A High-Level Review of Mappings in Musical iOS Applications

Thor Kell
IDMIL / CIRMMT, McGill University
thor.kell@mail.mcgill.ca

Marcelo M. Wanderley
IDMIL / CIRMMT, McGill University
marcelo.wanderley@mcgill.ca

ABSTRACT

We present a high-level review of mappings in musical iOS applications. All of the 38,750 music applications on the iOS store were reviewed and classified, using their title and descriptive text. Fifty music-making categories were found, along with sixteen non-music-making categories. Summarized mappings for each music-making category were defined and enumerated, by downloading and examining the screenshots for each app in each category. We present the total mappings, across all fifty categories, in terms of pitch, trigger, time, volume, and timbre. The sixteen non-music making categories were overviewed, but not discussed in detail. We also discuss broad trends and underutilized mappings, as well as suggesting areas for innovation. Finally, we provide public access to the created dataset, in order to further research around both iOS applications and text classification. This dataset includes the title, URL, and descriptive text for all applications, and is available both classified and unclassifed.

1. INTRODUCTION

Mobile touchscreen devices, such as the iPad and Nexus 10, continue to increase both in number and in power. As these devices become more powerful and as their sensing capacity increases, more and more new and novel applications are created for them. Music applications (apps) for these devices predominantly exist on the iOS platform [1]. The iOS app store has in excess of 38,000 apps in the Music category, ranging from complex synthesis engines to trivial radio applications.

The mapping of inputs to musical parameters in these apps is exceedingly important [2]. An iOS device's primary input mode is a simple touchscreen, but this can be mapped in essentially infinite ways. Likewise, the screen can present a variety of metaphors (keyboards, dials, strings, etc), in order to suggest a mapping to users. Fels has written about how this use of metaphor can inform a user's understanding of an app [3], in terms of the relationship between the mapping of musical parameters and the visual metaphor presented.

In addition to a touchscreen, iOS devices contain gyroscopes, accelerometers, a microphone, and one or more cameras. This range of controls enables a wide variety of musical applications with a wide variety of mappings. Hunt et al [4] have discussed how the mapping process can aid in relating these various control layers to the musical layer. Understanding how these controls are mapped to create music will allow app developers to design novel music making apps or focus their work into an extant category of apps.

A previous paper by the authors [5] reviewed the top 1,200 best-selling iOS applications, in terms of the interaction metaphor they presented to the user and the exact mappings that they use. That work defined ten main categories for iOS music apps, delineated their mappings, and discussed various outlying apps that did not fit into the ten standard categories. This paper extends that work by providing high-level classification for all 38,750 (as of January 28th, 2014) music apps, and a summary of mappings across all such classifications. This rather large scope was determined by the lack of access to smaller subsets of the data. The iTunes store lists the top 1,200 best-selling music apps, which was the data source used for the author's previous paper. The only other data source is the iTunes website, which provides data for all 38,750 apps.

Arner has also examined a small subset of iOS apps, with a focus on their gestural interaction and uses of multitouch [6]. Approaching the problem from the other direction, Tanaka et al [7] have provided a survey-based analysis of how mobile devices of all sorts are used musically. In terms of text classification, Zhu et al [8] have examined text and context-base machine-learning methods for automatically classifying apps, and Chen & Liu [9] have used similar techniques to attempt to model how popular various types of applications are.

This overview will provide large-scale data on how musical mappings and metaphors are defined on iOS. In addition to the ten categories defined in our previous work [5], we define forty new categories of musical apps and delineate their mappings. Moreover, in order to understand the iOS music ecosystem as a whole, we supply broad classifications for 'music' applications that do not allow the user to create music, such as radio and artist apps. We provide summaries of the total number of apps for each category, the total number of mappings across all apps, and offer thoughts on how to make use of this data when designing musical mappings and interfaces. We further create a dataset for future use, consisting of the title, URL, and descriptive text for each of the 38,750 apps, both with and without classification. This publicly available dataset will assist future studies of iOS applications, and text-based classification techniques.

2. METHOD

Data was downloaded from the web-facing iTunes website [1] , using a webcrawler built in Python with the Beautiful Soup [2] framework. In total, 38,750 apps were crawled. The app name, URL, and descriptive text were saved.

The analysis of this data had two goals. First, to find all apps that matched the ten known categories listed in the authors' previous paper [5]. These categories are: Piano, DJ, Digital Audio Workstation (DAW), MPC (A pad-based sampler/sequencer, based on the Akai MPC), Guitar, Drum Kit, Synthesizer, Sequencer, Karaoke, and Amp Sim. 'Radio' and 'Artist' apps were added to this list, due to the large numbers seen during cursory examinations of the data. The hope was to train a classifier to recognize these twelve known categories. Once apps that matched these categories were found, the second goal would be attempted: to discovered and count new categories, ideally using K-Means or similar processes.

In order to achieve the first goal, several supervised machine-learning methods were attempted, using both the TextBlob [3] and SciKit-Learn [4] [10] Python libraries. Training data was selected by examining apps that included the title of the category in their name or descriptive text, and then selecting apps that fit into the category in question. 25 to 50 apps were selected for each category. Both Bayesian classification and Support Vector Machines (SVM) were trained on this data. Using only the name of each application as a feature proved ineffective, as did using the entire descriptive text. Table 1 shows these poor results for both Bayes and SVM.

In terms of the Bayesian classifier, this poor performance is probably due to both the very high number of features and a high level of inconsistent dependencies among the dataset [11]. SVM, on the other hand, performs poorly when the number of features is much larger than the number of training samples [10]. In this case, each class had only 25 to 50 samples, with 8,882 features.

A whitelist of words important to each category was thus constructed. The whitelist can be seen in Table 2. This reduced the number of features to 114. As Table 1 again shows, this whitelist improved both the Bayesian and SVM classifications, using both the app name and descriptive text to 90% accuracy, on the test dataset.

As the SVM model using both the app name and the descriptive text was producing good results on the test data, the next step was to run the trained model on the entire dataset. This was done category by category, in order to remove classified apps with each iteration. The results from this, as seen in the first column of Table 3 seemed reasonable, at first blush. However, a manual examination of the remaining apps showed that many, especially Radio apps, were missed, suggesting that the models were overfitting to the test data. In hindsight, comparing the results between the columns of Table 3 show that some of the tested categories worked very well (Piano), while others did very, very badly (Radio).

These results were probably due to insufficiently trained models. Each category only had 25 to 50 apps to train on, and they were selected iteratively through the dataset, not at random. Radio apps, it would appear, are much more heterogeneous than the training data that was used.

In addition to attempting to classify known categories of applications, the second goal was to define new categories - ideally by clustering unclassfied apps together. This was first attempted on test data, and did not give good results. Using SciKit-Learn's K-Means algorithm on the twelve

[1] https://itunes.apple.com/us/genre/ios-music/id6011?mt=8
[2] http://www.crummy.com/software/BeautifulSoup/
[3] http://textblob.readthedocs.org/
[4] http://scikit-learn.org/

Table 1. Classification Methods

Method	Training Data	Whitelist	Accurary
Bayes	App Name	False	0.55
SVM	App Description	False	0.31
Bayes	App Name	True	0.77
Bayes	App Description	True	0.88
Bayes	Both	True	0.90
SVM	App Name	True	0.57
SVM	App Description	True	0.83
SVM	Both	True	0.90

Table 2. Whitelisted Words per Category

Category	Whitelisted Words
Radio	Radio, Station, FM
Artist	Upcoming, Latest, Bio, Connected, Official, Exclusive, Fan, News, Band, Musician, Composer
Piano	Piano, Keyboard, Chord, Scale, Key, Note, Theme, Hand, Harpsichord, MIDI
Drum	Drum, Drumming, Kit, Drummer, Snare, Kick, Crash, Ride, Cymbal, Percussion, Percussionist, Beat, Roll, Hihat, Hi-hat, Brush, Stick, Bongo, Conga, Taiko
Guitar	Guitar, String, Strum, Strumming, Vibrato, Tremolo, Electric, Tab, Twang, Mandolin, Steel, Pedal
Karaoke	Sing, Song, Karaoke, Star, Catalog, Share, Recording, Stage
DJ	Turntable, Deck, Scratch, Mix, Mixer, Mixing, Cue, Crossfader, Sync, Beatmatch
MPC	MPC, Pad, Sample, Production, Akai
Sequencer	Sequence, Sequencer, Groovebox, Beatbox, Step, MIDI, Pattern, Tempo, BPM, Machine
DAW	Loop, Record, Recording, Audio, Band, Mixer, Aux, Produce
Synth	Analog, Analogue, Engine, Filter, Fat, Envelop, Synth, LFO, Polyphonic, Monophonic, Sine, Square, Triangle
Amp Sim	Rig, Cabinet, Mic, Stomp, Amp, Tube

Table 3. Estimated Results vs. Actual Results

Category	Estimated	Actual
Radio	5288	10057
Piano	798	752
Drums	644	741
Karaoke	740	246
DAW	226	138
MPC / Sampler	220	136

Table 4. Clustering Results

Cluster	Number of Apps	Number of Categories
1	6	1
2	94	9
3	29	3
4	191	12
5	3	1
6	14	1
7	27	6
8	57	7
9	24	2
10	63	5
11	19	6
12	2	1

categories of test data was ineffective, even when using the whitelisted name and the whitelisted description. The apps both failed to cluster in groups around their categories, and failed to give correct numbers of apps per cluster. Table 4 shows the number of apps per cluster, and Table 5 shows the categories per cluster. Figures 1 and 2 shows the results of this clustering, with its dimensionality reduced via principle component analysis (PCA). As can be seen, each cluster does not contain only a single category. It was also hoped that PCA might allow for manual segmentation of each category. However, as can be seen by the PCA of the data in Figure 3, this was not possible: the categories are too intermingled to be able to draw useful segment boundaries.

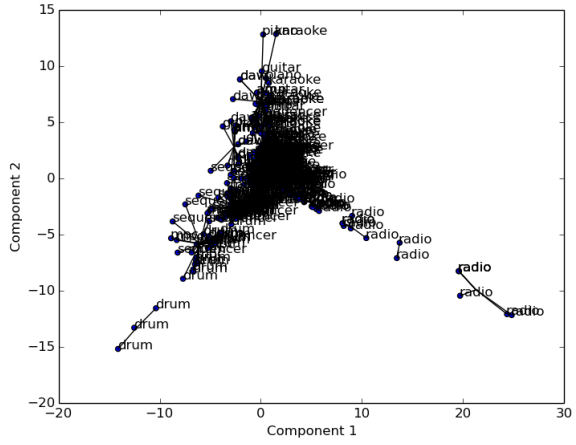

Figure 1. Labeled K-Means clusters.

Table 5. Clustering Breakdown

Cluster	Category Breakdown
1	Radio: 6.
2	Guitar: 29, Piano: 21, Karaoke: 14, DAW: 9, DJ: 8, Amp: 6, Synth: 3, Artist: 2, Sequencer: 2.
3	Drum: 15, MPC: 9, Sequencer: 5.
4	Artist: 67, Synth: 34, Piano: 25, DJ: 22, Guitar: 15, Sequencer: 13, Radio: 5, Amp: 3, MPC: 3, Drum: 2, DAW: 1, Karaoke: 1.
5	Drum: 3.
6	Radio: 14.
7	Karaoke 9, Amp: 8, Guitar: 5, Piano: 2, DAW: 2, Sequencer: 1.
8	Sequencer: 16, DJ 14, Synth: 10, MPC: 10, Drum: 4, DAW: 3.
9	Radio: 23, Artist: 1.
10	Drum: 26, MPC: 19, Sequencer: 13, Synth: 3, DAW: 2.
11	Amp: 9, DAW: 5, Piano: 2, DJ: 1, Guitar: 1, Karaoke 1.
12	Radio: 2.

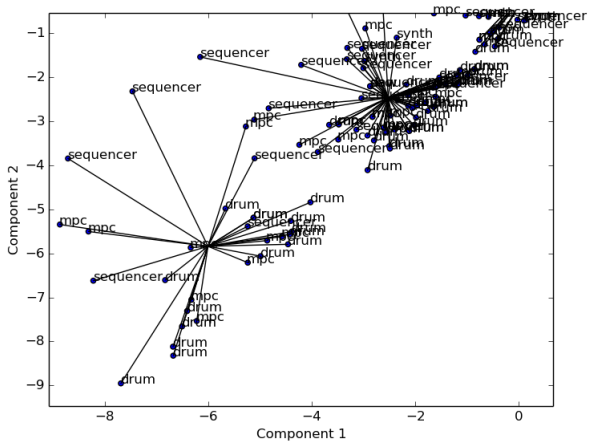

Figure 2. Labeled K-Means clusters, zoomed in.

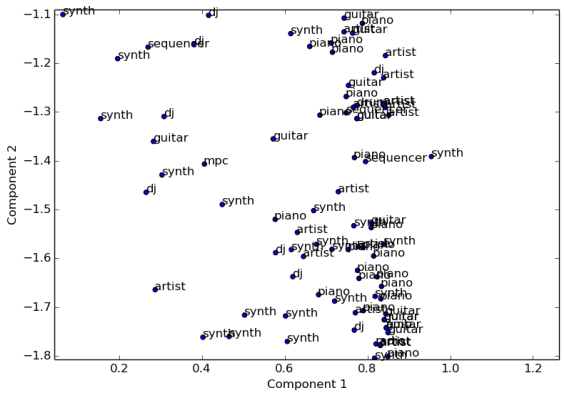

Figure 3. PCA data, zoomed in.

Given the difficulty clustering known data, perhaps due to K-Means' difficulty with clusters of varying shapes and densities (as seen in Figure 1), clustering the entire dataset was even less viable, especially as the total number of categories was not known and the whitelist would be ineffective on these unknown categories. This left us with a somewhat effective method of classifying known categories, and an ineffective method of finding new categories.

During this process it was discovered that, for a human, classifying each application based on the whitelisted name, the shortened app name in the URL, and the whitelisted descriptive text was simple to the point of trivial.

For example, the below three strings strongly suggests an Artist application:

- " ", "amon-amarth-mobile-backstage", "official fan exclusive fan"

Likewise, these strings suggest a Amp Sim application:

- " ", "ampkit", "amp guitar amp electric guitar amp guitar amp mic pedal guitar recording share guitar fan"

In contrast, the below descriptive text suggests a Synth, but the name suggests a novel application.

- " ", "anckorage-spring", "audio connected keyboard midi engine midi midi midi midi"

In this case, the entire descriptive text was checked in a second step, and the application was then correctly classified as a Synth. The descriptive text is excerpted below:

"*Anckorage Spring is a physical modelling audio synthesiser based on the simulation of a set of connected mass-spring, integrating non-linearities, fluid and static friction, mechanical limits, gravity and bouncing. It is designed to be controlled by a continuous controller (like the Haken Continuum www.hakenaudio.com)...*"

Using this method it was found that 500 apps, with the use of a Python script to skip through them, could be shunted into the initial twelve categories in as little as fifteen minutes, giving a 'mere' 39 hours to complete the task of manual classification. This is, of course, not to say that text-based machine learning is ineffective. Zhu et al [8] have made use of text data to successfully classify apps (though only 680 of them) of them, and Whitman et al [12] have used natural language processing and text-based machine learning on community metadata as a key component of their work in classifying music. The present paper, however, was simply looking to classify a large number applications with a high degree of accuracy, not investigate machine learning techniques. Manual classification also had the advantage of being completable in a known, though long, amount of time, whereas automatic classification presented a very open-ended problem. Furthermore, good, manually classified data could also be used as ground truth data for future investigations of text-based classifications and of the iOS app store.

It was thus decided to manually classify the data. After the first 10,000 apps had been classified by hand, two heuristics were added to speed the process: apps that had 'radio' in the name were immediately defined as Radio applications, and apps that had whitelisted descriptive text of 'official update new connect' were immediately defined as Artist applications.

During this process, many applications could not be fit into the twelve known categories. Those were logged separately, and then examine with the full text of their names and descriptive text. Out of *those* apps, new categories (accordion apps, for example) were defined, based on the descriptive text. Totally novel apps were again logged separately. Due to time constraints, the 481 novel applications were not examined in detail.

Once this two-tiered process was complete, each category was counted. In order to define the mappings for each category, the screenshots for each app in each category were downloaded and examined, again using a web crawler. In some few cases, (the Karinding, for example) videos of apps were examined in order to define the mappings. Only the general mappings for each category were defined. For example, if all but one Xylophone app maps pitch to the colours of the rainbow, the Xylophone category as a whole will be assigned this mapping of pitch to colour.

2.1 Verification

This classification process is not a perfect one. Even ignoring typos, this sort of fast human classification is prone to errors. In order to verify the quality of this method, 100 randomly selected apps were examined using their full name and full descriptive text: 94 were correctly classified, and 6 were incorrect. Then, 100 more randomly selected apps were tested, ignoring apps from the Radio, Artist, Media, and Non-English categories. Once again, 94 were classified correctly, and 6 were in error. It must also be noted that the Media, Educational, and Tool categories contain many interesting apps that are outside the scope of this paper. More in-depth app reviews would be well served to begin with these categorizations - to say nothing of the various applications in languages other than English.

3. RESULTS

3.1 Music-Making Applications

Applications that allow the user to produce music are, of course, the focus of this paper. Table 6 shows the number of applications in each category for these music-making applications. Each of these categories also include applications with similar layouts. The 'Guitar' category, for example, also includes lute, banjos, mandolins, ukeleles, and so on. Categories that may require further explanation are listed below:

- Ball Sim - Apps that trigger sounds via a physics simulation of balls or other objects moving around.
- Chord Sequencer - Apps that allow the user to sequence symbolic representations of chords, either in guitar tablature or text / numeric format
- Dulcimer - Western dulcimers, hammered dulcimers, and so on.

- Gamelan - Indonesian Gamelan instruments, include bells, gongs, and metallophones.
- Guqin - The guqin is an ancient Chinese zither, with angled strings.
- Hang - The hang is a modern pitched percussion instrument, similar to the steelpan.
- Kalimba - The kalimba, or thumb piano, is an African plucked percussion instrument.
- Karinding - The karinding is an Indonesian mouth harp.
- Looper - Apps that loop audio recorded by the user, rather than sequencing samples.
- Melodica - A reed-based wind instrument, with a small keyboard for selecting pitches.
- MIDI / OSC - Apps that output MIDI or OSC, to control other devices. As these apps vary wildly, their mappings are not included in the final count.
- Ocarina - The ocarina, a simple wind instrument, occurs in many cultures, but is perhaps most famous for its role in the 'Ocarina of Time' video game.
- Ondes Martenot - An early 20th century electronic instrument, featuring both ribbon and keyboard control of pitch.
- Steelpan - A pitched percussion instrument, originally from Trinidad & Tobago.
- Vuvuzela - A trombone-like South African instrument instrument, with a single pitch.
- Zither - Eastern zithers, including the guzheng, jentreng, qanun and gayageum.

3.2 Non-Music-Making Applications

Broad categories were defined for apps that do not make music. These make up the majority of the music section of the app store. Table 7 shows the numbers of apps per category, and each category is defined below. This section also includes 'Junk' apps that are not music apps at all, and apps that were unclassifiable due to their descriptive text not being in English.

- Radio - Apps for a particular radio station, that assemble many radio stations, and so on.
- Media - Apps that deliver non-auditory media, allow for the playback of auditory media in a non-musical way, including soundboards, 'best songs' for a genre, and so on.
- Artist - Apps for promoting a particular artist, a group of artists, a festival, a recording studio, and so on.
- Non-English - Apps with descriptive text not in English, and thus not reviewable in this paper.

Table 6. Number of Applications per Category, Musical Applications

Rank	Category	Number of Apps
1	Piano	752
2	Drum	741
3	Sequencer	606
4	Novel	481
5	Guitar	385
6	Synth	277
7	Karaoke	246
8	Effect	149
9	DAW	138
10	MPC / Sampler	136
11	Xylophone	132
12	DJ	119
13	Accordion	74
14	Band	67
15	Flute	67
16	MIDI / OSC	65
17	Harp	47
18	Amp Sim	45
19	Bells	40
20	Looper	36
21	Chord Sequencer	36
22	Bagpipes	33
23	Notation	33
24	Steelpan	33
25	Violin	31
26	Organ	25
27	Gamelan	24
28	Trumpet	23
29	Ball Sim	23
30	Zither	17
31	Harmonica	16
32	Kalimba	15
33	Clarinet	13
34	Water Glasses	11
35	Trombone	9
36	Dulcimer	9
37	Singing Bowl	9
38	Cello	9
39	Saxophone	8
40	Horn	7
41	Melodica	6
42	Vuvuzela	5
43	Ocarina	4
44	Washboard	4
45	Conductor	3
46	Hang	3
47	Pan Pipes	2
48	Ondes Martenot	2
49	Guqin	1
50	Karinding	1

- Educational - Apps for teaching an instrument, a theoretical concept, and so on.
- Tool - Apps for accomplishing music related tasks, including tuners, spectrum analyzers, and so on.
- Games - Apps for playing games about music or musicians
- Junk - Apps that have been mislabled and are not music apps.
- Remote - Apps for remote control of non-musical audio systems, such as home theatre systems.
- Discovery - Apps for finding new music, new playlists, and so on.
- Christmas - Apps about Christmas.
- Print - Apps for a particular print magazine, or emulating a print magazine.
- Recorder - Apps for recording sound.
- Social - Apps for communicating about music on Twitter or other social media platforms.
- Fitness - Apps for controlling music while working out.
- Fingerprint - Apps for fingerprinting audio.

Table 7. Number of Applications per Category, Non-Musical Applications

Category	Number of Apps
Radio	10057
Media	7416
Artist	7161
Non-English	2806
Educational	2052
Tool	1406
Games	905
Junk	354
Remote	334
Discovery	272
Christmas	268
Print	249
Social	220
Recorder	154
Fitness	50
Fingerprinter	20

4. MAPPINGS

Table 8 shows the total mappings, across all categories. The mapping definitions used are hopefully self-explanatory. In terms of those that are less clear, a 'Known Layout' refers to an app that matches the visual layout of a real instrument, and maps some parameter based on this in a way that does not fit into any other category. For example, a drum application maps timbre based on a Known Layout - that of a drum kit.

'Force' here means methods of determining how hard the user is tapping the device, often by polling the accelerometer or the microphone. A 'Gesture' indicates any motion more complex than a touch, typically a dragging or circular movement. When applied to the Volume parameter, this indicates that the speed of the gesture directly varies

Table 8. Mappings for Musical Applications

Mapping	Pitch	Trigger	Time	Volume	Timbre
Horizontal: Left-to-Right	2142	0	1559	138	141
Horizontal: Right-to-Left	11	0	0	128	0
Horizontal: Center-to-Edge	15	0	0	0	0
Vertical: Top-to-Bottom	35	0	152	0	0
Vertical: Bottom-to-Top	1307	0	0	1483	1358
Diagonal: Bottom-Left-to-Top-Right	38	0	0	0	0
Rotational: Clockwise	0	0	9	0	0
Circular	33	0	0	0	0
Radial: Edge-to-Center	33	0	0	0	0
Grid	43	0	0	0	0
Vertical Size	105	0	0	0	0
Overall Size	33	0	0	0	0
Colour	78	0	0	0	12
Symbolic / Text	69	0	33	33	33
Continuous	190	0	612	1630	1498
Discrete	3931	0	1042	33	33
Playback	0	1104	0	0	0
Toggle	8	272	0	9	1207
Touch	0	3096	0	24	25
Gesture	0	86	0	10	2
Shake / Swing	0	20	0	20	0
Known Layout	167	0	0	0	746
Microphone Input	0	282	0	36	0
Audio Input	0	194	0	0	0
Force	0	0	0	81	0
Physics	12	23	0	23	18

the volume of the sound. Finally, vertical mappings refer to the gesture used, not the metaphor presented: many apps present the user with rotary knobs which are actually controlled by vertical motion. This paper has used the mapping throughout, rather than the metaphor.

4.1 Pitch

Pitch is dominated by keyboard-like, left-to-right or bottom-to-top mappings. Discrete pitches are likewise much more prevalent than continuous pitch. Some few apps increase pitch from top to bottom (zithers, for example), and even fewer increase pitch from right to left (trombones and pan pipes, in particular). Outside of these linear mappings, the next most popular mapping for pitch is the 'Known Layout' of wind instruments, which is usually abstracted to a set of 3-6 buttons that additively modify the pitch: pressing two buttons together gives a new pitch, rather than two pitches.

Mappings of pitch to colour are not uncommon, but a single dominant mapping of colour to pitch was not found. Likewise, mappings of pitch to size exist, but are always secondary to some other mapping (horizontal in the case of xylophones, and circular in the case of steelpans). Symbolic and text mappings are entirely based on various Western systems, including the sharps / flats of traditional staff notation, and various representations of chords (V^6, Dm7, etc).

4.2 Trigger

Unsurprisingly, given that the primary interaction method on iOS devices is a touchscreen, mapping one touch to one sonic event is by far the most popular method for triggering sounds. Toggles are also popular, along with events or states that are often controlled by toggles, such as the playback of a sequencer or audio input from another device. Gestural mappings are not common, and mostly use simple movements: a circular motion to trigger a drum roll instead of a single drum hit, for instance. Making use of the device's other sensors, via a swing or a shake of the device, is not common. No applications were found that triggered sounds via a gesture made by moving the device itself - such mappings, may, however, exist in one of the unexamined Novel apps.

4.3 Time

Time moves from left to right, and from top to bottom. Discrete time is slightly more prevalent than continuous time. Some very few apps map time rotationally, clockwise. Even in Notation apps, where time & rhythm is represented symbolically, the flow of time is from left to right.

4.4 Volume

Volume is dominated by vertical mappings, usually presented continuously. Some apps make use of force-based or shake/swing methods for determining volume. These, along with wind instrument apps that base volume on the input from the microphone, are the closest to 'real' acoustic instruments. An even smaller but more interesting mapping is that of touch / gesture to volume. For instance, some Gamelan apps allow for virtual bars to be muted by touching them in particular locations, and some Singing Bowl and Water Glass apps play louder sounds based on the speed of the triggering circular gesture.

4.5 Timbre

Like volume, timbre is mostly controlled vertically and continuously. Many apps use toggles to change between preset timbres (in piano apps, for instance), and many use Known Layouts to control the timbre of the sound played - drum kits are a prime example of this. Other timbral controls are much more rare. Surprisingly, colour is only used rarely for timbral control. However, like Volume, a very small number of apps use additional touches to control timbre. To continue the Gamelan example, some apps also allow for a muted timbre to be played if a virtual bar is touched before triggering it.

4.6 Summary

From Table 8 and the above paragraphs, it is clear that most apps use typical mappings: pitch from left to right, sounds triggered by touch, and volume / timbre controlled by vertical faders. Most of these mappings do not take advantage of sensors outside of simple touch and location. Complex gestures, microphone input, and shaking/swinging the device are used to control parameters from pitch to volume to timbre for a small number of applications, but are in general ignored. Likewise, most apps separate the control of each parameter, mapping them to different controls and in different ways. Integral mappings are almost entirely ignored. The 481 apps in the Novel category have not been examined, however. They would almost certainly contribute to making Table 8 more varied.

5. DATASET

In order to further research around iOS music apps, we have made the dataset and Python scripts used to examine the data publicly available The data (consisting of the name, URL, and descriptive text for each app) is provided, classified and unclassified, in order to allow for a wide variety of machine-learning approaches and/or brute-force approaches. The complete collection of data and code can be found at idmil.org/projects/ios_mappings.

6. CONCLUSION

We have provided a high-level review of all music apps on the iOS app store as of January 2014. This builds upon the authors' previous paper [5], which provided an in-depth look at the most popular iOS music apps. We have also provided the raw text data, classified and unclassified, for future research around text-based machine learning, app classification and more.

In the review itself, we discovered many new iOS instruments that represent extant, acoustic instruments, ranging

from bagpipes to zithers. We also discovered a smaller number of new, purely electronic instrument categories, including loopers, chord sequencers, and bouncing-ball apps. We provided a high-level overview of mappings for each of these categories: this data will be useful both for understanding the overall iOS application ecosystem and the musical subset thereof. More importantly, this data can be used to understand how musical parameters are mapped on touchscreen devices, and thus influence how new musical applications are designed.

To be specific, we can see the dominance of simple mappings: pitch generally moving horizontally and discretely, volume and timbre moving vertically and continuously, and time moving from left to right. Although many applications make use of more complex mappings and more complex inputs, they are in a minority. This is a potentially rich area for innovation: one can easily imagine apps that use the microphone, accelerometer, and gyroscopes of iOS devices in new and interesting ways. Likewise, integral mappings for timbre and volume or non-traditional representations of pitch / time could both lead to interesting and innovative apps for making music.

Further work after such a high-level review is legion. A detailed examination of the Media, Educational, and Tool categories should be done, and would no doubt reveal sundry new ways to map musical parameters, in tuning apps, how-to-play apps, and so on. Indeed, a deep dive into each of the main categories described above could provided further detail about how each category maps parameters. Likewise, the 481 Novel applications should be examined in detail.

Touchscreen devices, and iOS in particular, are here to stay, and the ability of these platforms to create music at all levels of sophistication is only going to grow. This report has furthered the task of understanding how musicians and developers are dealing with the mapping of music parameters, and will hopefully result in a deeper understanding of the mapping process and its outcomes.

Acknowledgments

Special thanks to Vanessa Yaremchuk for her knowledge of machine-learning methods.

7. REFERENCES

[1] E. van Buskirk, "Developer Explains Why Android Sucks for Some Audio App," http://evolver.fm/2012/05/23/developer-explains-why-android-sucks-for-some-audio-apps/, 05 2012, accessed: 24/02/2013.

[2] A. Hunt, M. M. Wanderley, and M. Paradis, "The importance of parameter mapping in electronic instrument design," *Journal of New Music Research*, vol. 32, no. 4, pp. 429–440, 2003.

[3] S. Fels, A. Gadd, and A. Mulder, "Mapping transparency through metaphor: towards more expressive musical instruments," *Organised Sound*, vol. 7, no. 2, pp. 109–126, 2002.

[4] A. Hunt, M. Wanderley, and R. Kirk, "Towards a model for instrumental mapping in expert musical interaction," in *Proceedings of the 2000 International Computer Music Conference*, 2000, pp. 209–212.

[5] T. Kell and M. M. Wanderley, "A quantitative review of mappings in musical ios applications," in *Proceedings of the Sound and Music Computer Conference 2013*, 2013, pp. 473–480.

[6] N. F. Arner, "Investigation of the use of Multi-Touch Gestures in Music Interaction," Master's thesis, University of York, 2013.

[7] A. Tanaka, A. Parkinson, Z. Settel, and K. Tahiroglu, "Survey and thematic analysis approach as input to the design of mobile music guis," *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2012.

[8] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian, "Exploiting enriched contextual information for mobile app classification," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 1617–1621.

[9] M. Chen and X. Liu, "Predicting popularity of online distributed applications: itunes app store case analysis," in *Proceedings of the 2011 iConference*. ACM, 2011, pp. 661–663.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] H. Zhang, "The optimality of naive bayes," in *Proceedings of the FLAIRS Conference*, vol. 1, no. 2, 2004, pp. 3–9.

[12] B. Whitman and S. Lawrence, "Inferring descriptions and similarity for music from community metadata," in *Proceedings of the 2002 International Computer Music Conference*. Citeseer, 2002, pp. 591–598.