Intelligent Voice Recorder

Dhiraj Gurkhe
Arizona State University
Email: dgurkhe@asu.edu
ID: 1209305002

Sriram Vellangallor Subramanian
Arizona State University
Email: svellang@asu.edu
ID: 1209270383

Karumpudi Ramakrishna Reddy
Arizona State University
Email: rkreddy@asu.edu
ID: 1209319770

***Abstract*—Traditional audio recording applications lack the power to express events that may have interested the user while he was recording an audio, along the timeline. Our project aims to resolve this problem by adding time-synchronized metadata to audio recordings, by providing an interface to users to attach textual tags, called hooks, while they are recording. This allows users not only to playback the audio in a recording like existing systems, but also to display these hooks in sequence, enabling him to recall key events that he annotated while he was recording.**

Keywords*—*Voice Recording, Media Playback, Waveform, Android, GPS

I. Introduction

Our application— the Intelligent Voice Recorder on the Android platform— enables users to annotate the audio while recording, on the fly using keyboard. This app provides two functional modes: the record mode and the playback mode. In the record mode, our app allows recording an audio such as a class lecture and to simultaneously insert hooks at desired time-frames. In the playback mode, the user-interface is presented as a waveform of the recorded audio along a time-line. Annotations would be displayed in sequence while the audio is being played. Finally, we provide basic file organization functions by which user can organize his notes based on location fetched from the GPS, tags, start time of recording or even his own preferences.

II. Implementation

A. Voice Recording

The voice recorder is built using the android audio capture library[1]. The following bit of code is used to start recording and to save it later.

```
private void startRecording() {
    mRecorder = new MediaRecorder();
    mRecorder
    .setAudioSource(MediaRecorder
    .AudioSource.MIC);
    mRecorder
    .setOutputFormat(MediaRecorder
    .OutputFormat.THREE_GPP);
    mRecorder
    .setOutputFile(mFileName);
    mRecorder
    .setAudioEncoder(MediaRecorder
    .AudioEncoder.AMR_NB);

private void stopRecording() {
    mRecorder.stop();
    mRecorder.release();
    mRecorder = null;
}
```

To ensure the recording is not stopped when the app is pushed the background, all the recording activities are done in the background. While recording an intelligent note, the user can view the waveform of the recording real time. The communication with the service and activity is done through MPI. The ability to hook text is added where user can add text while the recording is going on. These hooks are stored in a file in a timed series relative to the start of the recording. Behind the scene important information like the location, and time is recorded. The locations is recorder at the start and at the end of the recording by using androids GPS location service[2]. The whole module is thoroughly tested, and also provides a custom build visualizer to show ripple effect based on amplitude while recording.

B. Playback

There are two ways to launch the playback mode— from the record screen after a note has been recorded, or from the folder organization screen once the user taps on a recording. The name of the recording is fetched from the previous screen using intent parameters and displayed on the title bar. The user interface is adapted based on the time (day, twilight and night mode, respectively) when the intelligent note was recorded so that user gains quick visual insight. Media playback is implemented using the MediaPlayer library[3].

1) Waveform visualization: The amplitude of the recording is plotted as a function of time in a bar chart by using the MPAndroidChart library[4]. The color code is configured for each UI mode regular amplitude values are plotted in one color.
2) Visual spots: The bar chart used in waveform visualization is reused for providing visual spots that indicate those time values for which a hook was inserted in the intelligent note while recording. These spots are plotted with a value equal to 1.5 times the maximum amplitude in the sampling list, in a high-contrast color. The high contrast not only serves as a visual cue to the user indicating a hook, but also eliminates ambiguity that it is an amplitude function. A basic synchronization issue was resolved— since there is no guarantee that a hook was inserted precisely at the time when an amplitude value was sampled during the recording phase, the hook's time instance is approximated to the nearest time instance of a sampled amplitude.

3) Playback controls: The play button starts playback and toggles to a pause button once it is pressed. During playback, a background thread called UpdateSongTime performs the following UI updates:
   a) Update the media seek bar and the bar chart: In addition to the seek bar, a translucent overlay color is added on the bar chart to enhance visual cues to the user on the current playback position. By adding a gesture recognizer to the bar chart, the app enables the user to seek to a location of the recording not only by using the seek bar but also by dragging along the bar chart. The setter methods of attributes in the PlayBack class are designed such that updating a single attribute updates all the desired UI elements (seekbar, graph chart and hook label) simultaneously[5]. Whenever the update thread is called, it searches for a hook in the hooks map few milliseconds ahead and behind the current playback time. If one is found, it displays it on the textview. Such a fuzzy look up is necessary to ensure that the nearest available hook is displayed in case a hook was not inserted coinciding with an instant when the amplitude was sampled, while recording the intelligent note.
   b) Rewind/forward functions: These buttons enable user to seek to the previous or the next available hook in the intelligent note. If either only one hook exists prior to the current hook or if more than 3.5 seconds of playback has completed from the current hook, tapping rewind button will seek the playback to the beginning of current hook. On the other hand, if there are sufficient hooks to seek back and user wishes to seek to the beginning of previous hook after completing 3.5 seconds of current hook's playback, he has to double tap the rewind button (on single tap it will seek to beginning of current hook). The double tap was implemented by tracking the time difference between two successive button taps, because Android affords handlers only for single tap events. Forward seek is straightforward— if user taps on forward button, player will always seek to the beginning of the next hook. Both these buttons are enabled only if the MediaPlayer object is currently in the playing state and there is a hook available in the respective direction of time frame.

C. Folder Organization

Map view— If multiple recordings in same (lat,long) then it should indicate an aggregate count. In such case take him to a folderview and filterdisplay all those recordings from that location. From there he can tap and go to a particular location. Use UI polymorphism so that if there is only one recording in one place then it should directly open the file if he taps on the place marker. One more improvement in map view: If a press reporter was recording while traveling, plot the route of his journey on the map to indicate a file instead of a single place marker. Use accuracy and approximation techniques to distinguish between movement of person within few feet versus an actual travel from city A to city B.

D. Database and Storage

1) Database: We used sqlite database for storing information for each recording. The table had the following schema structure:
   a) created-at :DATETIME
   b) Filename :Text
   c) startTime :Real
   d) longitudeStart : Real
   e) latitudeStart : Real
   f) StartCity : Text
   g) longitudeEnd : Real
   h) latitudeEnd : Real
   i) EndCity : Text
   j) Time : float

   The data was required for various module like folder orgainization, UI adaptation, download.
2) Storage: For every recording apart from the above data which was stored in database, we also had to store the following data files in the storage.
   a) rec.3gp : The actual sound data which was recorded
   b) hook$.txt :Text file having all the timestamps for all the hooked text for the recording relative to the start time of the recording.
   c) wav˜.txt : This was the time series of amplitude data collected during the recording. Which was used for better visualization during playback.

   To bring an extra layer of abstraction to the data we compressed all these file into a single file. This single file was given our own extension of .drs, which disables users from using Android's file managers to probe into our package.

III. Status Table

TABLE I: Status table

Serial No	Task	Assignee	Status
1	Voice recorder	Dhiraj	Completed
2	GPS tracker	Dhiraj	Completed
3	Note hooking	Dhiraj	Completed
4	Waveform timeline	Sriram	Completed
5	UI adaptation	Sriram (RK)	Completed
6	Playback mode	Sriram	Completed
7	Playback controls	Sriram	Completed
8	Folder organization I	RK	Completed
9	Folder organization II	RK	Completed
10	Folder organization III	RK	Completed
11	Download audio	Dhiraj	Completed
12	Visual spots	Sriram	Completed
13	Registration	RK	Completed
14	Background service	Dhiraj	Completed
15	Seeking to hooks	Sriram	Completed

IV. Screen shots

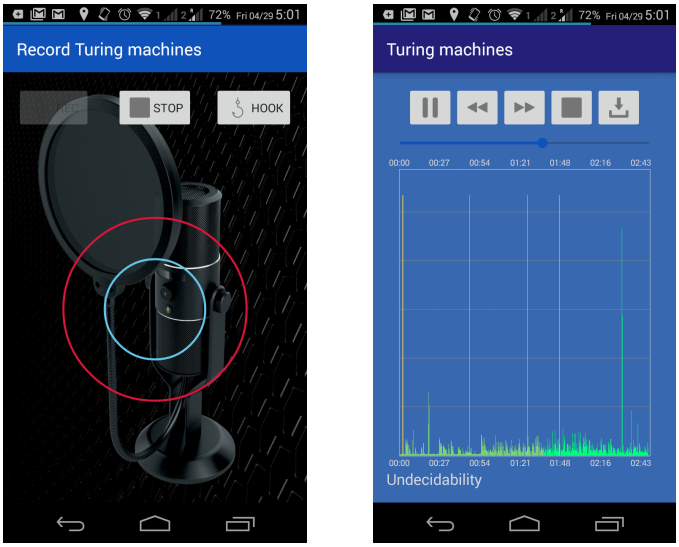

Fig. 1: Recording and Playback mode

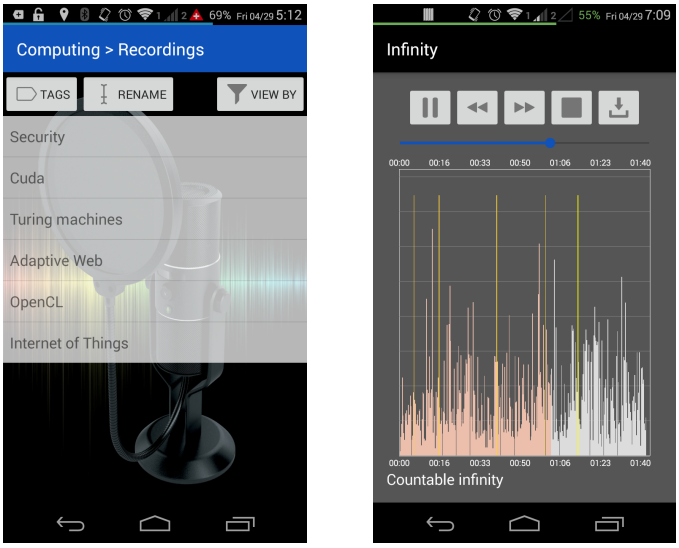

Fig. 2: Folder View

V. Conclusion

Throughout the course of the project we learned a lot of things. To start with was working with android and android studio environment. Understanding the sound data, how to work with it, and how to add extra meta data were some interesting challenges we faced and overcame. Location and SQL were two modules which were difficult to deal with to at the beginning but at the end we were able to use them to our full benefits. Building adaptive User Interface was fun, although we didn't use advance graphic libraries learning working with primitive ones was rewarding.

A. *Future Work:*

These are few things we thing will improve this application future:

1) UI adaptation— Use gradient color along the span of recording in case of long lectures (if it starts at day, starting width portion of screen will be blue. If it ends next day, 2nd section will be dark blue indicating twilight, 3rd section will be black indicating night, 4th section will be light blue indicating next day).
2) Map view— If multiple recordings in same (lat,long) then it should indicate an aggregate count. Plot the route of journey on the map to indicate a file instead of a single place marker while recording. Use accuracy and approximation techniques to distinguish between movement of person within few feet versus an actual travel from city A to city B.
3) Voice to text facility while the recording is on so that user can get a stream of keywords which he can select to add to hooks. Allowing scribbling while adding hooks

References

[1] http://developer.android.com/guide/topics/media/audio-capture.html
[2] http://developer.android.com/guide/topics/location/index.html
[3] http://developer.android.com/reference/android/media/MediaPlayer.html
[4] https://github.com/PhilJay/MPAndroidChart
[5] http://theopenacademy.com/content/lecture-6-polymorphism-controllers-ui