SensorTile as Game Controller in Unity: Motion Detection with Embedded ML

Final Project Report

Introduction

The motivation of this project is to apply to build a hand-free gaming controller. Players just need to wear a wearable device, e.g. smart watch, to play games using hand gestures or body movement. Another application can be human activity tracking and visualization in 3D game engine such as Unity.

In this project, SensorTile is used as the game controller that allow players to perform some motion patterns that are detected by an embedded machine learning (ML) model running on the SensorTile's processor. It allows decent machine learning technique running on a lightweight device and works independently to a target application. SensorTile detects user's gesture and send the result to the integrated game engine via Bluetooth Low Energy and Wifi connection.

Problem Description

In this project, SensorTile is programmed work as an hand-free game controller. The use case scenario is that users hold or wear an SensorTile-equipped device and perform some predefined motions such as tilting their hand to control movement of a game character, and/or trigger some actions. The objective of this project is to simulate a basic gamepad including a D-pad and two action buttons (Figure 1).

Figure 1 - SensorTile simulates a standard gamepad of 6 buttons

Motion patterns are detected by an embedded ML model [1] running on the SensorTile. There are 6 motions needed to be trained and detected to simulate triggers of 6 buttons in a standard gamepad. The motions proposed for this project are based on changing in rotation angle in relative to the starting position of the SensorTile. There are 6 motion gestures for 6 buttons, including (Figure 2):

- Tilt the SensorTile about X or Y axis for D-pad buttons
- Rotate the SensorTile about Z-axis clockwise and counter-clockwise for Button A and B.

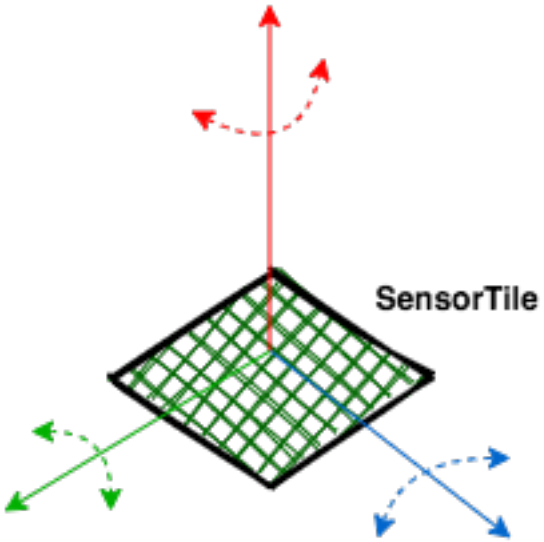

Figure 2 - 6 motions based on rotating SensorTile in 3 axises: X - Y for D-Pad buttons and Z-axis for action buttons A and B

Project Components

Motion detections are run as SensorTile's firmware where trading request or detected motions are sent via a Bluetooth Low Energy (BLE) connection. The chosen game engine is Unity. Instead of having Unity communicating with SensorTile directly via BLE connection, BeagleBone is used as an intermediate device to to receive SensorTile's data via bluetooth connection, process, and forward to Unity via WebSocket protocol. There are 3 main components in this project and how they are communicating are shown in Figure 3.

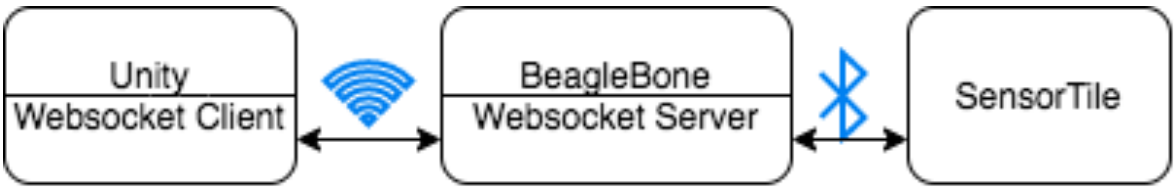

Figure 3 - Three main components of this project

In trade of requiring an additional single board computer like BeagleBone, the main advantage of this connection scheme is that it allows SensorTile to integrate with many types of game engines and end-user computer which may not support Bluetooth 4.0 which is required for BLE connection.

Data Acquisition and Feature Extraction

To detect 6 different rotation motions, rotation rate from SensorTile's microgyroscope sensor are collected to measure rotation angle (for X, Y, and Z axis) of SensorTile to its starting position. Rotation angle is computed by integrating the angular rate R using the following formula:

$$\theta(t) = \theta(t = 0) + \int_{\tau=0}^{\tau=t} R(\tau) d\tau$$

In case of SensorTile with discrete time intervals, the angle is computed by a sum using rotation rate collected by the sampling period, T_{sample}, applied to each axis. This is the same method described in Tutorial 13 [2].

$$\theta_X(t) = \theta_X(n = 0) + \sum_{i=0}^{i=n} \frac{R_X(i-1) + R_X(i)}{2} T_{sample}$$

$$\theta_Y(t) = \theta_Y(n = 0) + \sum_{i=0}^{i=n} \frac{R_Y(i-1) + R_Y(i)}{2} T_{sample}$$

$$\theta_Z(t) = \theta_Z(n = 0) + \sum_{i=0}^{i=n} \frac{R_Z(i-1) + R_Z(i)}{2} T_{sample}$$

The sample rate T_{sample} is 10ms which is the maximum rate of SensorTile. The time t is variant depending on how long the user performs a rotation motion. It is decided by two conditions:

1. Rotation angle magnitude of minimum 30 degrees; or
2. Up to 1 second (equivalent to 100 sampling cycles)

The threshold 30 degrees chosen in (1) is to strongly indicating that user is intended to perform a motion instead of some unintentional movement. The time limit of 1 second chosen in (2) allows to capture a normal human gesture taken within 1 second while not

blocking the execution of the main loop in the SensorTile. For example, if the user performs a rotation of 30 degrees in over one second, this motion is ignored and SensorTile continues to capture next motions.

Model Training and Testing

Rotation angle in X, Y and Z axis are used as features for training an artificial neural network (ANN). Features are normalized to have their values in range [-1, 1]. The ANN has the same topology as in Tutorial 13. 3-neural input layer accepts rotation angle in all 3 axises. 9 neural in the hidden layer and 6 neural in output layer corresponding to 6 detecting motions.

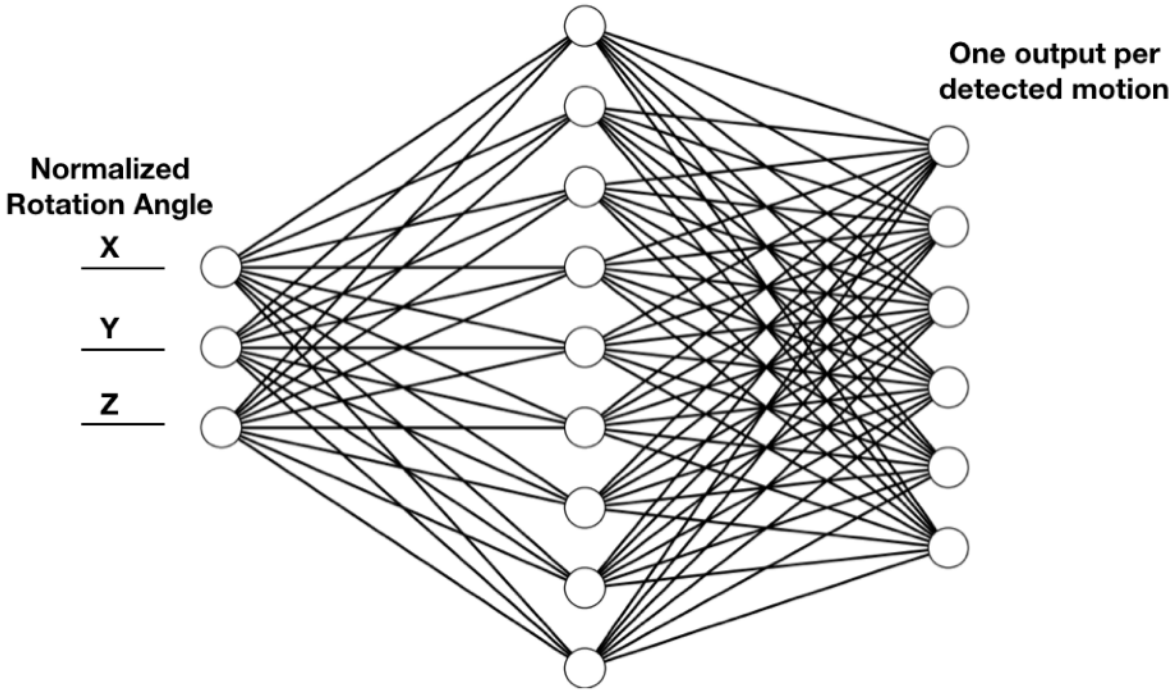

Figure 4 - Neural Network Topology for detecting 6 motions based on 3 extracted features from rotation angle about X, Y, and Z axis.

In my experiment, 6 training data instances are collected to train the ANN. Table 1 and 2 shows a training session and its test result. The training is completed in 60-80 epochs with z-score threshold of 1 as shown the screenshot in Figure 5.

Rotation Angle			Labeled Motion
X	Y	Z	
30	1	0	0
-30	3	0	1

Rotation Angle			Labeled Motion
X	Y	Z	
0	30	1	2
0	-30	0	3
0	0	30	4
2	5	-29	5

Table 1 - 6 data instances for a training session.

Rotation Angle			Detected Motion
X	Y	Z	
5	29	0	2
0	-31	0	3
1	-30	-1	3
29	-3	-2	0
-30	5	-1	1
-30	0	6	1
4	1	29	4
-9	-3	28	4
18	7	-22	5
2	-3	-33	5

Table 2 - Testing result that correctly detected 6 motions

```
Training Epochs: 60

State 0 Max 57  Mean 16         Z-score 128     Outputs 57      -40     18      22      18      23
State 1 Max 55  Mean 12         Z-score 170     Outputs -19     55      21      -1      9       9
State 2 Max 41  Mean 17         Z-score 134     Outputs 16      20      41      -13     19      20
State 3 Max 36  Mean 10         Z-score 154     Outputs 21      -8      -4      36      9       10
State 4 Max 53  Mean 15         Z-score 141     Outputs 25      8       23      7       53      -27
State 5 Max 61  Mean 13         Z-score 171     Outputs 14      3       16      12      -23     61
Index 0 Error State: 0
```

Figure 5 - Training completed within 60-80 epochs with Z-score threshold of 1.00

Project Code

The Github repository of this project contains 3 sub-projects including:

- `STile_Rotation`: Implementation of SensorTile firmware. This project extends on the project provided in Tutorial 13 to update the program flow and add BLE communication with a connected device, e.i. BeagleBone:
  - Receive command to for training step.
  - Send detected motion in running step
- `beaglebone/sensortile`: a NodeJS project that implements a server program running on a BeagleBone. This program provides the connectivity between a client program and SensorTile by:
  - Send and receive data to SensorTile via BLE connection. It is done by invoking gatttool installed on the operating system of BeagleBone.
  - Host a Websocket server to allow connection from a Websocket client, e.g. the demo Unity program, forward request and data received from SensorTile.
- `SensorTileController`: an Unity project that implements of the demo game scene in a Unity project

Conclusions and Future Works

In this project, I have successfully implemented a solution for the described problem that is to demonstrate the use of SensorTile as a simple game controller. My main contributions of this project is threefold: a SensorTile firmware, a server program running on BeagleBone and a demo game scene implemented in Unity. The SensorTile is able to run a ML model to detect human gestures, and utilizes wireless communication (Wifi and BLE) to provide a hand-free game controlling experience.

However, the designed motion pattern depends on the initial position of the SensorTile that require user to return to the starting position to perform next motion. This can be improved in future work of this project to acquire more features from accelerometer and magnetometer sensors.

References

[1] Charles Zaloom - EmbeddedML - Available at https://github.com/merrick7/EmbeddedML

[2] STMicroelectronics SensorTile Tutorial: Motion Pattern Recognition by Rotation Angle Classification with Machine Learning. Retrieved from https://sites.google.com/view/ucla-stmicroelectronics-iot/home