Gyrotify: Enabling Gyro Controls for Spotify Player

Lei Zhang
University of Michigan
raynez@umich.edu

ABSTRACT

Gyrotify is a system that utilizes the built-in Inertial Measurement Unit (IMU) of a cell phone to control the Spotify player. By manipulating with the cell phone, users can play/pause, skip to prev/next track, adjust volume up/down of the Spotify player.

Author Keywords

Inertial Tracking, Musical Interface, Human-Computer Interaction, Interaction Design

1. INTRODUCTION

The smart phone has been one of the most common consumer electronics nowadays. However, the Inertial Measurement Unit (IMU) inside a smart phone is still under poor deployment in our daily use. The IMU is capable of tracking both the movement and the pose of the smart phone, making it possible to provide information about how users are manipulating their phones. We present Gyrotify, a system that employs the built-in IMU of a smart phone to control the Spotify player. This system can connect to the Spotify account that the user provides and take control of the player. Users can shake their smart phones along the Y axis to play/pause the track, rotate their smart phones around the Z axis to adjust the volume of the track, and rotate their smart phones around the Y axis to skip to next/previous track.

2. USER INTERFACE

Since the only sensor we use here is the built-in IMU of a smart phone, the user interacts with the phone directly. The system is written in Python and deployed on a macOS 10.13 machine. The phone needs to install a App named *GyrOSC* in order to send the IMU data to the system.

3. SYSTEM DESIGN

Figure 2 shows the diagram of this system. Since we need to monitor the smart phone's accelerometer and gyroscope data in real-time, we use the technique of multiprocessing to separate the process of data monitoring and the process of data processing. Each question within Process 1 in Figure 2 represents a signal detector, which we will elaborate in 3.1. If a gesture signal is recognized, it will then push a corresponding task to the queue that we hold. In Process 2, the system executes the tasks in the queue based on the First-In-First-Out (FIFO) order. For each task, the system will send a web request using the Web API provided by Spotify, thus controlling the Spotify player. We will elaborate the process of sending web requests in 3.2.

3.1 Gesture Detector

Figure 1 shows the App setup on the smart phone. The target IP address and the port should be the same as the IP address and the port that our system is listening on. This App can send the built-in IMU data of the smart phone to the target IP address and the port. In this system, we will only utilizes the first two lines of data that this App sends, the gyroscope data and the accelerometer data, as shown in Figure 1.

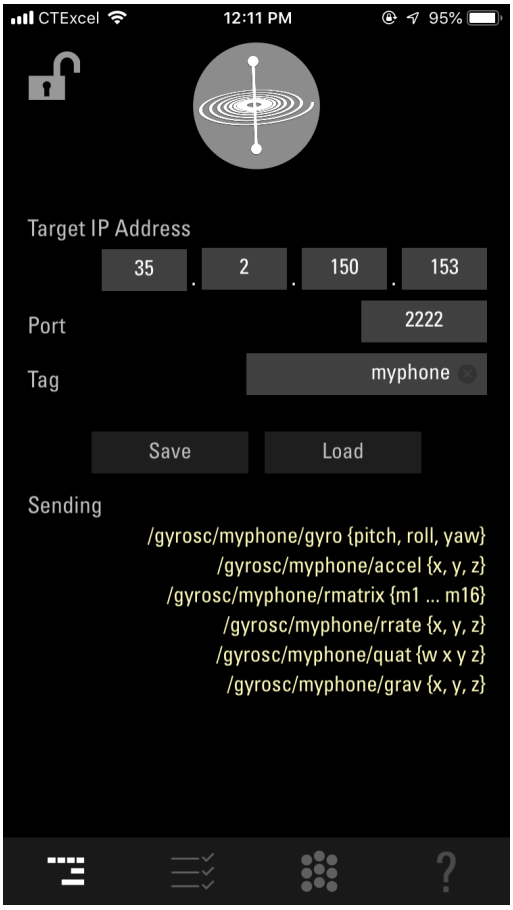

Figure 1: GyrOSC setup on the smart phone side.

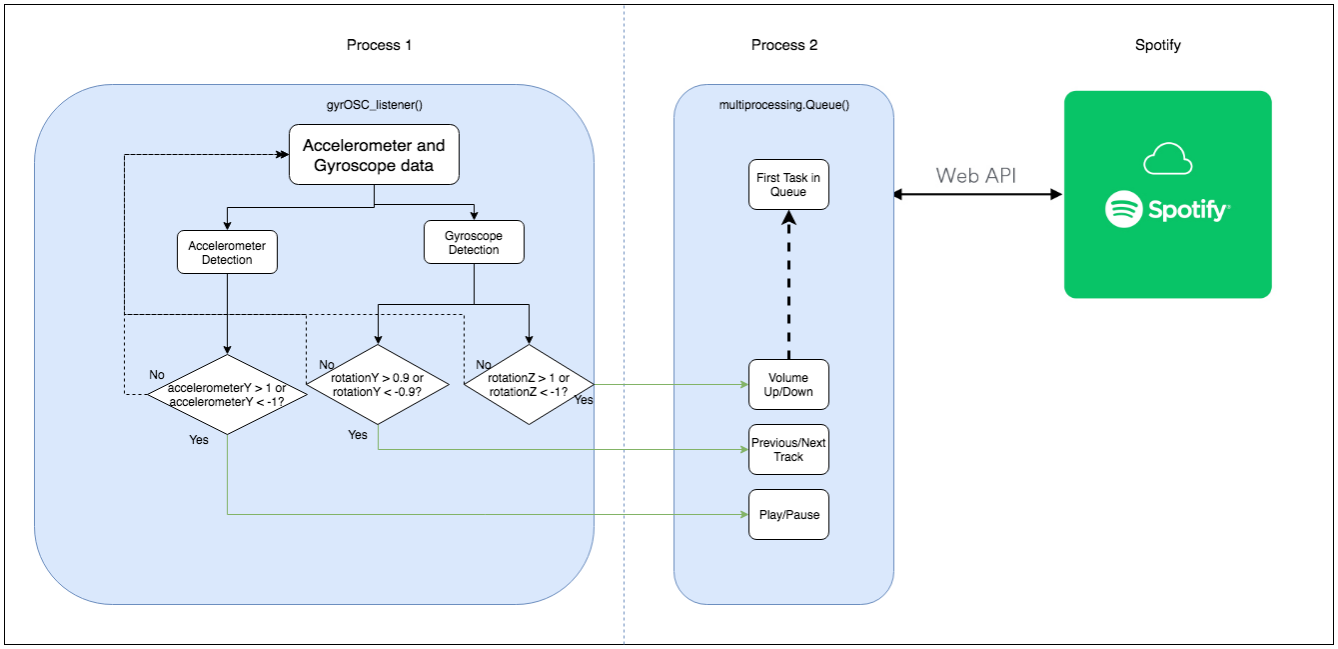

Figure 2: System Diagram.

Since each of the gyroscope data and the accelerometer data can provide data in three degrees-of-freedom (DOF), along/around three axis (X, Y and Z in Figure 3), the whole system can detect a total of six DOF of users' gestures. In this project we only utilize three of them, for controlling play/pause, skip to previous/next track and volume up/down.

For play/pause detection, we employ the accelerometer data from the IMU. When the user shake the smart phone along the Y Axis (in Figure 3) in either direction, the gesture will be recognized as 'shake along Y axis' if the accelerometer data along the Y axis exceeds the threshold we set. After several tryouts, we set the threshold to 1 and -1 in order to recognize the shaking action while filtering out the small shaking of hands.

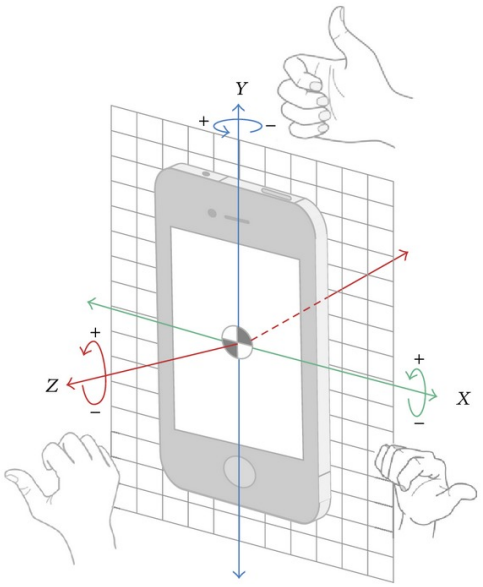

Figure 3: The axis of a smart phone.

For skip to previous/next track detection, we employ the gyroscope data from the IMU. When the user rotates the smart phone around the Y Axis (in Figure 3) in a positive direction, the gyroscope data around the Y Axis will be positive. When the gyroscope value exceeds the threshold (0.9), the gesture will be recognized and the Spotify player will skip to the next track. Similarly, when the user rotates the smart phone around the Y Axis in a negative direction and the value exceeds the minus threshold (-0.9), the gesture will be recognized and the Spotify player will skip to the previous track.

For volume up/down detection, we employ the gyroscope data around the Z Axis (in Figure 3) from the IMU. The system maps the range from 0.5 to 1.0 to five different volume levels. Similarly, the range from -0.5 to -1.0 corresponds to the other five levels of volume. This total of ten levels of volume are mapped to the volume of the Spotify player, which ranges from 0 to 100. Therefore, each level corresponds to volume 10 in Spotify. When the user rotates the smart phone around the Z Axis in a positive direction and the value is between 0.5 and 1.0, it will adjust the volume down according to the rotation level of the smart phone. Similarly, when the user rotates the smart phone around the Z Axis in a negative direction and the value is between -0.5 and -1.0, the volume will be adjusted up according to the rotation level. We set the threshold of 0.5/-0.5 in order to filter out the small shakings of hands.

3.2 Spotify Web API

The Spotify Web API is based on REST principles. Data resources are accessed via standard HTTPS requests in UTF-8 format to an API endpoint. The endpoints return JSON metadata about music artists, albums, and tracks, directly from the Spotify Data Catalogue. Such data access is enabled through selective authorization of the user. The authorization is based on OAuth 2.0 principles. Users can send PUT, POST, GET, DELETE requests to the endpoints in order to control the Spotify.

In this system, we only requires the scope of *user-modify-playback-state* from the Spotify account. Users can get their own token for authoriaztion through the following url:

Table 1: Methods of Different Operations

Method	Endpoint	Usage
PUT	/v1/me/player/play	Start/Resume a User's Playback
PUT	/v1/me/player/pause	Pause a User's Playback
POST	/v1/me/player/previous	Skip User's Playback To Previous Track
POST	/v1/me/player/next	Skip User's Playback To Next Track
PUT	/v1/me/player/volume	Set Volume For User's Playback

`https://developer.spotify.com/console/`. Then the user needs to pass the token to the system so that the system can access the specific Spotify player.

For different tasks of the Spotify player, the system will send different requests to through the Spotify Web API. The base address of Web API is `https://api.spotify.com`. The API provides a set of endpoints, each with its own unique path. The requests and endpoints are shown in Table 1.

4. RESULTS AND DISCUSSION

In this project, we presented a system that utilizes the IMU of a smart phone as user inputs to control the Spotify player. Generally this system works as expected. It is both novel and interesting for a new interaction technique to control a music player. The use of the accelerometer data for playing and pausing the player worked effectively during user tryout. The use of gyroscope data for adjusting volume as well as skipping to the previous/next track also worked well during the user tryout. However, there are also several limitations in the project. First, since the gestures and thresholds are pre-defined, it is not flexible enough for different preferences of users. Second, since the gyroscope data is tied to the rotation of the smart phone, it easily causes false positive results when the user tends to perform other activities of the cell phone. Even though switching App on the cell phone before manipulating it would ease the problem, it is still needed to add a system lock in case of false positive results. Finally, we can consider utilizing more DOFs of the IMU data to perform richer interactions, since we only employs three degrees of freedom of the IMU data. One cool thing about this project is that our system can connect directly to the playlist of the user's Spotify account, making it possible to deploy the system universally, enrich the interactions of Spotify users and enhance the musical experiences of users.