Evaluating Singing for Computer Input Using Pitch, Interval, and Melody

Graeme Zinck[1], Daniel Vogel[1]
[1]School of Computer Science, University of Waterloo
{gzinck,dvogel}@uwaterloo.ca

ABSTRACT

In voice-based interfaces, non-verbal features represent a simple and underutilized design space for hands-free, language-agnostic interactions. We evaluate the performance of three fundamental types of voice-based musical interactions: pitch, interval, and melody. These interactions involve singing or humming a sequence of one or more notes. A 21-person study evaluates the feasibility and enjoyability of these interactions. The top performing participants were able to perform all interactions reasonably quickly (<5s) with average error rates between 1.3% and 8.6% after training. Others improved with training but still had error rates as high as 46% for pitch and melody interactions. The majority of participants found all tasks enjoyable. Based on our results, we propose design considerations for creating interfaces that utilize singing interactions to supplement standard computer input.

KEYWORDS

music, non-verbal vocal interactions

1 INTRODUCTION

With the proliferation of voice-activated devices, people are increasingly using their voice to perform hands-free interactions: as of 2018, 27% of the global online population used voice search on mobile devices [12]. Most of these systems use verbal interactions that translate spoken words and phrases into commands. On the other hand, non-verbal vocal interactions provide opportunities for language-independent, simple, continuous controls [26]. Instead of recognizing words, these techniques detect properties of vocal sound, such as vowel sounds or pitch, making them simple enough to process locally on consumer hardware in real time. Despite its simplicity, much less work has explored the non-verbal vocal input design space.

Past research has evaluated how we can use various vocal features, such as blowing puffs of air [4, 28], hissing [19], producing vowel sounds [6, 8], and singing or humming pitches. Within this last category, past work has used pitch for continuous control by sliding pitch up and down [14, 24] or discrete control by assigning portions of the frequency space to different functionality [3, 18, 20, 22]. In these systems, users can either hum or produce sustained sounds like "la" and "do." However, most work does not leverage music theory to expand these interactions, and those that do have not performed formal evaluations.

We leverage music theory to design interactions that make computer use more musical and potentially more enjoyable. Our techniques are versatile: by using pitches from the major scale, a larger number of distinct interaction variations are possible. Furthermore, frequently using these interactions could help users improve their vocal ability. We envision using singing interactions as a way to replace or supplement current interface methods for functions like command shortcuts, mode switching, and parameter control by singing melodies from popular music. However, before we can build such applications, we need to first evaluate the feasibility of fundamental types of singing interactions.

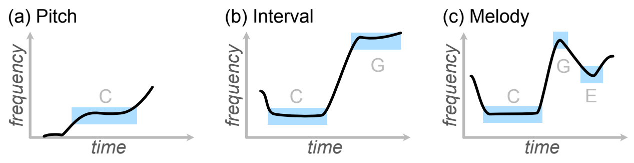


Figure 1: Examples of fundamental singing interaction techniques, visualized as vocal frequency over time: (a) pitch is a single note; (b) interval is a sequence of two notes; and (c) melody is a sequence of three notes.

We explore pitch, interval, and melody interactions, corresponding to sequences of one, two, and three distinct musical notes, respectively (Fig. 1). A 21-person study evaluates their feasibility and enjoyability for both singers ($n = 10$) and non-singers ($n = 11$). For each technique, participants were prompted to sing or hum 7 possible sequences of notes, both with and without background music. The findings demonstrate that top performers are able to perform all interaction types sufficiently fast (3.5–5s) with low error rates (<10%) after training. The lowest performers had comparable speeds (4.5–5.5s), but error rates as high as 46% after training. Of the three techniques, pitch interactions are the easiest to learn and interval interactions are the hardest to master. Participants found background music helpful in all tasks, despite having a limited effect on measured performance. Overall, the majority of participants found all interactions enjoyable.

Our work makes three contributions: (1) experimental results validating the effectiveness of pitch, interval, and melody interactions as a form of computer input; (2) an evaluation of how background music impacts performance; and (3) a set of design considerations to leverage these interactions.

2 BACKGROUND AND RELATED WORK

Pitch, interval, and melody interactions are types of non-verbal vocal interactions: they facilitate vocal control without the use of words. We provide an overview of non-verbal vocal input mechanisms, with a focus on those incorporating vocal pitch.

2.1 Non-Verbal Vocal Interactions

The simplest non-verbal vocal interactions are binary in nature. For example, with Pufftext [4], users blow into a microphone to select characters on a hands-free spinning keyboard for mobile phones. Similarly, with Blowclick [28], users blow into a microphone to make selections with low latency. Poláček et al. [19] leverage hissing to select characters on a virtual keyboard. These on/off interactions enable fast, precise, hands-free control, but they are limited: the only degree of freedom is the duration of the interaction.

Other techniques have more degrees of freedom by interpreting a wider range of vocal sounds. In particular, some systems map vowel sounds like "a," "e," "i," "o," "u," and their intermediaries to a single continuous parameter. For example, Vocal Joystick [5] is a voice-controlled mouse that moves based on the shape of a vowel, as classified by a multi-layer perceptron. A small Fitts' Law study with four expert users showed movement time was comparable to using a joystick, but much slower than using a mouse. The same strategy has been applied to drawing applications with VoiceDraw [7] and video games with Harada et al.'s Pacman [8]. Unlike our work, these techniques attempt to replace traditional inputs with more accessible non-verbal vocal inputs.

Some work has considered how non-verbal commands can work in tandem with other interactions for general computing. Igarashi and Hughes [11] use verbal commands followed by vowel sounds to support more precise control of various parameters. Sakamoto et al. [21] augment touch input with vowel sounds to allow continuous control of scrolling and zooming. VoicePen [6] augments pen input with vowel sounds to configure drawing parameters in an enjoyable manner. In a similar manner, we envision our singing interactions as augmenting standard user input.

2.2 Pitch for Continuous Control

We are interested in non-verbal vocal interactions that leverage pitch. While these typically still involve producing vowel sounds, the interaction is independent of vowel used. Instead, the fundamental frequency of the voice determines the interaction. Such interactions can be mapped to a single continuous parameter, in which users can sing any pitch to continuously adjust the parameter, or multiple discrete parameters, in which the sound frequency space is partitioned into ranges that trigger different functionality. Unlike the detection of vowel sounds, continuous pitch-based interactions intuitively map to relative movement control by singing higher or lower. Voodle [14] uses pitch and rhythm to control a one-dimensional robot, and VoiceBot [9] uses pitch and vowels to control a robotic arm. Similarly, Sporka et al. [24] use pitch to determine movement direction of a mouse pointer. Peixoto et al. [17] use pitch for smooth control of wheelchair speed. These works have evaluated application-specific task performance, but the results are not generalizable. Furthermore, in all cases, pitch is only mapped to a single parameter. Our work discretizes the spectrum of pitches to facilitate a wider variety of functionality.

2.3 Pitch for Discrete Control

Instead of directly using the continuous frequency space, discrete pitch-based interactions divide it into two or more ranges that map to different parameters. Some techniques use a single threshold pitch to determine when the voice is above, below, or crossing the threshold. Chanjaradwichai et al. [3] detect pitches above and below a threshold to select one of four cells in a grid. A small 6-person study indicated that short pitch-based vocalizations were faster than using speech commands and had error rates around 12%. Poláček and Míkovec [18] use hummed commands above and below a threshold for mouse clicks, finding the approach faster but more error-prone than speech commands (6% vs. 3.5%). This approach has also been adapted to the more complex context of text entry. Humsher [20] detects frequencies above and below a threshold to select characters for text entry and CHANTI [23] uses similar interactions with an ambiguous keyboard, where each keyboard key is associated with multiple possible letters. In all cases, a single threshold pitch still restricts control to a small number of parameters.

Other work has divided the frequency space with multiple threshold pitches, enabling control of more parameters. For instance, Sporka et al. [25] distinguish between four pitches for keyboard input, where each pitch sung divides the number of possible letters by 4 until only one possibility remains. More related is the work of Hämäläinen et al. [10], which uses many threshold pitches for music education video games. The pitch thresholds align with the twelve semitones of Western music, making this interaction technique more musical. However, their system is designed for showing visual pitch feedback when singing songs instead of providing an interaction mechanism for general use. They do not perform any formal evaluation of their technique.

Sporka [22] evaluates how effectively people can sing discrete pitches in general. The author compares performance when there are different sized ranges of pitches to sing, using ranges two, four, and eight times larger than a baseline size of one semitone. The findings demonstrate that non-singers have a lower error rate when the range is four times larger (23%) compared to the baseline (52%). Our work differs in four ways. First, our method of pitch recognition is different: instead of determining pitch based on the last frequency detected, which might slide off the desired pitch, we determine it by holding a pitch for 200ms. Second, we consider sequences of one, two, and three notes in our evaluation instead of just one. Third, we evaluate in what ways background music can affect performance, which was not considered in any past work. Finally, we design musically appealing interactions that are harmonious with background music rather than using arbitrary pitch targets.

3 TECHNIQUE

Our work evaluates the performance of pitch, interval, and melody singing interactions, where users can hum or sing on any vowel, such as "la," "tee," or "do." Before explaining the musical characteristics and system recognition method of these interactions, we first provide background on musical terminology we use.

3.1 Musical Terminology and Notation

Our approach to pitch-based interactions leverages concepts from music theory.

The *pitch* of a sound is the fundamental frequency, $f_0 \in \mathbb{R}^+$, of its waveform. A *note* is a sound that has some associated pitch. Following the Musical Instrument Digital Interface specification [1], each pitch f_0 is assigned an integer *note number*, $p = 69 + 12 \times \log_2(\frac{f_0}{440})$. For reference, A4 concert pitch has $f_0 = 440$ with note number 69. For a pitch with note number p, a *semitone* is the difference between p and $p+1$ and an *octave* is the difference between p and $p+12$.

A *scale* is an ordered set of pitches, independent of the octave. For our purposes, we represent a scale as an ordered pair $(t, \mathbf{s})$, where $t \in \mathbb{N}, 0 \leq t < 12$ is the *tonic*, or first pitch in the scale, and $\mathbf{s}$ is a vector with pitches in the scale relative to t. The vector $\mathbf{s}$ is defined such that $s_0 = 0$ and $s_{i-1} < s_i < 12$. For a given scale $(t, \mathbf{s})$, the *degree* of a pitch with note number p is d, where $1 \leq d \leq |\mathbf{s}|$ and $s_{d-1} = p - t \pmod{12}$.

Our work uses the major scale, $(t, \mathbf{s})$, for any tonic t and $\mathbf{s} = (0\ 2\ 4\ 5\ 7\ 9\ 11)$. This is because a large portion of Western music is based on this scale, making the pitches more pleasing and familiar for the average person. Because we use the major scale, there are effectively only $|\mathbf{s}| = 7$ distinct pitches that are used in our interactions.

When we discuss interval and melody interactions, we sometimes use the pitch one octave above the first pitch sung, which is always the tonic in our case. We abuse the standard definition of degree to refer to the pitch one octave above the first pitch sung as degree $d = 8$.

We assume users have a vocal range of one octave, which is reasonable for both singers and non-singers [22]. However, different people have different ranges: some can only sing high pitches while others can only sing low pitches. To accommodate different vocal ranges, we determine the appropriate tonic t for a given person.

For the remainder of this paper, whenever we refer to a note or pitch, we are referring to its degree as opposed to its note number.

3.2 Single Pitch Interactions

The simplest type of singing interaction we consider involves a single note. The system detects pitch by determining the f_0 of microphone input every 50ms using a fast Fourier transform with size 2048 and processing it using the McLeod Pitch Method [15]. As suggested by the method, if a frequency is below a minimum clarity (95%, in our case), it is ignored to reduce noise. Each time a frequency is detected and not ignored, we increment the current *time step* by 1. The frequency at the i-th time step is f_i. Note that time steps are at least 50ms apart, but gaps may be longer since unclear frequencies are ignored.

To ensure only intentional interactions are recognized, a user must sing a note for at least 4 consecutive time steps ($\geq$ 200ms). If the user sings a different note and holds for 4 time steps, the new pitch is recognized and the old one is discarded. The recognized pitch is confirmed after a silence of 500ms. This design allows users to correct their pitch during an interaction: they can start by singing one pitch, then "slide" into another based on visual feedback. This is crucial for usability since both musicians and non-musicians can have difficulty singing pitches precisely [16]. Our design also allows users to slide off pitch at the end of an interaction without penalty, unlike the pitch recognizer used by Sporka [22].

To compensate for variable audio quality, we smooth the detected frequencies using a recursive definition: $f_1^s = f_1$ and $f_i^s = \frac{f_i}{2} + \frac{f_{i-1}^s}{2}$. This can add an additional 50–100ms latency, but it improved performance in our pilot tests. Also, poor audio quality can sometimes result in large, sudden changes of pitch. To resolve this problem, if a frequency f_i is more than an octave away from f_{i-1}, it is ignored unless sustained for 3 time steps.

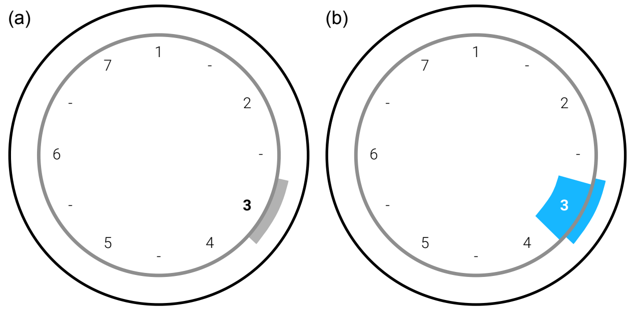


Figure 2: Our pitch visualization is a circle divided into 12 semitones, with the 7 degrees of the major scale labelled as 1 to 7: (a) the gray arc indicates the frequency detected and the bold "3" indicates degree 3 is detected but not yet recognized, and (b) the blue outer arc and thick inner arc indicate degree 3 is recognized.

Since we assume users have a one-octave range, our system is restricted to 12 possible semitones in Western music. However, since we are especially interested in interactions that sound harmonious with background music, we only evaluate interactions that use the 7 degrees in the major scale. Thus, there are $n = 7$ single pitch interactions.

An important part of our technique is a circular visualization for helping users find their desired pitch (Fig. 2). The visualization shows the currently detected pitch with an arc on the outside of the circle. Once a pitch is recognized, the outer arc turns blue and a thick blue arc appears inside the circle indicating which pitch is recognized. This also indicates when the user can stop singing. The visualization is easily extensible to interval and melody interactions by making the inner arc span across all of the recognized pitches. This contrasts with the visualization from Sporka [22] which uses a 1D horizontal line of frequencies and a vertical line indicating current pitch.

3.3 Interval Interactions

Interval interactions entail singing two notes in sequence. Users sing one note for at least 4 time steps ($\geq$ 200ms), change to another note for at least 3 time steps ($\geq$ 150ms), and fall silent for 500ms. To keep this interaction easy to perform during our study, we only consider intervals starting at the tonic t and ascending to an end pitch with degree $1 < d_e \leq 8$.

In total, we support $n = 7$ interval interactions. In the future, this approach is easily extensible to have $7 \times 7 = 49$ possible interval interactions within a one-octave range, with 7 possible start degrees $1 \leq d_s \leq 7$, and 7 possible end degrees $1 \leq d_e \leq 8, d_s \neq d_e$. Note that $d_s \neq 8$ by our definition of degree 8.

The interval visualization is identical to pitch, except once the second note is recognized, the inside arc spans from the first note d_s to the second note d_e.

3.4 Melody Interactions

Melody interactions go one step further with three or more notes in sequence. Users first sing a note for at least 4 time steps ($\geq$ 200ms), then sing at least two more notes for any duration, subsequently falling silent for 500 ms. Inspired Lin et al.'s system for evaluating people's ability to accurately sing songs [13], we use dynamic time warping (DTW) to determine the cost of warping a sequence to match each of the possible melodies. The melody with the lowest cost at the end of the interaction is the recognized melody. This makes it possible to have longer melodies without holding onto pitches for 150ms each, unlike our interval interaction technique. It also means a user does not need to accurately sing the second and third notes. As long as they briefly get close to the appropriate pitch, DTW can typically determine the intended melody.

For simplicity and tractability, we use $n = 7$ three-note melodies (Table 1), all of which start at the tonic note, use only notes in the major scale, and stay within a one-octave vocal range. The melodies are designed to be relatively easy to sing by typically focusing on degrees 1, 3, and 5 (the degrees in a major chord) or moving by one degree at a time. The number of melodies is theoretically unbounded: they can have different start notes and unbounded length. In practice, this is limited by the number that we can accurately and quickly distinguish using DTW.

Table 1: Melodies evaluated by our experiment.

Melodies (sequences of notes)			
1 – 3 – 2	1 – 5 – 1	1 – 5 – 3	1 – 5 – 8
1 – 2 – 3	1 – 4 – 1	1 – 3 – 5	

Our visualization for melodies is similar to that of intervals. The difference is that there are two arcs inside the circle showing the melody detected: one from the first to the second note and one from the second to the third note in the melody.

All three types of singing interactions could be used in a single interface. For instance, one could sing the interval 1−5, then melody 3−5−3, followed by pitch 4. The requirement for these combinations is that each starting pitch d_s can only be mapped to one type of interaction. In this example, 1 is mapped to interval interactions, 3 is mapped to melody, and 4 is mapped to pitch. Our exploration focuses on each technique individually, but an application could incorporate all of them.

3.5 Using Background Music

A novel part of our interaction techniques is how they use background music. We hypothesize that appropriate background music will make it easier for users to find their desired notes. Playing music with a major scale that emphasizes the tonic note should make it easier for users to sing relative to that note. Additionally, singing along with music has the potential to increase enjoyability. For our experiments, we play a single 30 second loop of solo piano music, transposed into a specific user's vocal range using their tonic t.

4 EXPERIMENTAL DESIGN

For each interaction technique (pitch, interval, and melody), we performed an experiment to evaluate its feasibility and enjoyability. Because our objective is to evaluate the effectiveness of these interactions in general, we ordered the three experiments based on the complexity of the tasks, moving from singing one note to eventually singing sequences of three. In this section, we explain the shared experimental design. An accompanying video also demonstrates the different experiment tasks.

4.1 Participants

We recruited 25 participants, of which 4 were excluded due to technical issues with their audio setup. These 4 excluded participants self-reported technical problems and we manually verified logs and audio recordings to confirm. The 21 included participants averaged 25.8 years of age ([23.6, 31.8], 95% confidence interval), of which 8 were female, 11 were male, and 2 were non-binary. Participants were recruited using email, social media, and an HCI course discussion board, receiving \$20 for successful completion of the study. No participants had any experience with our system prior to the experiments.

When asked to rate their agreement with the statement "I consider myself a good singer," 10 participants responded with "Agree" or "Strongly agree" and 11 participants responded with "Slightly agree" or lower. We used this as a guide to recruit participants with different musical ability. For analysis, we cluster them into skill levels based on their error rates.

4.2 Apparatus

The study was conducted online through a React web application.[1] All participants were required to use Google Chrome on a laptop or desktop with a microphone and headphones. Participants were instructed to perform the tasks in a quiet room to ensure accuracy of the pitch detection. Ambient noise was recorded and analysed periodically throughout the experiment to ensure the levels were sufficiently low. Participants completed an audio test using a custom phone web app that played notes to ensure their main device could recognize the correct pitches.

4.3 Tasks

Each of the three experiments prompted participants to perform tasks related to the interaction techniques explained in Section 3. Participants pressed a button to start the first trial in each block. Then, they were shown an interface with the interaction visualization at the bottom and the target above with the notes to sing (Fig. 3). Singing for at least 200ms and falling silent for 500ms ended the trial. During training, a green check mark or red x-mark appeared after each trial to indicate success or failure.

4.4 Procedure

After participants answered a series of questions on demographics, musical experience, and technology experience, they performed a calibration procedure to determine their vocal range and the tonic t for their interactions.

To determine vocal range, after watching a short tutorial video, participants were asked to sing one low note and one high note. Then, they were shown a slider to fine-tune their detected vocal

[1]Experiment source code is available at https://github.com/gzinck/singui

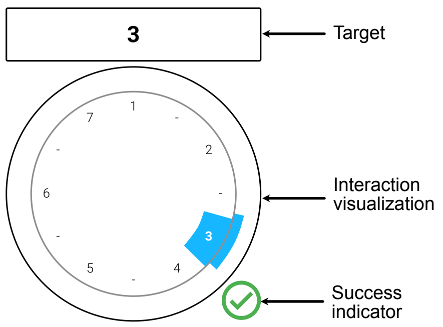


Figure 3: Illustration of the experiment task interface, showing an example for pitch tasks with target 3 while a participant is singing note 3.

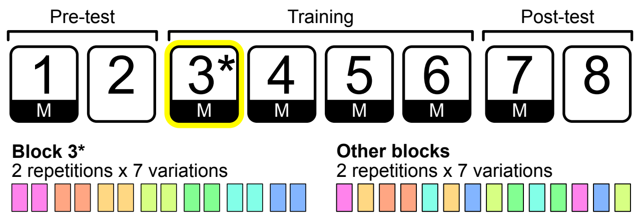


Figure 4: Example of the block structure for a participant. M indicates there is background music for the block. Colours indicate different task variations.

range. A green area above the slider indicated what the ideal range should be and deviating from the expected range caused a warning to appear. However, participants maintained full control over their selection, which was deemed important through pilot tests. Following this, the three experiments began.

For each technique, participants performed an experiment with three stages: pre-test (blocks 1–2), training (blocks 3–6), and post-test (blocks 7–8) (Fig. 4). The pre-test and post-test measured performance with and without background music. The training blocks allowed two attempts per task and used audio prompts to indicate the notes to sing. Block 3 played audio prompts before every trial, whereas blocks 4–6 played audio prompts only after failing the first attempt. All blocks had a random ordering of task variations except block 3, which had each variation twice in a row.

4.5 Design

Each experiment (pitch, interval, and melody) has a within-subjects design with two primary independent variables: TRAINING with 2 levels (UNTRAINED, TRAINED) and BACKGROUND with 2 levels (MUSIC, NO-MUSIC). For analysis, we classify participants based on their error rates across the three experiments. This classification is a between-subjects variable: SKILL with 3 levels (NOVICE, INTERMEDIATE, EXPERT).

There were 7 task variations for each technique, representing the pitch, interval, or melody to sing. Each block had 2 repetitions × 7 variations. In each of the pre- and post-test stages, there was 1 block × 2 BACKGROUNDS, where BACKGROUND conditions were counterbalanced. In the training stage, all 4 blocks had MUSIC. The training blocks are solely for training purposes, so our analysis relies solely on the data from blocks in the two test stages.

The primary measures computed from logs are *Total Time*, *Sing Time*, and *Error Rate*. *Total Time* is the time from the end of the previous trial until an interaction is completed. *Sing Time* is the time from the first detected pitch until the last detected pitch in a trial. *Error Rate* is the proportion of trials in a block that ended with an error.

In addition, a questionnaire after each interaction experiment provides 6 NASA-TLX metrics and 2 subjective measures (enjoyability and perception of background music). All measures are rated on a 7-point numeric scale for consistency.

In summary: 2 BLOCKS × 2 BACKGROUNDS × 7 variations × 2 repetitions = 56 data points per participant, per experiment.

5 RESULTS

In order to meaningfully explore the variation in objectively measured performance between participants, we used k-means to cluster participants into three SKILL levels based on their error rate across all three experiments. Resulting SKILL levels include EXPERT (8.8% error rate, $n = 8$), INTERMEDIATE (30.2% error rate, $n = 6$), and NOVICE (54.6% error rate, $n = 7$). Applying the non-parametric Spearman correlation method, we found that SKILL was correlated with years of private music lessons ($r_s = .48$, $p = .029$) and years of group music lessons ($r_s = .45$, $p = .043$), but interestingly, not years of music ear training ($p = .098$), years of vocal training ($p = .298$), participant perception of singing ability ($p = .195$), or frequency of day-to-day informal singing ($p = .19$). This indicates that while musical experience predicts performance to an extent, singing experience is not required or sufficient to perform well.

In the analysis to follow, a SKILL × TRAINING × BACKGROUND ANOVA with Tukey HSD post hoc tests was used, unless noted otherwise. When the assumption of sphericity was violated, degrees of freedom were corrected using Greenhouse-Geisser ($\epsilon < 0.75$) or Huynh-Feldt ($\epsilon \geq 0.75$). According to the Shapiro-Wilk Normality test, none of the residuals for collected data were normally distributed, so Box-Cox or ART-transformed [27] values were used for statistical analysis. For each measure, trials were aggregated by participant and factors being analysed.

5.1 Pitch Interactions

For every participant ($n = 21$), trial times more than 3 standard deviations from the mean time were excluded as outliers. In total, 15 trials (1.3%) were removed. Of the remaining trials, a minor error in client-side code resulted in 8 false positives and 2 false negatives that were corrected based on the system logs, representing 0.9% of the included trials.

5.1.1 Total Time. Time was consistent across most conditions. There were no main effects involving SKILL ($p = .32$), BACKGROUND ($p = .10$), or TRAINING ($p = .25$) on *Total Time* (Fig. 5).

We found that background music increased *Total Time* by 15% for novices, whereas it had an insignificant effect on other participants. There was an interaction between BACKGROUND and SKILL on boxcox-transformed *Total Time* ($F_{2,18} = 3.91$, $p = .039$, $\eta^2_G = .03$). Post hoc tests show, for the NOVICE participants, MUSIC (5129ms) was slower than NO-MUSIC (4447ms, $p < .0001$). While the standardized

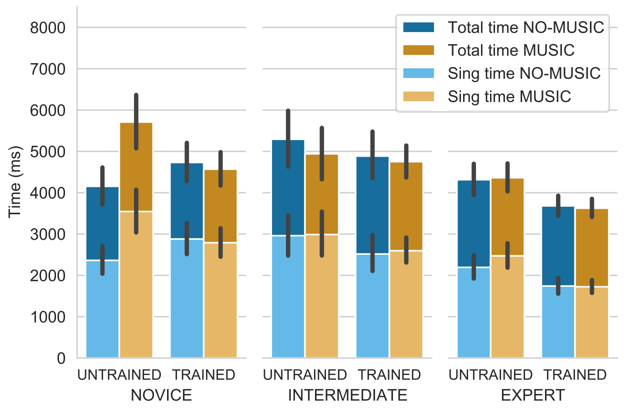


Figure 5: Pitch task *Total Time* and *Sing Time* by SKILL for each BACKGROUND (error bars in all graphs are 95% confidence).

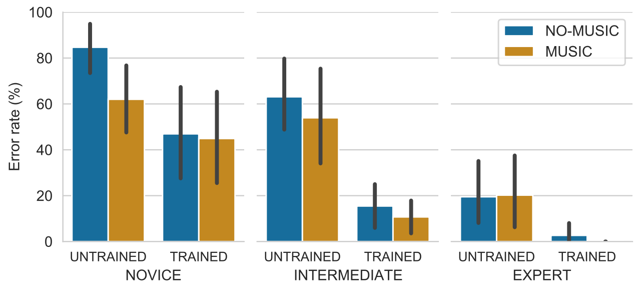


Figure 6: Pitch task *Error Rate* by SKILL and TRAINING for each BACKGROUND.

effect size of .03 is considered small [2], it is intriguing that music increases time.

5.1.2 Sing Time. Background music increased *Sing Time* by 241ms (Fig. 5). There was a main effect of BACKGROUND on on boxcox-transformed *Sing Time* ($F_{1,18} = 2.21$, $p < .007$, $\eta^2_G = .03$), where MUSIC (2648ms) was slower than NO-MUSIC (2407ms). This represents a 10% increase in time.

There was no interaction between BACKGROUND and SKILL for *Sing Time* ($p = .47$).

5.1.3 Error Rate. We found training decreased *Error Rate* by 29.1% (Fig. 6). There was a main effect of TRAINING on *Error Rate* ($F_{1,18} = 43.52$, $p < .0001$, $\eta^2_G = .71$), where TRAINED (19.6%) was lower than UNTRAINED (48.7%). The large .71 standardized effect size of TRAINING on *Error Rate* shows that people can improve, even if they perform poorly initially.

Unsurprisingly, skill affected *Error Rate* because we classified participant SKILL using this metric. There was a main effect of SKILL on ART-transformed *Error Rate* ($F_{2,18} = 43.73$, $p < .0001$, $\eta^2_G = .83$). Post hoc tests show how EXPERT *Error Rate* (10.6%) was lower than INTERMEDIATE (35.8%), which was lower than NOVICE (59.6%) (all $p < 0.01$). This represents a decrease in error rate of 49.0% for the strongest participants, and the standardized effect size of .83 is considered large [2].

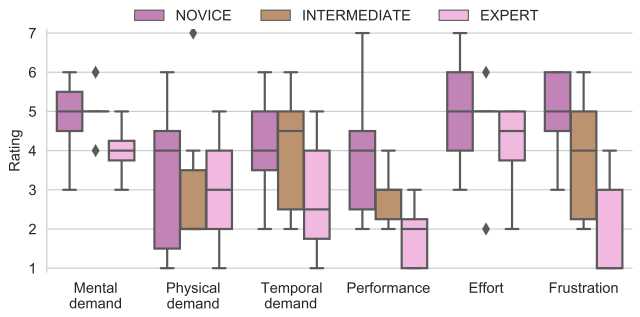


Figure 7: NASA-TLX ratings for pitch tasks. Lower values correspond to lower mental, physical, and temporal demand, as well as greater performance, lower effort, and lower frustration.

Training had a larger effect on *Error Rate* for intermediates than others. There was an interaction between TRAINING and SKILL on *Error Rate* ($F_{2,18} = 3.74$, $p = .044$, $\eta^2_G = .29$). The decrease in *Error Rate* after training was larger for INTERMEDIATES (−45%) compared to EXPERTS (−19%, $p = .014$). After training, EXPERTS had a very low error rate of 1.3%. Considering the large standardized effect size of .50, intermediates have much more to gain from training than the others, while experts are able to nearly perfect the technique.

5.1.4 NASA-TLX. Since measures were not normally distributed, all were analysed using Wilcoxon signed-rank tests with Holm-Bonferonni corrections. Experts had lower task load metrics than the other groups (Fig. 7). EXPERTS (4.0) perceived a lower *Mental Demand* than INTERMEDIATES (5.0, $p = .032$), but there were no significant differences between other groups. EXPERTS (1.9) also perceived better *Performance* than NOVICES (3.9, $p < .020$). Furthermore, EXPERTS (1.9) perceived lower *Frustration* than both INTERMEDIATES (3.8, $p = .039$) and NOVICES (5.0, $p < .003$). There were no significant differences between ratings based on SKILL for *Physical Demand*, *Temporal Demand*, or *Effort*.

5.1.5 Subjective Ratings. Overall, 71% of participants perceived that background music made the tasks easier. When asked how background music affected difficulty, 15 rated it slightly easier or better, 4 rated it slightly harder or worse, and 2 were neutral.

The majority of participants (62%) found pitch tasks enjoyable. 13 found the tasks slightly enjoyable or better, 5 found it slightly unenjoyable or worse, and 3 were neutral.

5.2 Interval Interactions

For every participant ($n = 21$), outliers were removed (16 trials, 1.4%) and false positives (3) and false negatives (16) were corrected (1.6% of included trials).

5.2.1 Total Time. None of the studied factors impacted *Total Time* for interval interactions (Fig. 8).

5.2.2 Sing Time. Similarly, none of the studied factors impacted *Sing Time* (Fig. 8).

5.2.3 Error Rate. We found training decreased *Error Rate* by 15.7% (Fig. 9). There was a main effect of TRAINING on *Error Rate* ($F_{1,18} = 9.28$,

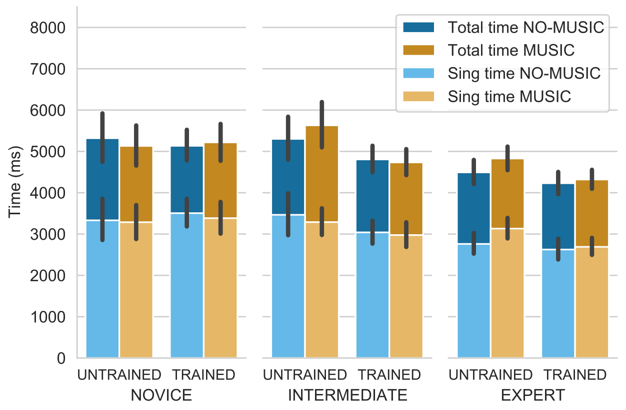


Figure 8: Interval task *Total Time* and *Sing Time* by TRAINING and SKILL for each BACKGROUND.

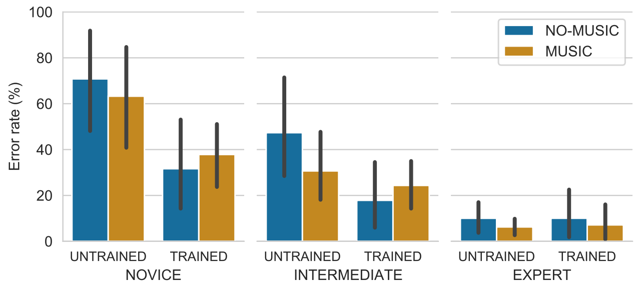


Figure 9: Interval task *Error Rate* by SKILL and TRAINING for each BACKGROUND.

$p < .007$, $\eta^2_G = .34$), where TRAINED (20.9%) was lower than UNTRAINED (36.6%).

As before, skill affected *Error Rate* due to our clustering. There was a main effect of SKILL on ART-transformed *Error Rate* ($F_{2,18} = 22.33$, $p < .0001$, $\eta^2_G = .71$). Post hoc tests show how EXPERT error rate (8.3%) was lower than both INTERMEDIATE (30.1%) and NOVICE (50.9%) rates (all $p < 0.002$). This represents a decrease in *Error Rate* of 42.6% for experts and the standardized effect size of .71 is considered large [2].

Interestingly, while training improved the *Error Rate* for most participants, it did not for experts. There was an interaction between TRAINING and SKILL on *Error Rate* ($F_{2,18} = 3.94$, $p = .038$, $\eta^2_G = .30$). Post hoc tests show the decrease in *Error Rate* after training for NOVICES (−32.3%) was much larger than for EXPERTS (+0.5%, $p = .014$). The large .30 standardized effect size shows that NOVICES improve much more than others.

While background music did not directly affect *Error Rate*, training improved error rates more when there was no background music present. There was an interaction between SKILL and BACKGROUND on *Error Rate* ($F_{2,18} = 6.31$, $p = .022$, $\eta^2_G = .26$). Post hoc tests show TRAINING reduced *Error Rate* for NO-MUSIC (−21.4%) more than for MUSIC (−9.9%) ($p < 0.007$). Since the TRAINED *Error Rates* are comparable for both conditions, with 19.5% for NO-MUSIC and 22.3% for MUSIC, this suggests training is needed most when there is no music.

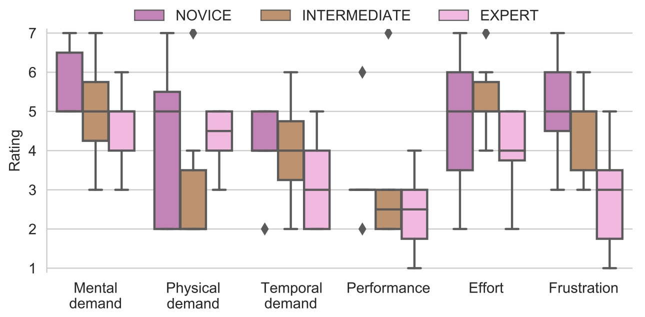


Figure 10: NASA-TLX ratings for interval tasks. Lower values correspond to lower mental, physical, and temporal demand, as well as greater performance, lower effort, and lower frustration.

5.2.4 NASA-TLX. Again, experts had lower task load metrics than other groups (Fig. 10). EXPERTS (4.3) perceived lower *Effort* than INTERMEDIATES (5.7, $p = .049$). EXPERTS (2.9) also perceived lower *Frustration* than NOVICES (5.1, $p = .018$). There were no significant differences between ratings based on SKILL for *Mental Demand*, *Physical Demand*, *Temporal Demand*, or *Performance*.

5.2.5 Subjective Ratings. Overall, 67% of participants perceived that background music made the tasks easier. When asked how background music affected difficulty, 14 rated it slightly easier or better, 4 rated it slightly harder or worse, and 3 were neutral.

As with pitch tasks, the majority of participants (57%) found the interval tasks enjoyable. 12 found the tasks slightly enjoyable or better, 1 found it slightly unenjoyable or worse, and 8 were neutral.

5.3 Melody Interactions

For every participant ($n = 21$), outliers were removed (15 trials, 1.3%) and false positives (8) and false negatives (10) were corrected (1.6% of included trials).

5.3.1 Total Time. Neither skill nor background music directly affected *Total Time* for melody interactions (Fig. 11). However, unlike the other tasks, training reduced *Total Time* by 607ms. There was a main effect of TRAINING on boxcox-transformed *Total Time* ($F_{2,18} = 14.36$, $p < .001$, $\eta^2_G = .07$), where TRAINED (5073ms) was faster than UNTRAINED (5680ms). This represents a 11% decrease in *Total Time*.

We found that training impacted novices differently than experts when music was present. There was an interaction between SKILL, BACKGROUND, and TRAINING on *Total Time* ($F_{2,18} = 3.95$, $p = .038$, $\eta^2_G = .03$). Post hoc tests show that with MUSIC, training increased *Total Time* for NOVICES (+211ms) and decreased *Total Time* for EXPERTS (−1076ms) (all $p < .002$). This might indicate that novices found music distracting for these complex tasks.

5.3.2 Sing Time. We found that training improved *Sing Time* by 406ms (Fig. 11). There was a main effect of TRAINING on boxcox-transformed *Sing Time* ($F_{1,18} = 7.57$, $p = .013$, $\eta^2_G = .04$), where TRAINED (3456ms) was faster than UNTRAINED (3862ms). This represents a 11% decrease in *Sing Time*.

Again, there was an interaction between SKILL, BACKGROUND, and TRAINING on *Sing Time* ($F_{2,18} = 4.67$, $p = .023$, $\eta^2_G = .03$). Post

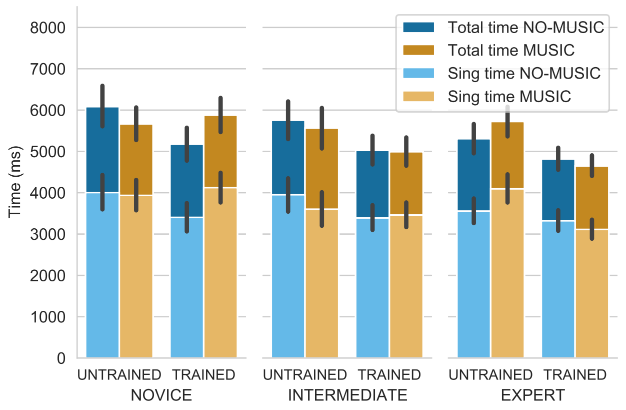


Figure 11: Melody task *Total Time* and *Sing Time* by TRAINING and SKILL for each BACKGROUND.

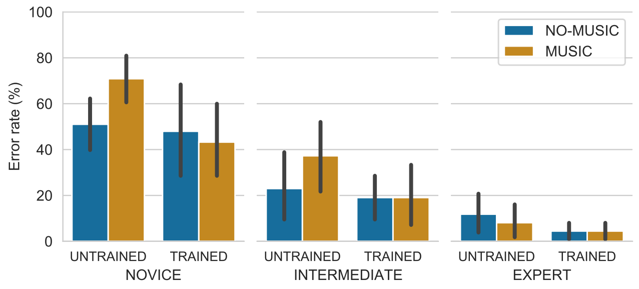


Figure 12: Melody task *Error Rate* by SKILL and TRAINING for each BACKGROUND.

hoc tests show that with MUSIC, training increased *Sing Time* for NOVICES (+189ms) and decreased *Sing Time* for EXPERTS (−982ms) (all $p < .002$).

5.3.3 Error Rate. We found training decreased *Error Rate* by 10.4% (Fig. 12). There was a main effect of TRAINING on *Error Rate* ($F_{1,18} = 4.82$, $p = .041$, $\eta^2_G = .21$), with TRAINED (22.3%) lower than UNTRAINED (32.7%). Thus, training improved both time and error rate for melody tasks.

As before, skill affected *Error Rate* due to our clustering. There was a main effect of SKILL on ART-transformed *Error Rate* ($F_{2,18} = 54.77$, $p < .0001$, $\eta^2_G = .86$). Post hoc tests show how EXPERT *Error Rate* (7.2%) was lower than INTERMEDIATE (24.6%), which was lower than NOVICE (53.3%) (all $p < 0.001$). This represents a decrease in error rate of 46.1% for the highest performing participants and the standardized effect size of .86 is considered large [2].

5.3.4 NASA-TLX. Again, experts had lower task load metrics than other groups (Fig. 13). EXPERTS (2.5) perceived a lower *Temporal Demand* than INTERMEDIATES (5.1, $p = 0.016$). EXPERTS (2.1) also perceived better *Performance* than NOVICES (4.1, $p = 0.015$). There were no significant differences between ratings based on SKILL for *Mental Demand*, *Physical Demand*, *Effort*, or *Frustration*.

5.3.5 Subjective Ratings. Overall, 57% of participants perceived that background music made the tasks easier. When asked how background music affected difficulty, 12 rated it slightly easier or better, 4 rated it slightly harder or worse, and 5 were neutral.

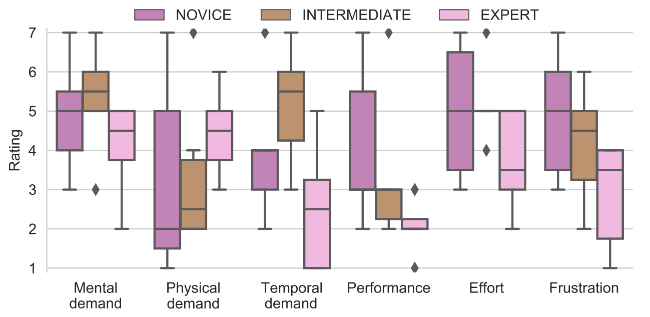


Figure 13: NASA-TLX ratings for melody tasks. Lower values correspond to lower mental, physical, and temporal demand, as well as greater performance, lower effort, and lower frustration.

Similar to the other tasks, 62% of participants found the tasks enjoyable. 13 found the tasks slightly enjoyable or better, 3 found it slightly unenjoyable or worse, and 5 were neutral.

6 DISCUSSION

The main objective for our experiments was to explore the feasibility and enjoyability of singing interactions. In this section, we compare results for the three types of interactions and discuss the design considerations emerging from our research.

6.1 Feasibility

In terms of feasibility, the main results are consistent across all types of interactions. For participants classified as experts, after training in all experiments, average time per trial was under 5s with error rates under 10%. For pitch tasks in particular, experts had a very low error rate of 1.3% after training. On the other hand, those we classified as novices performed poorly even after training, with average error rates between 35% (interval) and 46% (pitch and melody). This shows that these techniques are effective for a subset of people, such as trained musicians, but challenging for some others.

6.1.1 Impact of Training. Interestingly, while training improved performance for most groups and techniques, participants classified as intermediates showed the greatest improvement in error rates after training, particularly for the pitch tasks. This suggests that some people who perform poorly initially can improve with relatively little practice.

Of the three techniques, interval appears to be the hardest to master. Even experts had an error rate of 8.6% after training, whereas they achieved 1.3% on pitch and 4.5% on melody tasks. Furthermore, with interval tasks, experts slightly worsened after training, perhaps indicating fatigue or impatience.

The pitch technique showed the best performance after training. Before training, novices (73.4%), intermediates (58.5%), and experts (19.9%) all had high error rates. While novices continued to have high error rates after training (45.9%), error rates were reasonably low for both intermediates (13.1%) and experts (1.3%) after training. This suggests that this technique is the most effective.

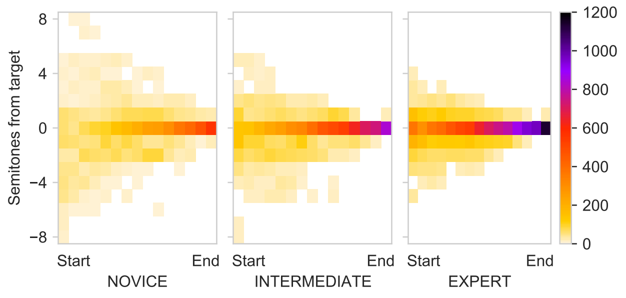


Figure 14: Heatmap for successful pitch trials showing frequency (number of semitones away from the target pitch) by time (from the start to the end of each trial) and SKILL.

6.1.2 Patterns in Successful Trials. Further analysis of the pitch experiment's data shows that experts were frequently able to find targets immediately, whereas novices typically had to slide into the appropriate pitch. The heatmap in Fig. 14 shows novices start on target for very few successful trials. On the other hand, for experts, the high density concentration on the correct target throughout the trial time (along $y = 0$ in the figure) shows they frequently start on target. Furthermore, from observing the distribution of frequencies at the start of the trials, most EXPERT trials were within 3 semitones of the target pitch right from the beginning, whereas NOVICES were frequently much further. This indicates novices heavily rely on the visualization to find the correct pitch, whereas experts only need it for fine-tuning.

6.1.3 Comparing Total Time. Unsurprisingly, total time increases with interactions that have more pitches. After training for all participants, pitch averaged 4319ms, interval averaged 4720ms, and melody averaged 5073ms. Interestingly, time does not increase proportionally to the number of notes in an interaction. While between-experiment learning may play a role, the surprisingly small increase in time is largely due to differences in recognition methods. After singing the first note, subsequent notes can be shorter (150ms for intervals, 50ms for melodies). Thus, adding additional notes to our interactions have less of an impact on time than one might initially expect.

In summary, all of our presented techniques are feasible for a subset of the population. However, some adjustments are necessary to make them easier for non-musical users.

6.2 Enjoyability

Overall ratings of enjoyability indicate that all interactions were at least somewhat enjoyable for most participants. P6 (EXPERT), P7, and P13 (INTERMEDIATES) remarked that they really enjoyed the single pitch tasks. Furthermore, beyond their uses for general computing, the interactions could be used for ear training while performing day-to-day computing tasks. P6 remarked they would "definitely use it to train my pitch in the future!"

P6 (EXPERT) and P16 (NOVICE) noted that by the melody experiment, they had mental and vocal fatigue. One can mitigate vocal fatigue by singing more softly or humming. With practice, the techniques would likely become much less mentally fatiguing. Nevertheless, these techniques are demanding and should be used in moderation in a real-world application.

Some participants found the tasks very challenging. P21 (NOVICE) said tasks were stressful and difficult to understand. Regarding melody tasks, P1 (INTERMEDIATE) said, "As a person who absolutely does not sing, I would have rather ran a 5km than do the task haha." Our techniques are probably best suited for motivated users who want to sing and improve their pitch.

It is possible that enjoyability could be improved by customizing the interactions based on the user's musical tastes and knowledge. For instance, a user could define their own interactions in an application, such as an interval or melody from a favourite song. This would make the interactions more familiar. Furthermore, using melodies from songs they enjoy might help them enjoy the interactions.

6.3 Design Considerations

Our three experiments highlight key design considerations for using pitch, interval, and melody interactions in real computer interfaces.

6.3.1 Use Background Music. Interfaces using singing interactions should incorporate some form of background audio with an appropriate tonic and tuning. While background music did not affect error rate, and despite slightly increasing total time for pitch tasks, the majority of participants found tasks easier with background music in all experiments. P14 (EXPERT) stated that not having background music at the start of the study made it difficult to find pitches until they figured out their tonic, and P4 (EXPERT) said the music helped them "count up" from the tonic to find pitches. It also had positive effects for ensuring consistent performance over time: P12 (EXPERT) found it prevented them from getting out of tune, and P5 (INTERMEDIATE) said, "the music made it easier to hold the note steady." Thus, while background music did not always improve error rate, it did improve subjective metrics.

Background music should be simple and subtle. P4 (EXPERT) and P19 (NOVICE) found the music distracting and P12 (EXPERT) suggested simply playing the tonic every few seconds would be as good as having full background music. This might explain why background music increased total time for pitch tasks. P16 (NOVICE) also said that when the background music played the tonic, it was easier to sing pitches correctly. This suggests that simpler music that places even more emphasis on the tonic might improve error rates. Future work should explore how various types of auditory stimulation impacts performance.

6.3.2 Facilitate Re-calibration. For interfaces using singing interactions, it should be easy to recalibrate the system to a different vocal range. We did not enable this to ensure differences in performance were due to training. However, P19 (NOVICE), P5 (INTERMEDIATE), and P24 (EXPERT) felt tasks would have been easier with a different range than the one they selected initially.

6.3.3 Reduce Pitch Detection Granularity. Pitch detection can use a lower granularity to reduce errors. Our system was capable of recognizing any of 12 possible pitches in Western music by determining which of the 12 semitones were closest to the f_0 of the voice. This was a high level of granularity, resulting in small margins

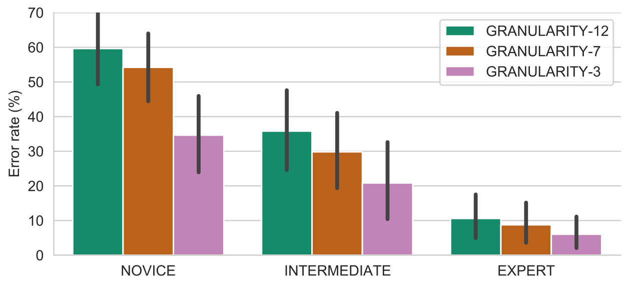


Figure 15: Pitch task *Error Rate* by SKILL for each GRANULARITY.

for error. This made it challenging to sing some notes according to comments from P1 (NOVICE), P12, and P24 (EXPERTS). We can reduce the granularity to allow for a larger margin of error.

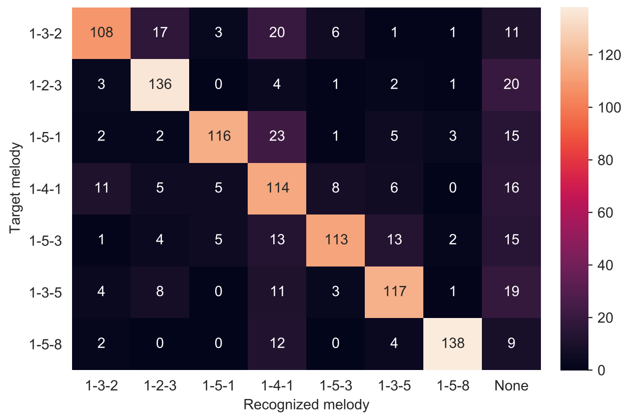
We simulated the three experiments with logged data using three pitch recognizers that supported a subset of the 12 semitones. In this follow-up analysis, we add an independent variable to our existing model: GRANULARITY with 3 levels (GRANULARITY-12, GRANULARITY-7, GRANULARITY-3). Each level represents a different pitch recognizer: GRANULARITY-12 recognizes all 12 pitches, as in our original experiments; GRANULARITY-7 recognizes only the 7 pitches in the major scale; and GRANULARITY-3 recognizes only degrees 1, 3, and 5. In the former two, all interactions were possible because our interactions only used pitches in major scale. In the latter, only 3 interaction types could be used for each technique because of the much more limited granularity.

For brevity, we only present results for the pitch technique (Fig. 15). The interval technique has similar results, while the melody technique shows minimal improvement due to the use of DTW. In the analysis to follow, a GRANULARITY × SKILL ANOVA with Tukey HSD post hoc tests was used on ART-transformed *Error Rate*.

We found that decreasing granularity decreased *Error Rate* by as much as 14.3% for the pitch technique. There was a main effect of GRANULARITY on ART-transformed *Error Rate* ($F_{2,36} = 115.13$, $p < .0001$, $\eta^2_G = .86$). Post hoc tests show GRANULARITY-12 *Error Rate* (34.1%) was higher than GRANULARITY-7 (29.9%, $p < .003$), which was higher than GRANULARITY-3 (19.8%, $p < .0001$). This shows a large improvement in *Error Rate*, as indicated by the standardized effect size of .86.

Decreasing granularity primarily improved *Error Rate* for novices and intermediates. There was an interaction between SKILL and GRANULARITY on *Total Time* ($F_{4,36} = 27.8$, $p < .0001$, $\eta^2_G = .76$). Post hoc tests show decreasing granularity from GRANULARITY-12 to GRANULARITY-3 decreased *Error Rate* much more for NOVICES (−25.0%) and INTERMEDIATES (−15.0%) compared to EXPERTS (−4.6%) (all $p < .0001$). This suggests that while EXPERTS still perform better with lower granularity, they are quite capable with higher granularity interactions.

In practice, one should certainly use GRANULARITY-7 instead of GRANULARITY-12. It has a positive effect on performance and virtually no drawbacks since we already recommend only using the 7 pitches from the major scale. For NOVICE users, it might make sense to use GRANULARITY-3 to further improve performance and make the techniques more accessible initially. Of course, there is a



Target melody	1-3-2	1-2-3	1-5-1	1-4-1	1-5-3	1-3-5	1-5-8	None
1-3-2	108	17	3	20	6	1	1	11
1-2-3	3	136	0	4	1	2	1	20
1-5-1	2	2	116	23	1	5	3	15
1-4-1	11	5	5	114	8	6	0	16
1-5-3	1	4	5	13	113	13	2	15
1-3-5	4	8	0	11	3	117	1	19
1-5-8	2	0	0	12	0	4	138	9

Figure 16: Confusion matrix with the number of times each melody was recognized for each target melody.

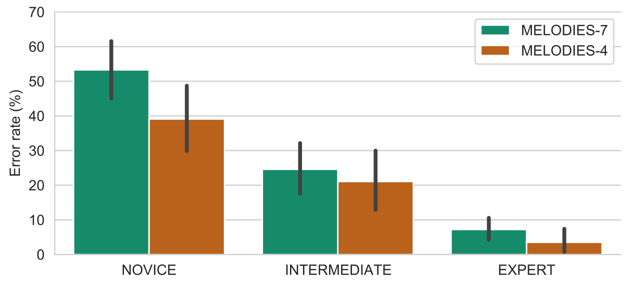


Figure 17: Melody task *Error Rate* by SKILL for each set of MELODIES.

tradeoff in the number of commands or modes that can be invoked in a real application. Since the improvement is relatively small for experts, one could use GRANULARITY-3 as a beginner mode before a user is ready for GRANULARITY-7.

6.3.4 *Reduce the Number of Melodies.*

Choosing melodies requires care and using fewer melodies can improve performance. We evaluated the use of 7 melodies, chosen to be easy to sing and distinguish. Despite this, some melodies were frequently confused with one another, as indicated by the confusion matrix in Fig. 16.

Three of our targets were frequently confused with other melodies: $1-3-2$, $1-5-1$, and $1-5-3$. This aligns with participant feedback: P5 (INTERMEDIATE) found it difficult to sing melodies that went up and then down again, especially when they did not return to note 1.

We simulated the melody experiment using a melody recognizer that only recognized the other four melodies to see if error rates improved. In this follow-up analysis, we add another independent variable: MELODIES with 2 levels (MELODIES-7, MELODIES-4) representing the number of possible melodies. In the analysis to follow, a MELODIES × SKILL ANOVA with Tukey HSD post hoc tests was used on ART-transformed *Error Rate*.

We found that decreasing the number of melodies decreased *Error Rate* by 7.5% (Fig. 17). There was a main effect of MELODIES on *Error Rate* ($F_{1,18} = 42.21$, $p < .0001$, $\eta^2_G = .70$), where MELODIES-7 (27.5%)

was higher than MELODIES-4 (20.4%). This is a large improvement, as indicated by the standardized effect size of .70.

Decreasing the number of melodies was especially helpful for novices. There was an interaction between MELODIES and SKILL on *Error Rate* ($F_{2,18}$ = 6.01, $p < .01$, $\eta^2_G = .40$). Post hoc tests show NOVICES (−14.1%) improved their rate more than EXPERTS (−3.7%, $p < 0.003$). This result mirrors how decreasing granularity especially improved *Error Rate* for novices.

The practical implications of these results is that, while experts can perform a wide variety of melodies, using fewer melodies can improve performance for novices. To make interactions easier, one can consider using melodies that only ascend or that return to the tonic, potentially making them easier to perform. Furthermore, one must ensure melodies are very different from one another to ensure they can be distinguished effectively.

6.3.5 Interface Visualization Improvements. The visualization should clearly communicate when a user can stop singing and what melody is recognized. P2 (INTERMEDIATE) noted it was hard to see how long they needed to hold a pitch. A clearer indication than just changing the colour of the arc (see Fig. 2), such as changing the background colour, might reduce *Total Time*. Furthermore, both P4 (EXPERT) and P7 (INTERMEDIATE) found it challenging to understand the visualization for melody tasks because it moved frequently based on the most recent estimate from the DTW algorithm. This could be improved by displaying the recognized melody less frequently, perhaps once per second, as opposed to every 50ms.

6.4 Limitations

One potential limitation in our work is the variable system setup, which is a consequence of being a remote study. While all devices passed our system test, some participants likely used slower systems in suboptimal environments. This may have reduced performance compared to a tightly controlled experiment, but it allowed us to achieve some degree of external validity in varied environments.

Another limitation is that visualizations we used were not explicitly evaluated. While there was no strong evidence of the visualization causing significant problems, it is possible alternative visualizations could result in different performance. Because we needed a visualization that would support pitch, interval, and melody interactions, we deviated from the simple linear visualization suggested by Sporka [22]. Pilot studies guided our design for comprehension and visual appeal, but more research is needed to evaluate the effect of visualization on task performance.

6.5 A Vision for Using Singing Interactions in Real Interfaces

We envision singing interactions as supplementing traditional input methods, similar to how keyboard shortcuts supplement mouse input. As one concrete example, consider a drawing application (Fig. 18). When drawing, one could switch to the eraser tool by singing pitch 4, then back to the paintbrush with 5. Changing brush size could be done with an interval starting at 1 and sliding up to the desired brush size. The melody 3 − 2 − 1 could copy the user's selection and 3 − 4 − 5 could paste. In such an application, the singing could complement the creative flow of the work.

It would also be possible for users to create custom mappings for specific workflows. For instance, mapping the melody of the "Batman" theme song to inverting colour, or the starting interval in "All Star" by Smash Mouth to selecting a star shaped brush. These mental connections between songs and interactions could make shortcuts memorable. In the accompanying video, we include a clip of an envisioned drawing application using post-production video effects.

7 CONCLUSION

Our work has explored the design of pitch, interval, and melody interactions for computer input and evaluated their feasibility and enjoyability. In a series of three experiments, we found that the highest performers could perform all interaction types with reasonable speeds and low error rates after training, whereas the lowest performers struggled with high error rates.

Moving forward, we hope to make singing a viable input modality to assist in drawing pictures, editing documents, and browsing the web.

REFERENCES

[1] MIDI Manufacturers Association. 1996. Complete MIDI 1.0 Detailed Specification.

[2] Roger Bakeman. 2005. Recommended effect size statistics for repeated measures designs. *Behavior Research Methods* 37, 3 (Aug. 2005), 379–384. https://doi.org/10.3758/BF03192707

[3] Supadaech Chanjaradwichai, Proadpran Punyabukkana, and Atiwong Suchato. 2010. Design and evaluation of a non-verbal voice-controlled cursor for point-and-click tasks. In *Proceedings of the 4th International Convention on Rehabilitation Engineering & Assistive Technology (iCREATe '10)*. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre, Midview City, SGP, 1–4.

[4] Jackson Feijó Filho, Wilson Prata, and Thiago Valle. 2013. Pufftext: A puff controlled software-based hands-free spin keyboard for mobile phones. In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 468–471. https://doi.org/10.1145/2493190.2494661

[5] Susumu Harada, James A. Landay, Jonathan Malkin, Xiao Li, and Jeff A. Bilmes. 2006. The vocal joystick: Evaluation of voice-based cursor control techniques. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '06)*. ACM, New York, NY, USA, 197–204. https://doi.org/10.1145/1168987.1169021

[6] Susumu Harada, T. Scott Saponas, and James A. Landay. 2007. VoicePen: Augmenting pen input with simultaneous non-linguisitic vocalization. In *Proceedings of the 9th International Conference on Multimodal Interfaces (ICMI '07)*. ACM, New York, NY, USA, 178–185. https://doi.org/10.1145/1322192.1322225

[7] Susumu Harada, Jacob O. Wobbrock, and James A. Landay. 2007. Voicedraw: A hands-free voice-driven drawing application for people with motor impairments. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '07)*. ACM, New York, NY, USA, 27–34. https://doi.org/10.1145/1296843.1296850

[8] Susumu Harada, Jacob O. Wobbrock, and James A. Landay. 2011. Voice games: Investigation into the use of non-speech voice input for making computer games more accessible. In *Human-Computer Interaction – INTERACT 2011 (Lecture Notes in Computer Science)*, Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler (Eds.). Springer, Berlin, Heidelberg, 11–29. https://doi.org/10.1007/978-3-642-23774-4_4

[9] Brandi House, Jonathan Malkin, and Jeff Bilmes. 2009. The VoiceBot: A voice controlled robot arm. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 183–192. https://doi.org/10.1145/1518701.1518731

[10] Perttu Hämäläinen, Teemu Mäki-Patola, Ville Pulkki, and Matti Airas. 2004. Musical computer games played by singing. In *Proceedings of the 7th International Conference on Digital Audio Effects*. Naples, Italy, 367–371.

[11] Takeo Igarashi and John F. Hughes. 2001. Voice as sound: Using non-verbal voice input for interactive control. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST '01)*. ACM, New York, NY, USA, 155–156. https://doi.org/10.1145/502348.502372

[12] Global Web Index. 2018. *Voice Search Insight Report.* Technical Report.

[13] Chang-Hung Lin, Yuan-Shan Lee, Ming-Yen Chen, and Jia-Ching Wang. 2014. Automatic singing evaluating system based on acoustic features and rhythm.

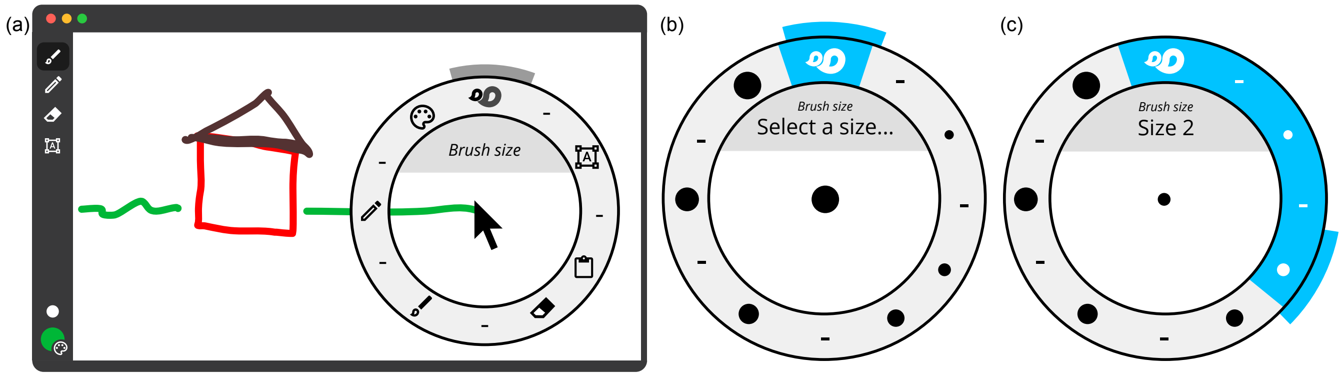


Figure 18: Illustration of how pitch can be used in a drawing application: (a) singing opens a visualization around the cursor with possible tools and settings; (b) when the first pitch of an interval interaction is recognized, more options appear for the second pitch; (c) when the second pitch is recognized, the interface displays the resulting effect.

In *2014 International Conference on Orange Technologies (ICOT '14)*. IEEE, Xi'an, China, 165–168. https://doi.org/10.1109/ICOT.2014.6956625

[14] David Marino, Paul Bucci, Oliver S. Schneider, and Karon E. MacLean. 2017. Voodle: Vocal doodling to sketch affective robot motion. In *Proceedings of the 2017 Conference on Designing Interactive Systems (DIS '17)*. ACM, New York, NY, USA, 753–765. https://doi.org/10.1145/3064663.3064668

[15] Philip McLeod and Geoff Wyvill. 2005. A smarter way to find pitch. In *Proceedings of the 2005 International Computer Music Conference (ICMC '05, Vol. 5)*. ACM, Barcelona, Spain, 138–141.

[16] Thomas Murry. 1990. Pitch-matching accuracy in singers and nonsingers. *Journal of Voice* 4, 4 (Jan. 1990), 317–321. https://doi.org/10.1016/S0892-1997(05)80048-7

[17] Nathalia Peixoto, Hossein Ghaffari Nik, and Hamid Charkhkar. 2013. Voice controlled wheelchairs: Fine control by humming. *Computer Methods and Programs in Biomedicine* 112, 1 (Oct. 2013), 156–165. https://doi.org/10.1016/j.cmpb.2013.06.009

[18] Ondřej Poláček and Zdeněk Míkovec. 2010. Hands free mouse: Comparative study on mouse clicks controlled by humming. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems (CHI EA '10)*. ACM, New York, NY, USA, 3769–3774. https://doi.org/10.1145/1753846.1754053

[19] Ondřej Poláček, Zdeněk Míkovec, and Pavel Slavík. 2012. Predictive scanning keyboard operated by hissing. In *Proceedings of the 2nd IASTED International Conference on Assistive Technologies (AT '12)*. ACTA, Innsbruck, Austria, 862–869. https://doi.org/10.2316/P.2012.766-002

[20] Ondřej Poláček, Zdeněk Míkovec, Adam J. Sporka, and Pavel Slavik. 2011. Humsher: A predictive keyboard operated by humming. In *Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '11)*. ACM, New York, NY, USA, 75–82. https://doi.org/10.1145/2049536.2049552

[21] Daisuke Sakamoto, Takanori Komatsu, and Takeo Igarashi. 2013. Voice augmented manipulation: Using paralinguistic information to manipulate mobile devices. In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 69–78. https://doi.org/10.1145/2493190.2493244

[22] Adam J. Sporka. 2009. Pitch in non-verbal vocal input. *ACM SIGACCESS Accessibility and Computing* 94 (June 2009), 9–16. https://doi.org/10.1145/1595061.1595063

[23] Adam J. Sporka, Torsten Felzer, Sri H. Kurniawan, Ondřej Poláček, Paul Haiduk, and I. Scott MacKenzie. 2011. CHANTI: Predictive text entry using non-verbal vocal input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2463–2472. https://doi.org/10.1145/1978942.1979302

[24] Adam J. Sporka, Sri Hastuti Kurniawan, and Pavel Slavík. 2004. Whistling User Interface (U3I). In *User-Centered Interaction Paradigms for Universal Access in the Information Society (Lecture Notes in Computer Science)*, Christian Stary and Constantine Stephanidis (Eds.). Springer, Berlin, Heidelberg, 472–478. https://doi.org/10.1007/978-3-540-30111-0_41

[25] A. J. Sporka, S. H. Kurniawan, and P. Slavík. 2006. Non-speech operated emulation of keyboard. In *Designing Accessible Technology*, John Clarkson, Patrick Langdon, and Peter Robinson (Eds.). Springer, London, 145–154. https://doi.org/10.1007/1-84628-365-5_15

[26] Adam J Sporka, Ondřej Poláček, and Jan Havlik. 2012. Segmentation of speech and humming in vocal input. *Radioengineering* 21, 3 (2012), 7.

[27] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only ANOVA procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, Vancouver, BC, Canada, 143–146. https://doi.org/10.1145/1978942.1978963

[28] Daniel Zielasko, Sebastian Freitag, Dominik Rausch, Yuen C. Law, Benjamin Weyers, and Torsten W. Kuhlen. 2015. BlowClick: A non-verbal vocal input metaphor for clicking. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction (SUI '15)*. ACM, New York, NY, USA, 20–23. https://doi.org/10.1145/2788940.2788953