ArmKeyBoard: A Mobile Keyboard Instrument Based on Chord-scale System and Tonal Hierarchy

Jun-qi Deng
The University of Hong Kong
jqdeng@eee.hku.hk

Francis Chi Moon Lau
The University of Hong Kong
fcmlau@cs.hku.hk

Yu-kwong Kwok
The University of Hong Kong
ykwok@eee.hku.hk

ABSTRACT

Traditional keyboard instruments, with their sheer size and key anisotropy, although are versatile in musical expression, are difficult to learn and inconvenient to carry around; and its linear layout somewhat rules out the musical possibility of non-linearity. Trying to address this, we design a keyboard with both linear and non-linear layouts based on chord-scale system and tonal hierarchy. Several flipping mechanisms and mapping algorithms are devised to try to equip this small portable keyboard with as much musical expression capability as possible as compared with a traditional keyboard. Evaluation results show that both the musical outcome and user experience of ArmKeyBoard are satisfactory, although people may still prefer a linear keyboard to a non-linear one.

1. INTRODUCTION

The keyboard, although is versatile and popular, is not necessarily the most ideal device for music generation in all situations and to all users. First, it is not most easy to be carried around. Second, a non-player wanting a device to quickly express a musical idea would find the learning, which is non-trivial, too much an overhead. Third, the same type of chord or scale in different keys are laid out differently, which adds to the learning difficulty. The list can go on. Additionally, the keyboard has a linear layout of the keys, which works well with music expressions that exhibit certain linearity [1], but is less effective for modernistic non-linear styles such as that of serialistic and stochastic music which is gaining acceptance in the musical world.

1.1 Existing Mobile Keyboards

Replacing the physical keyboard by a mobile app that mimics the keyboard might solve the size problem. There are many keyboard apps in the market [2][3], which try to fit the 88 keys into a small touchscreen. They copy verbatim the keyboard layout and provide extra buttons or a sliding mechanism to switch between different octaves. In exchange for the shrunk size, the user has to put up with the trouble of changing octaves which easily gets in the way during playing. The learnability problem does not get resolved as the user is playing essentially the same thing—the keyboard. These designs have fallen in the trap of the NIME design principle of "Copying an instrument is dumb" [4]. A smarter app [5] would let the user choose from a collection of chord-scales, which are musical scales fitting the underlying chord or chord progression harmonically, and lay out the selected scale in white keys only. Since all the keys shown are those of a scale, users can play beautiful melodies at once without much learning. But still, changing chord-scale or octave in such apps requires excessive extra movements that may hamper real-time performance. Some researchers have proposed other ways of keyboard layout [6][7] on the tablet, aiming to ease learnability. These are great attempts towards making the keyboard accessible to more users, but the learning curve is still prohibitively steep for many who have no prior experience with keyboard playing. Using them to generate a beautiful melody could be a challenge for ordinary users. And since these kinds of keyboard contain all the possible notes on the screen regardless of chord-scale, they can only be implemented on a tablet. Most of the existing approaches we are aware of are somewhat far from the goal of a full-fledge portable keyboard and easiness to learn, and almost none of them consider or have incorporated non-linearity in their design.

1.2 ArmKeyBoard

Trying to solve the above mentioned problems, we design a new keyboard. Based on the NIME design principles [4]—specifically the "Make a piece, not an instrument or controller" and "Instant music, subtlety later"—our keyboard leverages a chord-octave-scale sequence grid to pack 88 keys into a 15–17 keys-sized screen, and features an almost zero learning curve for the production of beautiful and sophisticated melodies. It offers both linear and non-linear layouts. The non-linear layout is mapped to a user chosen image by an algorithm based on contour separation and tonal hierarchy. We call this keyboard "ArmKeyBoard", where "Armkey" means suitable, comfortable, and in-tune, in our spoken dialect.

The following sections are structured as follows: section 2 briefly introduces the characteristics of linear and non-linear keyboard; section 3 examines the user behaviors and the affordances of a mobile smart phone; based on the previous two sections, section 4 describes Armkeyboard's design of note-space based on chord-scale system, together with some expression controlling mechanisms; section 5

describes how Armkeyboard leverages the knowledge of linear and non-linear layout to map the notes to image regions based on tonal hierarchy. Then we have the evaluation at section 6 based on an existing digital instrument evaluation framework, followed by the discussion section.

2. TWO TYPES OF KEYBOARD LAYOUT

In the remaining discussion, "keyboard" refers to an instrument implementing a series of key-note pairs and deterministically generates a note when a key is pressed. "Layout" refers to the spatial arrangement of those key-note pairs.

Linear layout is characteristic of a piano. From left to right, each key is mapped to a unique note value. Every next key is mapped to a note value exactly 1 (in MIDI terms) higher than the previous one. This has a significant impact on music making, since people naturally feel more comfortable with playing on adjacent keys than non-adjacent keys, leading to smaller intervals appearing more often than larger ones, as can be seen in music literature such as "the Real Book" [8].

Non-linear layout can have many more possibilities, such as a random note being paired with a random key or one note being paired with several keys. Note that in our current discussion, several notes being paired with one key is not valid by the definition of keyboard. Non-linearity may further allow using any key setting other than the traditional setting. For example an image can be divided into several sections, each serving as a key of the keyboard. The idea of non-linear keyboard is not new. There are existing applications such as [9] or papers such as [10] talking about similar ideas, most of which are built around the idea of sampling. In our design, however, the audio content generated by a key is a note, and we focus on the non-traditional arrangement of keys and notes, and not sampling.

3. MOBILE SMART PHONE

If portability is a concern, the best solution to having a keyboard is to implement it on a smart phone. We had the following considerations concerning smart phones before went ahead with the design.

1. We assume the users are mostly non-musicians. This calls for a flat learning curve so that the users are able to play good but not too simple music, which can help reach the goal of making the piano keyboard available to as many people as possible.

2. The smart phone's screen is small. This means there could be fuzziness in touching a certain key. And the degree of fuzziness is dependent on the number of keys. If there is only one key which occupies the whole screen, there is no fuzziness issue at all because every tap on the screen results in the same note. If all 88 keys are to be equally distributed on an iPhone 5 screen, each key has only a space of about $82mm^2$ (less than $1cm^2$), which translates to much fuzziness. In a word, since the screen is small, we have to make good use of the space, do clever mapping and embrace fuzziness.

3. The smart phone is programmable. This means we can apply whatever mapping and try whatever level of fuzziness we want. We can make the layout linear or non-linear. For non-linear layouts, there are countless possibilities. Musical concerns can be leveraged to rule out some of these possibilities. For example, if we assume that within any short range of a musical process, all the perceptible notes must belong to a certain chord-scale if this short range itself belongs to a larger meaning group, then the keyboard should minimally play one chord-scale at a time. In our design, we actually apply this assumption, which is reasonable for many classical or non-classical music forms [11], although it may not apply to some modern or contemporary music forms such as serial music, twelve-tone music, or stochastic music. For these different music forms, we need to make different musical assumptions ahead of the design.

4. NOTE SPACE AND CONTROL

Now we start our discussion of the design of ArmKeyBoard. This section is about how to make efficient use of the available screen space and how to make it possible for a new user to create good music instantly. The next section talks about the linear and non-linear keyboard layout adopted by ArmKeyBoard and the key-note mapping algorithms in different keyboard layouts.

4.1 Chord, Scale and Octave

With respect to the assumption in the last section—the keyboard should minimally play one chord-scale at a time, ArmKeyBoard treats the small screen as a cache, caching the currently playing chord-scale in the current octave range, while other octaves and chord-scales are waiting to be loaded when needed. In view of the small screen size, we decide to cache 15–17 notes—two octaves of a scale.

Changing chord-scale or octave on a piano in real time is easy for a pianist, but could be a nightmare for non-pianists. Therefore, ArmKeyBoard needs a special mechanism to load other chord-scales and octaves into the foreground, so that the player can easily switch music expression ranges in real time. To this end, we adopt a chord-octave-scale sequence grid as shown in Figure 1.

Users can switch between different chord-octave-scales using a gravity X gesture (Figure 2). Meanwhile, users can also flip octaves within the same chord-scale using swipe gestures. To summarize, in Figure 3, the screen is a cache of the active note space, while the spaces around the active space can be loaded in real-time via gravity X or swipe gestures.

4.2 Expression Parameters

Because our design is based on the piano keyboard, the most dominant expression parameter, velocity, should be implemented. In the linear layout, since key-note mapping is 1-to-1 and the position of each key is equally distributed along the y-axis, velocity can be easily controlled by position X. While in the non-linear layout, position X cannot be used because keys can be in any shape and anywhere;

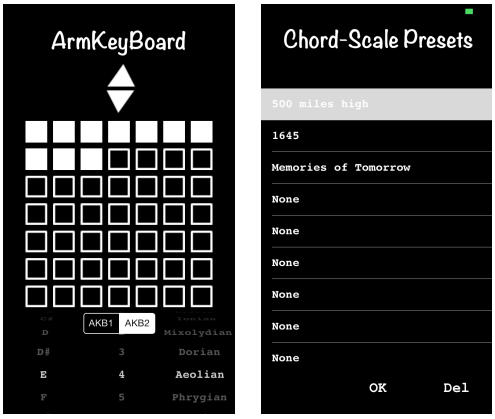

Figure 1. On the left is the chord-octave-scale grid, where each square can be set as one chord-octave-scale combination (such as C-4-Lydian), and consecutive squares form a sequence which can be saved as preset; The sequence is read from left to right, and when it reaches the rightmost, back to the leftmost square on the next line; On the right is the chord-octave-scale preset browser looking at the already saved chord-octave-scale sequence presets.

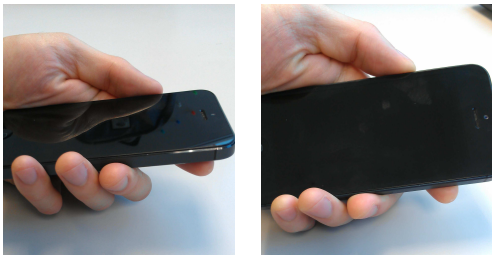

Figure 2. Gravity X gesture, which is used for switching to the next or previous page of notes determined by the chord-octave-scale combination at the next or previous square within the sequence.

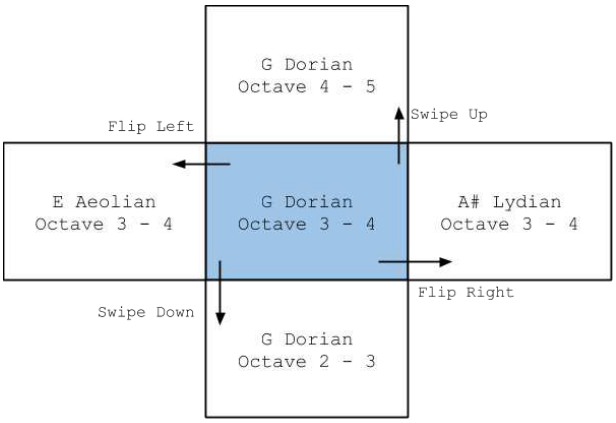

Figure 3. Chord-octave-scale control. The horizontal arrows indicate changing page of notes according to chord-octave-scale sequence, while the vertical arrows indicate changing page of notes to a higher or lower octave only.

thus we use gravity Y to control the velocity (Figure 4). Note that we do not consider sustain, since if a hand gesture were to convey what is originally conveyed by foot, it might make learning difficult for the ordinary users. So we decided to sustain every note; or we give the user a choice to plug in an external foot pedal controller. Of course, we sometimes need to quit the keyboard and set up everything again. In that case, we use a gravity Z gesture to handle it (Figure 4).

Figure 4. Gravity Y gesture (on the left), which is used for controlling note velocity, leading to a smaller velocity with a larger angle to the horizontal plane; Gravity Z gesture (on the right), which is used for quitting the current keyboard to reset everything again

5. KEY-NOTE MAPPING

ArmKeyBoard has both linear and non-linear keyboard layout, called "AKB1" and "AKB2" respectively. Since our current implementation is on an iOS device, our discussion focuses on the iOS platform.

5.1 Linear Layout and Mapping

"AKB1" (Figure 5) contains 15–17 notes within the active chord-octave-scale and they are mapped linearly to 15–17 bars equally divided along the y-axis. The velocity is controlled by the X position.

5.2 Non-linear Layout and Mapping

"AKB2" is a user selected image (Figure 5). The image is algorithmically divided into contours and they are algorithmically mapped to the 15–17 notes within the currently active chord-octave-scale. The algorithms are described below.

5.2.1 Contour Separation

The contour separation is processed using opencv [12]. The image is first transformed to opencv Mat which is then passed to a contour separation function. The function then: Step 1, turns the Mat into gray scale and slightly performs a blur operation on it; Step 2, passes the output of step 1 (a gray scale Mat) to an edge detection function (the output is a binary Mat with the edge pixels set as step 1); Step 3, passes the output of step 2 to a findContour function, which finds contours, stores them in an array and calculates the contour hierarchy (a tree structure describing the inclusion relationship of contours); Step 4, calculates the

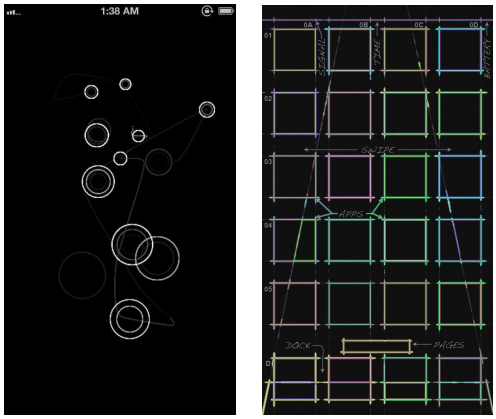

Figure 5. AKB1 (on the left) is a linear keyboard with a higher pitch at smaller y position value, and larger note velocity at larger x position value, each page contains 15–17 notes; AKB2 (on the right) is a non-linear keyboard mapping the page of 15–17 notes to the detected contours within the user selected image, where the mapping procedure is based on the correlation between the importance of contours (or regions) and the tonal hierarchy of notes.

area of each contour, discards those below a certain size and deletes their nodes in the hierarchy; Step 5, creates an outer contour which is the whole screen subtracted by all the contours output by step 4. The output of all the above steps is an array of valid contours (each contour is itself an array of its vertices), and a hierarchy structure describing the inclusion relationship of these contours.

5.2.2 Contour Ranking

Remember that our goal is to map 15–17 notes to the contours, we need to decide which note to map to which contour. The minimal musical concern is, when the keyboard is being played, the notes being generated should at least imply the currently active chord-scale most of the time. Note that we would not demand it should "always" behave this way, but "most of the time". For example, in G-Ionian, the keyboard is supposed to generate notes that form a tonal gravity at G and imply G major chord most of the time, but sometimes it may also sound like C-Lydian (tonic at C).

We still need more assumptions to connect this musical concern to contours. We assume that most user tends to tap on: 1, a contour with a larger area; 2, a contour closer to the center of the screen; 3, a contour that contains more sub-contours. Based on how often most users will tap on a contour, its importance can be determined. Thus in the implementation, we rank the contours based on the weighted sum of the above three indices. This corresponds to how important a note is in implying a certain chord-scale, which will be discussed below.

5.2.3 Tonal Hierarchy

Similar to ranking contours, if we also rank the notes within a chord-scale, then what is left is to map the two rankings. According to [13], there is a certain tonal hierarchy within

Table 1. Tonal Hierarchies in ArmKeyBoard. L1 is the first level of notes which are to be mapped to regions with highest importance, and L2 to be mapped to regions with second highest scores, then L3 to be mapped to the least important regions.

Scale	L1	L2	L3
Lydian	1, 5, 3, 7	2, #4, 6	
Ionian	1, 5, 3, 7	2, 6	4
Mixolydian	1, 5, 3, b7	2, 6	4
Dorian	1, 5, b3, b7	2, 4	6
Aeolian	1, 5, b3, b7	2, 4	b6
Phrygian	1, 5, b3, b7	4	b2, b6
Locrian	1, b5, b3, b7	4, b6	b2
Lydian b7	1, 5, 3, b7	2, #4, 6	
Altered	1, 3, b7	#4, b6, b2, #2	5
Sym. Dim.	none	none	
Mel. Minor	1, 5, b3, 7	2, 4, 6	

a chord-scale being played in bebop style jazz music, and this finding actually corresponds to the avoid note issue [14]. The tonal hierarchy theory says during the performance of a certain chord-scale, some notes are more often heard than others. If the notes are to be divided into a hierarchy according to how often they appear, the first class contains chord notes, the second class contains those a whole step above the chord notes and finally those half step above, with exceptions. The avoid note issue says basically the same thing, but with "more often heard" replaced by "more often played". This is not coincidence, because both of them originate from jazz music; note also that the former one is from bebop jazz while the latter one from modal jazz. The chord-scale system belongs to modal jazz, a successor of bebop jazz. One might argue that not all music are jazz, and therefore the chord-scale system may not apply for everything. It is true, but the chord-scale system in the macro perspective is a very good generalization of both traditional harmony and some of the modern harmony. We admit that using the chord-scale system as a crucial cue in designing ArmKeyBoard is neither perfect nor complete, but to music itself, there is no absolute right or wrong, only different assumptions. In our design, the chord-scale system is the main assumption.

Using the tonal hierarchy theory, we come up with Table 1 which lists all the hierarchies [14] of some of the most frequently used chord-scales [15]. The scale degree notation is used instead of note names. Note that "Sym. Dim." stands for symmetrical diminished scale and it has no hierarchies in our implementation, which means that all the notes are equally important. To deal with hierarchy across octaves, we follow a rule which dictates that the same pitch-class belongs to the same hierarchy level and a pitch with a lower octave has a higher priority than a pitch with a higher octave.

5.2.4 The Final Mapping

The final mapping is not so obvious as it may seem. Although we have a ranking of contours and a ranking of 15–17 notes within a chord-scale, they are by no means simply

1-to-1 mappings, because in reality we do not know how many contours there are and how large each of them is until the user selects an image. In view of this complication, we implement a simple but effective algorithm to do the final mapping:

1. Divide the screen size by the number of notes, and name the result RPN.
2. Look at the $ratio = Area(contour)/RPN$ of the top item of the sorted contour list (we treat it like a stack). If $ratio >= 1$, do step 3; otherwise do step 4. Repeat until no contour left in the stack.
3. Map notes to contour: pop the contour, pop the top $ceil(ratio)$ notes and pair them up. Go back to step 2.
4. Map contours to note: pop the contour, pair it up with the first note. Add $ratio$ to $accum$.
   If $accum >= 1$, clear $accum$ and pop the note. Go back to step 2.

With this, the most important notes are mapped to the most important contours, and contour with larger areas will contain more notes. But since multiple notes cannot be mapped to a single key, we need to decouple the notes within a contour that has more than one note. Instead of further separating a shape-unpredictable contour into several subcontours, we notice that the real "key" in question is composed of pixels, and thus we use a heuristic way to decouple the notes is to distribute their keys across the contour according to a formula related to pixels and their RGB value:

$$noteIdx = ((X + Y)\%10 + (R + G + B))\%(15\ or\ 17)$$

Where 15 or 17 is the number of notes. We try to make position affect less and RGB affect more, while making sure all the notes are included regardless of the image. With this final step, we finish our discussion about the design and implementation of ArmKeyBoard.

6. EVALUATION

This section is divided into three parts. First, we talk about the evaluation framework, which is the basis of the whole evaluation. Then it comes to the evaluation method, which is designed according to the evaluation framework. And finally we give the evaluation result.

6.1 Evaluation Framework

Reference [16] is a paper on how to evaluate digital musical instruments. Basically it divides the evaluation process into four perspectives, namely, the audience's perspective (in what sense an audience knows the performer is "performing"), the performer's perspective (can the instrument successfully translate the performer's idea into sound?), the designer's perspective (evaluating playability and playing experience) and the manufacturer's perspective (marketing consideration). Based on this framework as well as one of its practical use cases [17], and taking into account the current state of ArmKeyBoard, we decide to build our evaluation around two dimensions—audience experience and performer experience covering many dimensions in [16] while discarding some of them. For example, since the instrument is still in its initial stage, we omit the manufacturer's perspective, and since this instrument is not supposed to be only linear, it makes little sense to evaluate how well it can replicate existing musical clips. Besides, we also incorporate an important evaluation idea derived from a famous quote of Duke Ellington "If it sounds good, it IS good." [18] to be the very first and fundamental criterion of audience experience. Actually although ArmKeyBoard tries to pack a lot of key-note pairs into a relatively small screen, it is by no means a piano with 88 keys. A piano keyboard, of course, is unparalleled in terms of musical expression if proficiently mastered, but still we are interested in to what extent, and how well, ArmKeyBoard can do by a non-pianist in terms of musical expressions.

6.2 Evaluation Method

We gather evaluation results by questionnaires. According to the framework, we design two sets of questionnaires for audience experience and performer experience respectively. [1]

The audience experience questionnaire asks the audience to watch several videos capturing ArmKeyBoard played by a non-pianist either in improvisation scenario or solo scenario and then seeks the answers for: "Does it sound good?", "How much score will you give to the musical outcome of ...", "Piece 1 sounds like (choose from a variety of musical style)", etc. The questions are focused on the positive or negative ratings of musical outcomes, and not on the interaction between the performer and audience or whether the audience realizes the physical casualty relationship between the performer's gestures and the musical outcome.

The questionnaire of performer experience is to be filled in after the participants have played ArmKeyBoard in the solo scenario. Before they play, they are given the same instruction video showing them how to enter AKB1 and AKB2 and what are the gestures needed to control their expression. Beyond these, nothing else will interrupt the participant; the participant can ask the experimenter anything related during the test. The questions in this set are basically around the experience of using different images, the controlling mechanisms and multiple playability dimensions borrowed from [17], such as: "Do you think choosing different images will result in different music outcomes?", "How would you rate the chord-scale flipping mechanism?", "How much score will you give to the following aspects of the user experience of AKB1 - Fun", "How much score will you give to the following aspects of the user experience of AKB2 - Creativity", etc. To let the participants better understand those abstract items such as "Creativity", explanations are implied in the answering scale, such as a 0 is for "there isn't any creative points" and a 10 is for "there

[1] Links for the questionnaires and outcomes: http://gdriv.es/armkeyboardaudienceformfin http://gdriv.es/armkeyboardperformerformfin

are much room for a creative user to explore in terms of musical possibility".

The audience experience questionnaires are distributed via the authors' personal social networks on the Internet, and the receivers may further distribute them to their networks. The frontpage of the questionnaire link will inform the participant: "Thanks for your participation! This is a set of evaluation forms targeting the audience experience of a new musical expression interface. Please fill in both forms via the below two links, which takes you about 20 minutes. Please don't be afraid of being honest, because we have no idea who you might be. Your feedback will be very important to our research, therefore we sincerely appreciate your effort in offering help!" Thus the participants finish the questionanaires on their own computers without any interruption from the authors of this paper.

To gather results from the performer experience, we invite people we know to participate, and they are also told the same thing as shown above before filling in the evaluation. The duration of each evaluation case depends on subject's need, so as to let them explore Armkeyboard as much as possible. The average evaluation time is above 30 minutes. During the evaluation, the designer of Armkeyboard is with the evaluation subject in case he or she may have any questions. The designer keeps silent unless the subject raise a question regarding to Armkeyboard.

6.3 Evaluation Results and Discussions

The evaluation was closed on Jan. 19th 2014. All the summaries of the evaluation results are contained in the links provided in the last subsection. Here are some of them. All the scores are in a scale of 10.

6.3.1 Audience Experience

There are totally 33 responses from improvisation evaluation and 31 from solo evaluation, 5 and 4 of them regard themselves as musicians or amateur musicians respectively. Actually there could have been more musicians since the classification questions are added to the questionnaire after a few responses. But this does not affect the overall result. The improvisation evaluation is summarized as follows:

- 30 say the musical outcome of video one (AKB1) sounds good, 29 say the musical outcome of video two (AKB2) sounds good.
- The average score of the musical outcome of video one and video two are 7.64 and 7.30, with standard deviation 1.75 and 1.93 respectively. While the average scores of the 5 musicians are both 7.6, with standard deviation 2.07 and 1.52 respectively.

The solo evaluation is summarized as follows:

- 23 say the musical outcome of video one (AKB1) sounds good, 26 say the musical outcome of video two (AKB2) sounds good.
- The average scores they give to the musical outcome of video one and video two are 6.90 and 6.84, with standard deviation 2.23 and 2.07 respectively. While the average scores of the 4 musicians are 7.75 and 7.25, with standard deviation 2.06 and 3.40 respectively.
- In the "three musical outcomes with three different images" question, they are asked to rate how much the outcomes differ from each other in terms of the overall feeling; the average score is 5.3 with standard deviation 1.77, while the average score of musicians is 4.3 with standard deviation 1.50, where score 1 means "They are totally the same", and score 10 means "They are drastically different".
- Many of them regard piece 1 sound like jazz, while the answers to piece 2 vary.

It can be shown that ArmKeyBoard is quite satisfactory in that it got an average score of around 7 out of 10 in both improvisation and solo as rated by both non-musicians and musicians, and musicians seem appreciate the solo outcome more. For the open question "What do 3 - 5 sound like" in the solo evaluation, some musicians comment: "Debussy style of music" or "Most of the time it sounds like some one who doesn't know how to play the piano is trying to make some sound out of a piano. Some time it sounds like avant-grade music.", while a non-musician thinks: "to be honest it sounds like someone hitting the keyboard randomly, what eight year-old child would do when they are given a piano to play on." Comparing all the outcomes of AKB1 and AKB2, they all seem to like the former better, which may be due to their long-time exposure to linear musical outcomes. Interestingly in the solo evaluation, about 10 people say piece one sounds like jazz. So the conclusion here is that piece one is jazzy! As for piece two, 6 are for stochastic, 6 for Romanticism and 5 for Impressionism, and thus it can be concluded that for many people piece two is quite unstructured but at the same time contains some sort of meaning or beauty.

6.3.2 Performer Experience

15 people participate in the performer experience evaluation:

- 14 of them agree choosing different images will result in different music outcomes.
- The average score of chord-scale flipping is 7.60, with standard deviation 1.55, octave flipping 7.60, with standard deviation 1.72, velocity control of AKB1 8.75, of AKB2 8.27, with standard deviation 1.06 and 1.39 respectively.
- Other average scores and standard deviations in a format of AKB1 (std) - AKB2 (std): Fun, 8.40 (1.50) - 7.53 (1.46); Controllability, 7.93 (1.58) - 6.47 (1.73); Learnability, 8.33 (1.45) - 6.87 (1.96); Creativity, 8.33 (1.45) - 7.87 (1.30); Repeatability, 7.13 (2.42) - 5.33 (2.55); Overall, 8.33 (1.11) - 7.13 (1.68).

The mapping algorithm of AKB2 based on tonal hierarchy has been successfully implemented, which leads to a

different note distribution with a different image, and according to the third item of audience experience of the solo evaluation, although the overall feelings when different note distributions are played are not drastically different, they are different. The attitudes towards various control mechanisms are good, but with no base to compare within this evaluation. As with those other user experience dimensions, AKB1 always scores higher than AKB2, but still their average overall scores are both above 7. Below are some of the comments: "I like the first layout (AKB1) better because it's extremely convenient to use. I think the first layout is suitable for and will be attractive to both professionals as well as amateurs."; "It's hard to start with for musical novices ... AKB2 is not fun enough, although combining image and music is a very good idea, the musical outcome is not that good compared with AKB1, plus it can not repeat good phrase which might accidentally be played by users, thus they probably will not stick with AKB2."; "It would be great if there's more tutorial material and forum to share the creative art made by different artists (using ArmKeyBoard)."

Most results are as expected, since AKB2 is a non-linear keyboard, it is doomed to have controllability, learnability and repeatability problems, but what is not quite as expected is AKB2's low scores on fun and creativity. A possible explanation would be most of the participants, and even the authors themselves, have not spent much time playing with AKB2 thus they might have no idea how much musical possibilities could there be by feeding and playing it with hundreds of different images. This explanation somewhat corresponds to a comment from one player, who propose that if more time are allowed in the evaluation, he will definitly find AKB2 more interesting to play with. Theoretically speaking, AKB2 is more capable than AKB1 in terms of musical expression, since AKB2 is armed with arbitrary keyboard layout and non-linear note mapping, which are superset of a fix layout and linear note mapping. So we argue that AKB2 actually will be a more interesting choice if more time is spent to explore its various possibilities.

7. CONCLUSION

In this paper, we introduce the design concept of ArmKeyBoard, which came from the three deficiencies of traditional keyboard instruments and existing mobile keyboard instruments: 1. big size; 2. difficult to learn; 3. linear. Then based on the knowledge of two types of keyboard layout (linear, non-linear) and three characteristics of the smart phone (musical novice users, small screen size and programmability), the design details of ArmKeyBoard are elaborated, including the chord-scale and octave flipping mechanism, velocity control mechanism, linear note-position mapping and non-linear note-contour mapping algorithm based on tonal hierarchy theory. Finally the evaluation framework and method are described, and results are provided for audience experience and performer experience respectively. The results are quite positive in both the audience's and the performer's dimension, while the linear layout is more positive than the non-linear one.

Prompted in part by some of the feedbacks from the evaluation forms, the authors are now considering further possible improvements. Specifically, some AI modules will be added to ArmKeyBoard to: 1. automate the chord-scale grid setting procedure by analyzing an image, that is, to map an image's feeling to a specific chord-scale progression; 2. automate the improvisation by auto-flipping chord-scales—i.e., to let the device listen to the backing and get the right chord-scale at the right time; 3. automate the playing process, that is, to make the device "know" and suggest what a user should play in order to make good melodies.

8. ACKNOWLEDGMENTS

We'd like to give special thanks to Qing Cai, who helped us record the performer instruction video and provided us a lot of constructive feedbacks on the design. Thank Yixin Cao for many initial testings. Thank Shuting Chen, without whom the name of this keyboard wouldn't have come to our mind. And of course, thank all the participants who took part in our evaluations.

9. REFERENCES

[1] M. Uchida, "Mitsuko Uchida on Schoenberg's Piano Concerto," https://www.youtube.com/watch?v=PmWRttCo7lo, accessed: 2014-1-3.

[2] iGrand Piano, http://www.ikmultimedia.com/products/igrandiphone/, accessed: 2014-01-10.

[3] Piano Sharp, http://rambax.com/pianosharp.html, accessed: 2014-01-8.

[4] P. R. Cook, "Principles for Designing Computer Music Controllers," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2001, pp. 3–6.

[5] GarageBand for iOS, http://www.apple.com/ios/garageband/, accessed: 2014-01-8.

[6] M. M. Gao, "CYCLIC-N, a Real-time Mobile Multitouch Application for Composition, Improvisation and Education," in *Proceedings of the International Computer Music Conference*, 2011, pp. 79–82.

[7] S. Maupin, D. Gerhard, and B. Park, "Isomorphic Tessellations for Musical Keyboards," in *Proceedings of Sound and Music Computing Conference*, 2011, pp. 471–478.

[8] D. Ellington, M. Davis, J. Coltrane, etc., *The Real Book*, 5th ed. Hal Leonard.

[9] KONTAKTS, http://www.native-instruments.com/en/products/komplete/synths-samplers/kontakt-5/, accessed: 2014-1-14.

[10] N. Kruge and G. Wang, "MadPad : A Crowdsourcing System for Audiovisual Sampling," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2011, pp. 185–190.

[11] S. Smith, *Jazz Theory*, 4th ed., 2008, pp. 73–79.

[12] OpenCV, http://opencv.org/.

[13] T. Jarvinen, "Tonal Hierarchies in Jazz Improvisation," *Music Perception*, pp. 415–437, 1995.

[14] B. Nettles and A. Ulanowsky, *Harmony 1–4*. Boston, Massachusetts: Berklee College of Music, 1987, pp. 33–35.

[15] G. Burton, "Jazz Improvisation," http://www.jazzpiano.co.nz/wp-content/uploads/2012/09/Gary-Burton-Course.pdf, accessed: 2014-1-16 coursera lecture notes.

[16] S. O'Modhrain, "A Framework for the Evaluation of Digital Musical Instruments," *Computer Music Journal*, vol. 35, no. 1, pp. 28–42, 2011.

[17] B. Johnston, O. Vallis, and A. Kapur, "A Comparative User Study of Two Methods of Control on a Multi-Touch Surface for Musical Expression," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2012.

[18] D. Ellington, "If it sounds good, it IS good," http://en.wikiquote.org/wiki/Duke_Ellington, accessed: 2014-1-8.