USER GUIDE FOR THREE MICROPHONE DIRECTION OF ARRIVAL ESTIMATION ON ANDROID SMARTPHONE FOR HEARING AID APPLICATIONS

Serkan Tokgoz, Anton Kovalyov, Issa Panahi

STATISTICAL SIGNAL PROCESSING LABORATORY (SSPRL)
UNIVERSITY OF TEXAS AT DALLAS

APRIL 2020

Table of Contents

INTRODUCTION

The 'Three Microphone Direction of Arrival (DOA) Estimation' app is designed for estimation of the speaker direction for hearing aid users in real time direction on a graphical user interface (GUI). The application is trained to perform in noisy conditions as well. The contents of this user guide gives you the steps to implement the 'Three Microphone Direction of Arrival (DOA) Estimation' algorithms on Android devices (that have two microphones) and the steps to be followed after installing the app on the smartphone. This app will be an open source and portable research platform for hearing improvement studies.

This user guide covers the software tools required for implementing the algorithm, how to run C codes on Android devices and usage of other tools that are quite helpful in creating audio apps for audio playback in real time.

The MATLAB and C codes used for the 'Three Microphone Direction of Arrival (DOA) Estimation' algorithm are made available publicly on the following website: http://www.utdallas.edu/ssprl/hearing-aid-project/. The codes can be accessed and used with proper consent of the author for further improvements in research activities related to hearing aids. The screenshot of the first look of our app is as shown in Figure 1.
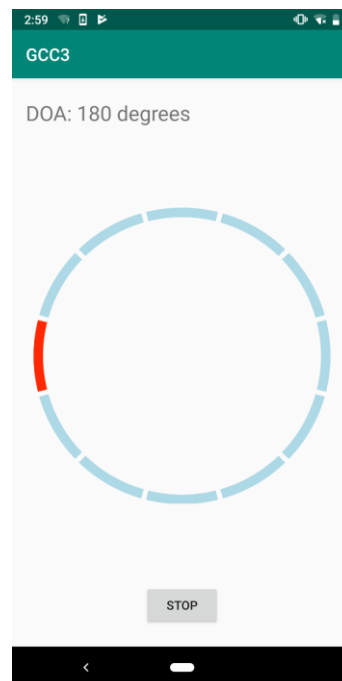

Figure 1

1. SOFTWARE TOOLS

Android is an open-source operating system developed by Google for mobile phones and tablets. The Android apps are usually coded in Java. In this section, it is shown how to set up the Android Studio IDE (Integrated Development Environment) for developing Android apps.

Android Studio IDE requires either Windows Operating System or Apple Operating System. Android Studio IDE can directly build and upload source codes into Android smartphone or generate a APK file which can be downloaded and installed on the Android smartphone.

To download the latest version of Android Studio

1. Open the Android Studio website to download.
   (https://developer.android.com/studio/index.html)
2. Click Download Android Studio Button (Figure 2).
3. Install the execution file after download.

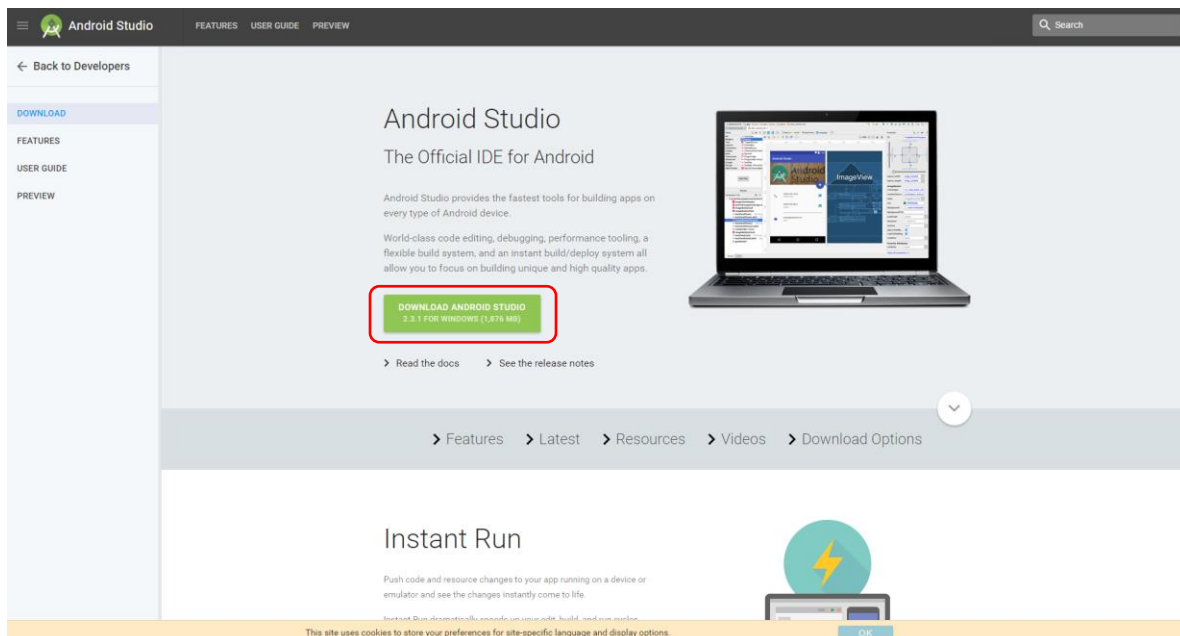Figure 2

2. BUILD AN ANDROID APP

2.1 Programming Language

For creating Android apps, Java is used to create the required shell. The Java Development Kit (JDK) needs to be firstly installed on your computer. This link contains the latest version of JDK:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

2.2 Creating Android Apps

After installations of Android Studio and JDK completed, android apps can be created using Android Studio.

1. Open Android Studio.
2. Under the "Quick Start" menu, select "Start a new Android Studio project." (Figure 3)
3. On the "Create New Project" window that opens, name your project "HelloWorld".(Figure 4)
4. If you choose to, set the company name as desired*.
5. Note where the project file location is and change it if desired.
6. Click "Next."
7. Make sure on that "Phone and Tablet" is the only box that is checked. (Figure 4)
8. If you are planning to test the app on your phone, make sure the minimum SDK is below your phone's operating system level.
9. Click "Next."
10. Select "Blank Activity." (Figure 4)
11. Click "Next."
12. Leave all of the Activity name fields as they are. (Figure 4)
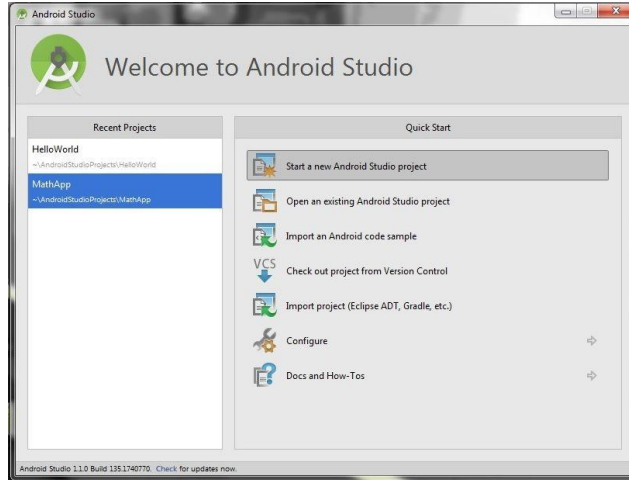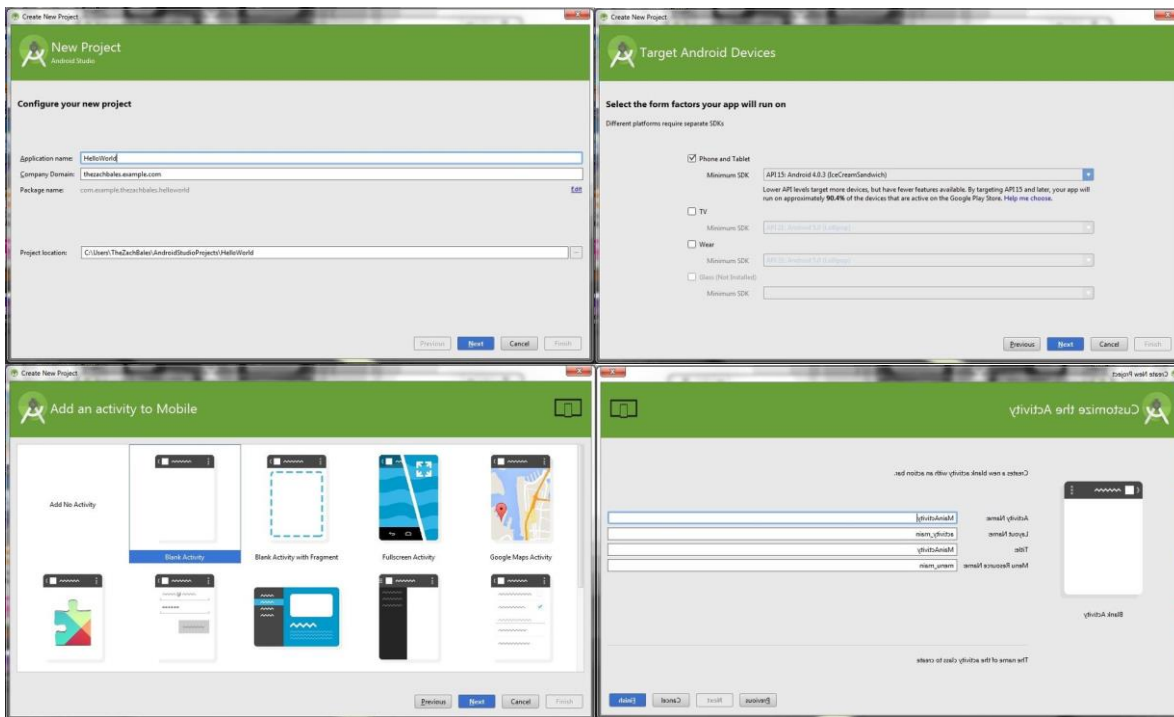13. Click "Finish."

Figure 3

Figure 4

2.3 Adding C File

Android apps are typically written in Java, with its elegant object-oriented design. However, at times, you need to overcome the limitations of Java, such as memory management and performance, by programming directly into Android native interface. Android provides Native Development Kit (NDK) to support native development in C/C++, besides the Android Software Development Kit (Android SDK) which supports Java.

2.3.1   Installing the Native Development Kit (NDK)

1. Menu "Tools" > "Android" > "SDK Manager" (Figure 5)
2. Select tab "SDK Tools"
3. Check "Android NDK"[or "NDK"] if it is not checked
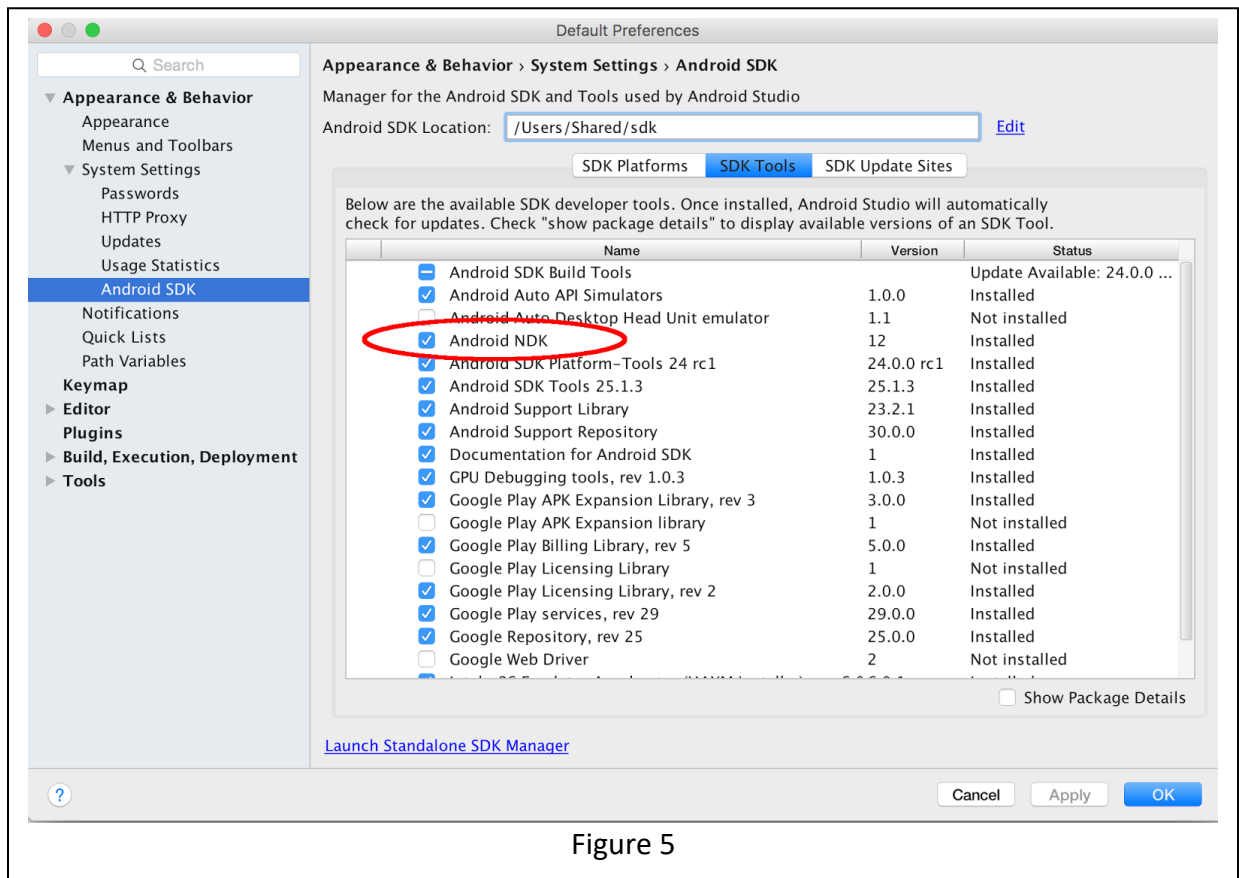4. Sync and re-build the project.

Figure 5

2.3.2 Writing a Hello-World Android NDK Program

Creating a new project with support for native code is similar to creating any other Android Studio project, but there are a few additional steps:

1. In the Configure your new project section of the wizard, check the Include C++ Support checkbox.

2. Click Next.

3. Complete all other fields and the next few sections of the wizard as normal.

4. In the Customize C++ Support section of the wizard, you can customize your project with the following options:

   o C++ Standard: use the drop-down list to select which standardization of C++ you want to use. Selecting Toolchain Default uses the default CMake setting.

   o Exceptions Support: check this box if you want to enable support for C++ exception handling. If enabled, Android Studio adds the -fexceptionsflag to cppFlags in your module-level build.gradle file, which Gradle passes to CMake.

   o Runtime Type Information Support: check this box if you want support for RTTI. If enabled, Android Studio adds the -frtti flag to cppFlags in your module-level build.gradle file, which Gradle passes to CMake.

5. Click Finish.

After Android Studio finishes creating your new project, open the Project pane from the left side of the IDE and select the Android view. As shown in figure 6, Android Studio adds the cpp and External Build Files groups:
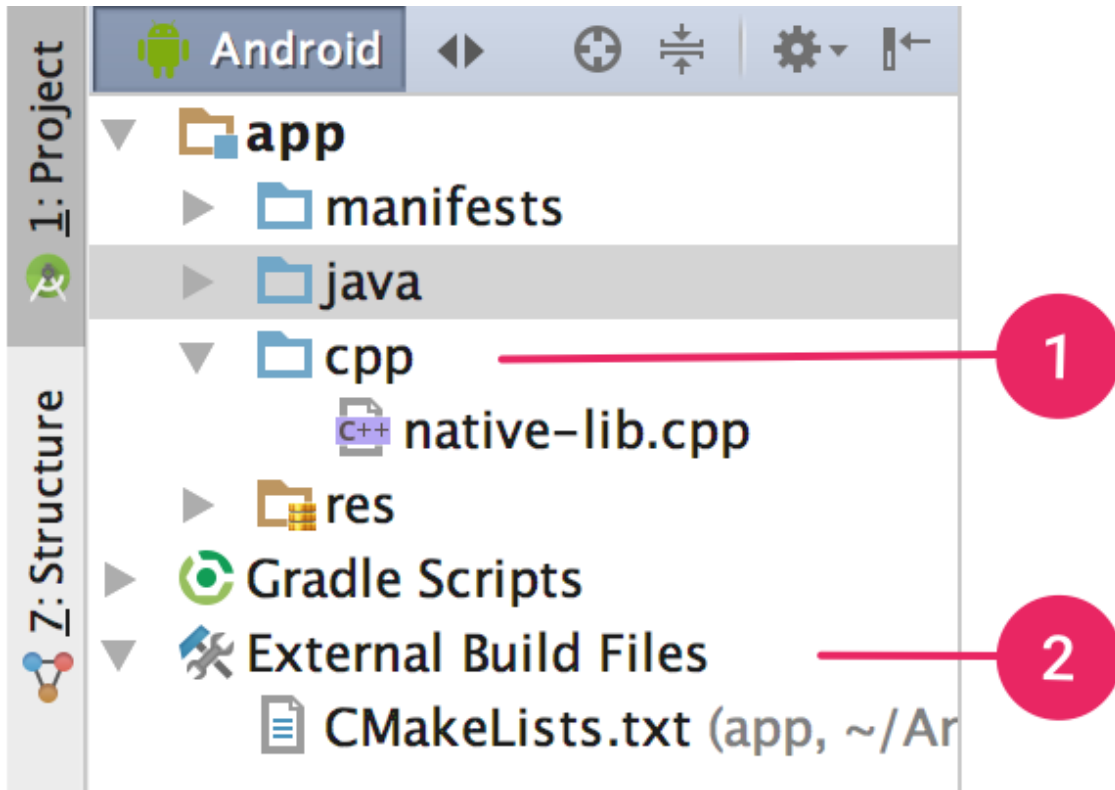
Figure 6

1. The cpp group is where you can find all the native source files, headers, and prebuilt libraries that are a part of your project. For new projects, Android Studio creates a sample C++ source file, native-lib.cpp, and places it in the src/main/cpp/ directory of your app module. This sample code provides a simple C++ function, stringFromJNI(), that returns the string "Hello from C++".

2. The External Build Files group is where you can find build scripts for CMake or ndk-build. Similar to how build.gradle files tell Gradle how to build your app, CMake and ndk-build require a build script to know how to build your native library. For new projects, Android Studio creates a CMake build script, CMakeLists.txt, and places it in your module's root directory.

3. MULTIPLE MICROPHONE ACCESS

Accessing two microphones for Android devices is done by rooting smartphone in previous version. This problem is solved for with this version using proposed framework for audio signal processing. One of these libraries and classes is AudioRecord class which administer the audio sources for recording audio from built-in microphones of the platform [1]. We use Android libraries for accessing two mics at the same. A few parameters have critical importance for signal capture. These are explained as follows.

- *Audio Source* = AudioRecord class allows us to select an audio source of recording. Although there are many options to choose a source, it is recommended that MIC and CAMCORDER. MIC denotes for the bottom microphone of the smartphone and CAMCORDER is used for microphone audio source which is near to the camera.

- *Sampling Frequency* = It is used for determining the sampling rate of recording. According to the developer of Android operating system, 44.1 kHz or 48 kHz is suggested as a sampling frequency for modern Android smartphones. These sampling rates provide high quality and minimum audio latency [2].

- *Channel Configuration* = Number of audio channel is defined via channel configuration parameter. Stereo channel can be selected at max. However, Android doesn't guarantee two-channel recording for all Android phones. Google Pixel 2 XL, Google Pixel, and Samsung Galaxy S7 are tested, and it has seen that two-channel recording is possible for these phones.

- *Audio Format* = The audio data is represented by ENCODING_PCM_16BIT. Each sample is denoted by 16 bit, and Pulse Code Modulation is used for the data.

- *Buffer Size* = The size of recorder buffer is crucial. Since we don`t desire to lose any data when capturing signal. Hence, getminbufsize() function is used for determining buffer size. In order to avoid any data losing while signal capturing, function getminbufsize() has been used for determining the minimal buffer size. The return value from the function above is assigned to Buffer Size.

4. THREE MIC DIRECTION OF ARRIVAL (DOA) ESTIMATION APPLICATION

4.1 DOA ESTIMATION APPLICATION

The algorithm is developed for providing the hearing-impaired user with an estimate of the speaker direction on a graphical user interface (GUI). The application is trained to perform in noisy conditions as well. See Figure 8 for a screenshot of the Android application.

Figure 6

Following are the key things to keep in mind while using the application:

- Before starting the application, please make sure the Smartphone is placed on an elevated surface (so that the microphones are not covered). The GUI should still be fully visible to the user.

- The application will receive the microphone signals and indicate the direction of the speaker using numeric values as well as using graphical marker.

- In presence of speech, the red marker will continuously follow the speaker in all directions. In absence of speech, the blue maker will point to the last estimated speaker location, even with a high-energy noise source present in the room (such as Vacuum cleaner or coffee machine).

To learn more about the application, please refer to our video demos on http://www.utdallas.edu/ssprl/hearing-aid-project/.

5. REFERENCES

[1] https://developer.android.com/reference/android/media/AudioRecord.html

[2] https://developer.android.com/ndk/guides/audio/sampling-audio.html