ms/beat.

For *All I Ask of You*, the following adaptations were made for testing purposes:

- Selected errors added to melody:
  - Skipping out the C at bar 1 beat 2
  - The wrong note played on bar 2 beat 2 (a B natural instead of a Bb)
  - Bars 3 to 4 repeated (play bar 3, then bar 4, then bar 3, then bar 4)
  - During the first repeat of bars 3-4, the rhythm in bar three is played incorrectly: a dotted minim (3 beats) followed by a crotchet (1 beat) rather than two minims (2 beats each)
  - Wrong notes played in bar 6. The F (beat 2) becomes a D, the Bb becomes a C and the D becomes an E natural.
  - Skipping out the first two notes of bar 7 (so that that part of the bar is missed out entirely and only two beats of bar 7 are actually played, hence the F on bar 7 beat 3 is held for 6 beats, as scored)
  - The Bb is held for a beat too long before being released (so it is still being played during the first beat of the next bar)
- Selected embellishments added to melody:
  - A *turn* is added between the second C of bar 1 and the C at bar 2 beat 1
  - Two *grace notes* (Definition E.3) (Eb and F) are added before the first G in bar 3
  - An *appoggiatura* (Definition E.2) A is added before the G in bar 4
  - An *inverted mordent* (Definition E.4) is added before the A in bar 5
  - A *slide* of notes (Definition E.5) is played from the F at bar 6 beat 2 down to the Bb on beat 3
  - A *trill* is added to the last F, starting on bar 8 beat 1
- Selected tempo adjustments made whilst playing the melody:
  - The system was told that the soloist would be playing at 500 milliseconds per beat (ms/beat), and the soloist does start playing at that speed.
  - The soloist speeds up gradually and consistently during bars 1 to 4, reaching a final speed of 300 ms/beat at the start of bar 5

- At the start of bar 6 the soloist's tempo drops suddenly to 600 ms/beat (as if they had realised they were speeding up and overcompensating)
- On the third beat of bar 7, the soloist's tempo returns to the original speed of 500 ms/beat, and remains at that speed for the rest of the piece

For *Danse Macabre*, the following adaptations were made for testing purposes:

- Selected errors added to melody:
  - Skipping out the Bb at bar 3 beat 3 (such that this whole beat is missed out)
  - An extra Bb crotchet is added in between beats 2 and 3 of bar 5 (so this bar now has 4 beats instead of 3)
  - At bar 7, the notes at beats 2 and 3 (the Eb and the second Bb) are missed out. (This mistake is inspired by a mistake that was consistently made by one of the testers during the subjective evaluation).
  - A wrong note is played on the last quaver of bar 9 (the Eb is played as a C)
  - Bars 11 and 12 are repeated (so bar 11 is played, then bar 12, then bar 11, then bar 12)
  - The Eb on beat 1 of bar 13 is played as an E natural (then the second Eb, on beat 2 of this bar, is played correctly as an Eb)
  - An extra quaver is played in bar 14 (An Eb quaver is added in between the G on beat 1 and the F on beat 2)
  - Several notes are skipped in bars 15-16. The run of notes Eb G Eb F G from bar 15 beat 2 to bar 16 beat 1 are omitted entirely.
  - The last Eb in bar 16 (the second half of beat 3 of this bar) is played as an E natural.
  - The rest on beat 3 of bar 17 is replaced by the playing of a staccato crotchet D
- Selected embellishments added to melody:
  - A *slide* of notes is added between the second Bb in bar 3 (beat 3) and the first Eb in bar 4 (beat 1)
  - *Grace notes* Bb and C are added just before the D on beat 1 of bar 9
  - *Grace notes* Eb and F are added just before the Eb on beat 1 of bar 11



- 
- Selected tempo adjustments made whilst playing the melody:
  - The system was told that the soloist would be playing at 200 milliseconds per beat (ms/beat), and the soloist does start playing at that speed
  - At the start of bar 3, the soloist changes tempo so they are playing at 150 ms/beat
  - This tempo is maintained until bar 9, beat 2, where the soloist's tempo changes to 175 ms/beat
  - At bar 11, beat 1, the soloist's tempo slows to 200 ms/beat
  - At bar 13, beat 1, the soloist's tempo slows further to 300 ms/beat (these continual decreases in tempo are representative of a typical reaction to harder passages, which is to slow down for these harder parts of the piece)
  - At bar 15, the soloist's tempo picks up to 200 ms/beat (representing that they are getting more comfortable with playing this part of the piece)
  - The last bar is played at a slightly faster tempo of 150 ms/beat (representing how the soloist might rush the last bar as they have reached the end of the piece).

# Appendix E

## Glossary of terms used for musical ornamentations and embellishments

The following diagrams are taken from <http://www.gc-music.com/Ornament.htm> (except for the grace note and slide diagrams which are more modern interpretations of embellishment than is covered in this list).

In choosing which musical ornamentation to use during testing, I aimed to choose embellishments which I felt matched the style of the piece in each case (though I acknowledge that this is a subjective process).

For further details on different types of ornamentation and embellishments of notes, <http://www.gc-music.com/Ornament.htm> and [http://en.wikipedia.org/wiki/Ornament\\_\(music\)](http://en.wikipedia.org/wiki/Ornament_(music)) both give good descriptions and examples of common musical embellishments.



Figure E.1: *Acciaccatura*: a rapidly played note added just before the scored note, played on the beat



Figure E.2: *Appoggiatura*: A short note added just before the scored note, played on the beat and sharing half the duration of the scored note



Figure E.3: *Grace notes*: Quickly played note(s) added just before the scored note, played just before the beat



Figure E.4: *Inverted Mordent*: Two quick notes, the note above the scored note and then the scored note, played sequentially and added just after the scored note, played just after the beat



Figure E.5: *Slide*: a scalar run of notes moving gradually up or down, in quick succession



Figure E.6: *Trill*: a repeated playing of two notes one after another: the scored note and the pitch just above it



Figure E.7: *Turn*: a sequence of a few notes in a pattern of up-down-up movement, added between two scored notes and played off the beat