OUT OF SYNC: A SIMPLE SCORE BASED GUI FOR QUICK CORRECTION IN PERFORMER COMPUTER INTERACTION

John Chowning
CCRMA
Department of Music
Stanford University
Stanford, California 94305

ABSTRACT

A simple, but effective GUI allows a graphic and/or traditional score to serve as the basis for monitoring and updating performer/computer synchronicity. The GUI is passive (monitoring) until the two become out of sync at which time a single click in a desired region of the score resets the computer to be in sync with the performer.

1. INTRODUCTION

In this author's first composition project based upon computer-performer interaction (Voices for solo soprano\(^1\)), the computer accompaniment is triggered by the voice of the performer singing into a head microphone where the only intervention is a tap on the space bar to advance the computer when there is a rest in the vocal score—that was the goal. However …

Nothing has been as unnerving as the condition where the position in the score in the computer and performer fall out of sync—the computer fails to “trap” a pitch and falls behind or traps a pitch before its time and jumps ahead. The causes for error can be several, including a noisy environment, shift in placement of the head-microphone and less often, performer pitch error. Whatever the cause, the situation must be quickly corrected by moving the computer’s score position to the performer’s position, rather than the reverse as it is unlikely that a well-rehearsed performer has scrambled the order of pitches.

There is extensive experience in score following that has accumulated over the years from which one can gain insight and useful information in developing one’s own strategies [3]. Difficult problems involving tracking tempo, rhythm and “expressive pitch” have been attacked with results that have enriched the palette of possibilities available to us today and also revealed the extraordinary complexity of the seemingly simple act of musicians playing together. Results of Miller Puckette’s work in pitch detection of the singing voice [2] have in some form found their way into commonly used languages like pd and MaxMSP, of which this author has been a direct beneficiary in this project.

The score following strategy used in Voices is based upon pitch matching, but makes no attempt at automatic recovery when there is an error, rather relying on the intervention of the musician at the computer or the performer herself.\(^2\)

This paper is about an interface that is musically intuitive, one that allows a quick recoup from the most common errors and one that can get to any point in the score quickly.

2. TARGET PITCHES

The main database consists of lists of numbers that represent the “target pitches,” a subset of all of the pitches written in the vocal score. The target pitches are points at which synthesis or spatialization processes are initiated or a timing mark is set from which processes are initiated. The numbers are much like midi note numbers, but with two differences: the tuning division is non-standard– the same as that used in Stria [1]– and they are real numbers where the fractional part specifies, as a percent of the target pitch frequency, the width of the frequency band in which that frequency must fall to have a successful “trap” and advance to the next target pitch.

Having this “trap band” associated with every pitch has been extremely useful for several reasons: it allows for a large trap band for large interval jumps where the chance for pitch error is greatest, it allows for pitch inflection scaled to interval size, an important part of singing expressivity, and it allows for small trap bands where a proximate non-target pitch is small in interval and subject to confusion with the target pitch.

The non-target pitches in the database are only present for the purpose of rehearsal, where pitch cues can be given.

3. GRAPHICAL INTERFACE

The program, written in MaxMSP, was originally constructed so that a two or three digit number typed into a number box places the program’s score follower at the appropriate point in the database. The difficulty is in quickly translating a “sync problem” that is aurally detected, having both musical and numeric

---

1 *Voices* for soprano and interactive computer was commissioned by GRM and first performed March 12, 2005 in Paris, v.1. The second performance, after extensive additions, was performed in the Berkeley Symphony series on March 29, 2006, v. 2. Maureen Chowning, to whom the work is dedicated, sang both performances.

2 Voices was originally conceived as a piece sufficiently robust (simple!) that the soprano could perform it herself—we have not yet been so bold.
Figure 1. A screen shot of a page for Voices is shown in Figure 1c with annotations. To move the grey score marker to a new position, the mouse pointer is clicked at any point in one of the large rectangular CLICK AREAS. The program then detects its position on the 1000 unit x-axis and passes that number to a cascade of split objects set with the right x-position boundary, Figure 1b. The appropriate "slot" then releases the exact x-position, resets the marker and sends the subsection and target number to the main synthesis/processing program’s database. The score database, Figure 1a, is composed of the target pitch identifier, the x-position of the target pitch and the right boundary of the target pitch click area. Clicking the mouse pointer at either of the extremes of the window sets the marker to the last target pitch on the previous page or the first target pitch on the next page.
representation—one on a paper score the other as text/numbers on a computer screen—to a numeric “fix” involving typing numbers into a box. If too slow the correction is already behind, or worse, a typing error could place the score follower at some unintended position in the database.

A solution is to link the score directly to the database by making the paper pages of the score into a set of pictures stored in memory. A horizontal *hslider* object can be placed over the score whose knob position lines up with the target pitch. Better still, the *pictslder* object in MaxMSP provides for transparent overlays, one placed over the entire area of the score and a second, with a semi-transparent target pitch position marker placed “back” of the first, as shown in Figure 1c. In normal circumstances, the marker simply advances from target pitch to target pitch, as all of the soprano’s pitches are trapped and compared to the “next” target pitch. The pages are sequenced automatically. However, in the event of pitch trap error, a click on the screen anywhere within the click area of a target pitch on the x-axis and anywhere on the y-axis moves (jumps) the marker to that position in the database as shown in Figure 1c. How does this work?

### 3.1. Linking the GUI to the Database

Once the score, with the target pitches labeled, is a set of pictures in memory, a score database is created for the x-position of the target pitches on a 1000 unit x-dimension, as shown in Figure 1a. The positional data of each target pitch on each page is stored with the target pitch identifier (subsection number and its target pitch number as single number e.g. 1.3) and its right-boundary, half the distance to the next target pitch. This is quickly done with a score-editing patch, using the *pictslider* object that records the visually determined click position of the marker at each target pitch in sequence. The patch calculates the right-boundary at the mid-point position between the next target pitch position and then stores the three numbers in a list. The list is indexed by score page number in the score database, as shown in Figure 1a.

In the normal condition the data flow is from the main program database to the GUI. The marker indicating the next target pitch is advanced by a link to the main program managing the main database through the target pitch identifier. With each successful pitch trap, the identifier is updated and sent to the GUI. The score database is searched for the identifier and the associated x-position value advances the marker.

If (when) the performer and computer fall out of sync, the musician at the computer clicks *anywhere* close to the desired target pitch on the “front” *pictslider*—within the click area. This x position, N, is passed through a cascade of *split* objects that have been loaded with the right-boundary values for each of the target pitches on the page, as shown in Figure 1b. When N is ≤ to a stored right-boundary, the *split* object “bangs” a corresponding input in a *funnel* object that has been loaded with the x positions of the target pitches, Figure 1b. The x position of the target pitch is then sent to the “back” *pictslder*, the marker jumps to the new position and the target pitch identifier is sent to the main program, which resets the pointer in the main database. Because the control is at the parametric level rather than signal level, no clicks are introduced.

Clicking at either the rightmost or leftmost extremes of the score window changes the pages accordingly, with the pointer and target pitch identifier set at the last or first target pitch on the page respectively. The Max objects’ values are loaded with each new page.

### 4. CONCLUSION

This score based GUI is a familiar, contextual representation, giving both the performer and computer-musician an easily manipulated performance tracking platform. Changing a target pitch is done with a click on the screen anywhere close to the intended target pitch. Page turns are automatic, but can be forced, back or forward, with a click at either of the outside score boundaries.

The *pictslder* object could be used in a two dimensional space over a graphic score that would allow position/event based control rather than time-based.

### 4. REFERENCES

