BeatSaver: Conducting Gestures to Metronome Beats

Ryuhaerang Choi, Yewon Kim
Korea Advanced Institute of Science and Technology (KAIST)
{ryuhaerang.choi, yewon.e.kim}@kaist.ac.kr

Abstract

A conductor is a messenger for the composer, and musicians' duty is to fully understand the conductor's messages through gestures and produce unified music in a musical ensemble. However, in many cases where the conductor is absent, musicians inevitably resort to their faint memories or abstract note-taking to infer the conductor's instructions. To revamp this limitation, we propose BeatSaver that converts conducting gestures to metronome beats. Musical details that a conductor instructed are recorded in beats and dynamics so that musicians can refer to even when they are practicing on their own. BeatSaver consists of Time Signature Classifier, Beat Detector, and Musical Dynamics Classifier, which we prove their effectiveness through real-world experiments. BeatSaver is lightweight and practical as it only utilizes Arduino Nano 33 BLE Sense attached to the conducting baton and an external server to extract the details after the practice session, making it easily applicable in the wild.

1 Trajectory Tracking

Tracking the trajectory of a user using inertial measurement unit (IMU) sensors have been studied over the last two decades [10, 13, 18, 21]. In this paper, we tried to build an accurate trajectory tracking method using Arduino Nano 33 BLE Sense [1]. We focused on minimizing sensor errors to improve the performance of trajectory tracking.

1.1 Manual Sensor Calibration

When sensors are not calibrated, the outputs of them may vary per instance on chip [25]. Therefore, we manually calibrate the sensors in the following steps:

1. Place Arduino Nano 33 BLE Sense on a flat table
2. Install and run LSM9DS1 Library to Arduino Nano 33 BLE Sense
3. Start accelerometer calibration
   (a) Measure X-, Y-, and Z-axis in both directions
   (b) Calculate offset of accelerometer
4. Start gyroscope calibration
   (a) Measure offset of gyroscope's each axis while rotating Arduino Nano 33 BLE Sense over rotation axis
   (b) Calculate offset of gyroscope

Sensor	X-axis offset	Y-axis offset	Z-axis offset
Accelerometer	-0.021173	0.001834	0.012719
Gyroscope	0.031862	-0.421852	-0.025839

Table 1: The accelerometer' and gyroscope' offset values of the sensor. The offset values are not the same for all modules.

The results of offsets are in Table 1. We compensate the offsets by modifying the Arduino LSM9DS1 library before starting trajectory tracking (See Fig. 1).

1.2 Position Estimation

After sensor calibration, we explored sensor readings while moving the sensor. We observed that several unexpected peaks might make huge errors when we estimate the position of the sensor using the values. To reduce the impact of such peaks in sensor readings, we estimated the position based on the mean value of three consecutive sensor readings. Figure 2 describes how we estimate position in x, y, and z axis. The output, D, of function *avg* is the set of average values of the variables, the group of raw sensor readings. The function *PE* takes sensor readings from the accelerometer and gyroscope as input and estimates the position, $P = \{x, y, z\}$ of the sensor. The sequence of the estimated position (i.e., $\{P_1, P_2, P_3, ...\}$) is the trajectory of the sensor.

2 Trajectory Tracking: Evaluation

We followed the evaluation methods, test case #1 and #2, in **Project Overview** document. The only difference is that the ground truth is set to 20 centimeters. For each evaluation method, we conducted the experiment twenty times.

The error is obtained by calculating the absolute difference between the ground truth and the measurements in centime-

LSM9DS1.h

```
// Accelerometer Calibration
float accelOffset[3] = {acc_x_offset, acc_y_offset, acc_z_offset};
float accelSlope[3] = {acc_x_slope, acc_y_slope, acc_z_slope};
// Gyroscope Calibration
float gyroOffset[3] = {gyro_x_offset, gyro_y_offset, gyro_z_offset};
float gyroSlope[3] = {gyro_x_slope, gyro_y_slope, gyro_z_slope};
```

(a) Add new variables representing the offsets to be calibrated in LSM9DS1.h in Arduino LSM9DS1 library.

LSM9DS1.cpp

```
int LSM9DS1Class::readAcceleration(float& x, float& y, float& z)
{
  int16_t data[3];
  if (!readRegisters(LSM9DS1_ADDRESS, LSM9DS1_OUT_X_XL, (uint8_t*)data,
      sizeof(data))) {
    x = NAN;    y = NAN;    z = NAN;
    return 0;
  }
  //x = data[0] * 4.0 / 32768.0;
  //y = data[1] * 4.0 / 32768.0;
  //z = data[2] * 4.0 / 32768.0;
  x = accelSlope[0] * ((4.0 * data[0] / 32768.0) - accelOffset[0]);
  y = accelSlope[1] * ((4.0 * data[1] / 32768.0) - accelOffset[1]);
  z = accelSlope[2] * ((4.0 * data[2] / 32768.0) - accelOffset[2]);
  return 1;
}
int LSM9DS1Class::readGyroscope(float& x, float& y, float& z)
{
  int16_t data[3];
  if (!readRegisters(LSM9DS1_ADDRESS, LSM9DS1_OUT_X_G, (uint8_t*)data,
      sizeof(data))) {
    x = NAN;    y = NAN;    z = NAN;
    return 0;
  }
  //x = data[0] * 2000.0 / 32768.0;
  //y = data[1] * 2000.0 / 32768.0;
  //z = data[2] * 2000.0 / 32768.0;
    x = gyroSlope[0] * ((data[0] * 2000.0 / 32768.0) - gyroOffset[0]);
    y = gyroSlope[1] * ((data[1] * 2000.0 / 32768.0) - gyroOffset[1]);
    z = gyroSlope[2] * ((data[2] * 2000.0 / 32768.0) - gyroOffset[2]);
  return 1;
}
```

(b) Compensate offsets by modifying accelerometer and gyroscope sensor readings in LSM9DS1.cpp in Arduino LSM9DS1 library. The code lines commented out are the original codes.

Figure 1: Accelerometer and gyroscope calibration codes.

ters. For the first test case, the average and standard deviation of errors were 2.75 and 1.39, respectively, in a centimeter scale. For the second test case, the average and standard deviation of errors were 4.75 and 3.50, respectively in a centimeter scale.

3 BeatSaver: Motivation and Background

3.1 Introduction

Conducting is crucial in musical ensembles so that musicians can play unified music that the conductor intends to convey to the audience. However, the conductor is not always present in the practice room; it is common for musicians to practice independently and meet their conductor every fixed interval of time, e.g., one week. Therefore, when practicing without the conductor, musicians often heavily depend on their faint memory or abstract note-taking to infer the details that the conductor instructed. In other words, although it is important

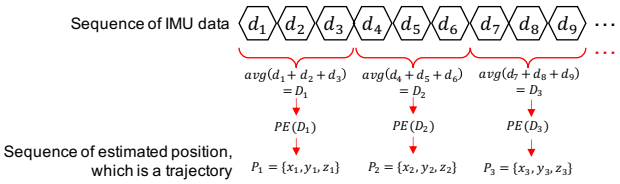


Figure 2: Diagram of position estimation process.

for the performer to remember the conducting details, there is no tool that helps musicians do so.

There have been several attempts to track conducting gestures using video recordings [2, 3, 6, 14]. However, it is often hard to place a camera and record the gesture correctly in a small and crowded practice room. Moreover, vision approaches often involve privacy-related issues, which makes the system not widely applicable [20].

To provide a better practicing environment for performers, we present BeatSaver, an automatic recording system that converts conducting gestures into metronome beats. BeatSaver only utilizes Arduino Nano 33 BLE Sense [1] that is small enough to attach to a tip of a conducting baton. The recorded data are passed to an external server where we use *Time Signature Classifier*, *Beat Detector*, and *Musical Dynamics Classifier* to generate beats and dynamics from the conducting gestures.

3.2 Related Work

Mobile sensing technologies that utilize sensors from mobile devices provide diverse services such as human activity recognition [11, 24], gesture recognition [22, 26], healthcare [16], and authentication [5, 15, 17]. Few works have tackled the problem of recording conducting details by using depth camera or leap motion sensors that utilize vision information [2, 3, 6, 14]. However, placing such sensors in front of a conductor while practicing is difficult because the device can block the musician's view, and the practice room is often small and crowded. Moreover, vision approaches often raise privacy-related issues and require high computational cost, which keeps the system from being widely applicable [20]. One study proposes a similar idea to utilize sensors, yet the authors do not propose detailed methods to realize the system [14].

4 BeatSaver: System Design

4.1 Overview

We consider a scenario where a conductor conducts a musical ensemble, e.g., orchestra or chorus, with a conducting baton where an IMU sensor is attached to its tip. The goal of BeatSaver under the scenario is to record the beats and dynamics of conducting gestures so that the musical ensemble can refer to even when the conductor is not present, e.g., when practicing on their own. Figure 3 illustrates the workflow of our

framework. BeatSaver trains two deep learning models: *Time Signature Classifier* that classifies the time signature, and *Beat Detector* that classifies whether the beat has changed, given accelerometer and gyroscope readings. We provide further details in section 4.2 and section 4.3. Given the detected beat sequences, we use *Trajectory Tracker* defined in section 1 to extract the length of the trajectories between detected beats. Using the extracted lengths, we statistically induce the threshold length between two representative dynamics: piano and forte, which we simply call *Dynamics Classifier*. We detail this process in section 4.4.

4.2 Time Signature Classifier

Time signature is an abbreviation for the music's rhythms; for example, the time signature of 4/4 means that four beats are equivalent to one measure. This project considers three kinds of time signatures for simplicity: 2/4, 3/4, and 4/4. We utilize the fact that conducting gestures for each time signature differs (see Figure 4), which makes it possible to use only accelerometer and gyroscope data to detect the time signature. We apply traditional deep learning approaches to train *Time Signature Classifier*. Model hyperparameters are detailed in section 5.1.2.

4.3 Beat Detector

4.3.1 Detecting Moment that Beat Changes

After the time signature is detected, we use corresponding *Beat Detector* to determine whether the beat was changed within the input data chunk. We train three types of *Beat Detector* for each time signature: 2/4, 3/4, and 4/4. The gestures that indicate beat change are illustrated as white circles in Figure 4. We train a deep-learning-based classifier that receives accelerometer and gyroscope data recorded for a fixed duration and outputs True if the input sensor data includes a beat change. We train two kinds of beat detectors: binary classifier and multi-class classifier. A binary classifier returns true if any beat change is detected, while a multi-class classifier differentiates each gesture indicating beat changes. As a result, a multi-class classifier of time signature N/4 has $N+1$ classes, e.g., {1st beat change, 2nd beat change, not changed} for 2/4 time signature.

Rather than training one unified beat detector, we train three separate detectors for each time signature since some gestures of beat change are hard to distinguish from the gestures that do not indicate beat changes. Specifically, in Figure 4, we can observe that the drastic angular changes indicate the beat change in most cases, as illustrated in the gestures of time signatures 2/4 and 3/4. However, in the gesture of time signature 4/4, two of the beats change with relatively fewer angular changes, which makes training a unified model difficult. We conducted an exploratory experiment by integrating all data and training the unified model and found out that the performance was deficient (34.7% in accuracy). We reckon that a larger amount of data and a more complex deep learning model structure will solve the issue, yet we leave this as future work.

4.3.2 Ensemble Learning Approaches

From the experiments, we observed that the outputs of the single trained model result in relatively low accuracy (73.3% on average), possibly due to a lack of data. To revamp the issue, we take ensemble learning approaches to obtain more reliable results. We combine the predictions of several estimators trained with diverse structures and configurations. Specifically, we train various classical convolutional neural networks and long short-term memory (LSTM) networks, with configurations including varying batch sizes, random seeds, optimizers, data preprocessing protocols, and the number of classes. Here, the data preprocessing protocols mean whether to down-sample the data for even distributions or not, which is detailed in section 5.1.1. The number of classes are explained in section 4.3.1. From the trained models, we select the ones whose best model's validation accuracy is higher than 80% and determine the prediction with a majority vote.

4.4 Musical Dynamics Classifier

The musical dynamics is the variation in loudness between notes and phrases in music. In score, it is represented by specific musical notations (e.g., *pp*, *mp*, *mf*, and *ff*). When conducting, the conductor expressed musical dynamics through the relative volumes of the conducting gestures. In addition to recording each beat and type of beats, BeatSaver identifies the musical dynamics from conducting gestures.

4.4.1 Tracking baton-tip trajectory

The volume of conducting gesture, which implies the musical dynamics, directly matched with the length of the baton-tip trajectory. Therefore, BeatSaver determines musical dynamics the conductor represented by measuring the length of baton-tip trajectory. The methods for trajectory tracking are described in section 1. On top of the trajectory tracking methods, we segment the baton-tip trajectories by each beat which can be obtained from *Beat Detector*.

4.4.2 Statistical threshold

We estimated every trajectory of the conducting in the dataset used in this paper and statistically analyzed their lengths. When the musical dynamics is *piano* (*p*), the mean and standard deviation of trajectory length is 26.59 (cm) and 7.76 (cm) respectively. When it comes to *forte* (*f*), the mean and standard deviation of trajectory length is 48.35 (cm) and 1.53 (cm) respectively. Because there is a gap between the length of trajectories when musical dynamics is *p* and *f*, we decided to set a threshold to determine the musical dynamics between different dynamics. For BeatSaver we implemented, threshold for section 4.4 set to **40.00** (cm).

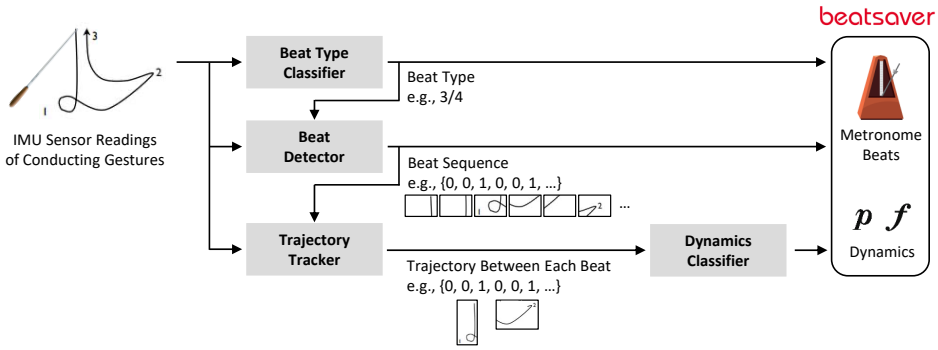


Figure 3: An overview of BeatSaver that converts IMU sensor data of conducting gestures to metronome beats and musical dynamics.

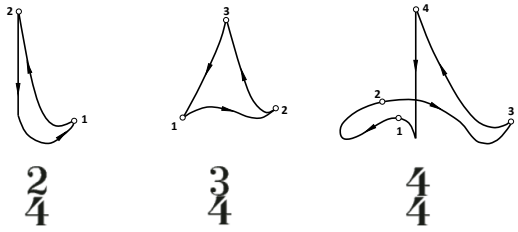


Figure 4: Conducting gestures of three kinds of time signatures used in the project.

Figure 5: Arduino Nano 33 BLE Sense attached to the conducting baton.

5 BeatSaver: Evaluation

5.1 Settings

5.1.1 Data Collection

We detail the data collection protocol and preprocessing methods of our dataset. We used Arduino Nano 33 BLE Sense for collecting data. We increased the sampling rate of the sensor from the default value of 119Hz to 476Hz. To simulate a scenario where the sensor records conducting gestures, we attached the Arduino Nano 33 BLE Sense [1] to the tip of the conducting baton [19]. (see Figure 5)

Each data collection session started by generating five synchronization pulses by knocking the sensor with a hard object. We video-recorded the gestures for a labeling purpose. Each session lasted 5 minutes, and total of 24 sessions were held to collect 120 minutes of raw data (40 minutes per each time signature.) We differentiate the dynamics of the conducting gestures: *piano* and *forte*. Each dynamics takes 20 minutes per each time signature. The collected data was hand-annotated against video recordings. We recorded timestamps of every beat change using BORIS software [4].

We recorded each x, y, and z-axis values of accelerometer and gyroscope at the maximum sampling rate and downsampled data to 399Hz. We divide the data with the window length of 133 or 266 with an overlapping ratio of 0.5. Window length of 133 is used for *Beat Detector*, and the length 266 is used for *Time Signature Classifier* to capture long-term data information. Data processed by the window length was labeled to include "beat change" if the data instance with the timestamp that beat change was detected was included in the data chunk.

5.1.2 Implementation

We design *Time Signature Classifier* with 1D-convolutional neural networks (CNN) followed by fully-connected (FC) layers, since CNN is commonly used in mobile sensing fields [7, 8]. Specifically, the model consists of five CNN layers and three FC layers. Rectified Linear Unit (ReLU) is used for activation function. We train the model with Adam optimizer and learning rate of 0.001, which is scheduled to reduce with a ratio of 0.1 every 50 epochs. We train the model for 200 epochs and select the best model with the highest evaluation accuracy.

Three model structures are designed for *Beat Detector* for ensemble learning. First is the same as *Time Signature Classifier*. The second design is the lighter version of CNN, where it has five simpler CNN layers and one FC layer. The learning rate is set to 0.001, with the same learning rate scheduler as *Time Signature Classifier*. For the last structure, we use unidirectional long short-term memory (LSTM) network, which consists of 50 hidden dimensions and two layers. The LSTM network is connected with two fully-connected layers. We set the learning rate as 0.01 with the same learning rate scheduler as above.

We trained the models with various model hyperparameters for ensemble learning (details in section 4.3.2.) Specifically, we vary batch sizes (32 or 64), random seeds (0 or 1), optimizers (Adam or SGD), data preprocessing protocols, and the number of classes. For the data preprocessing protocol,

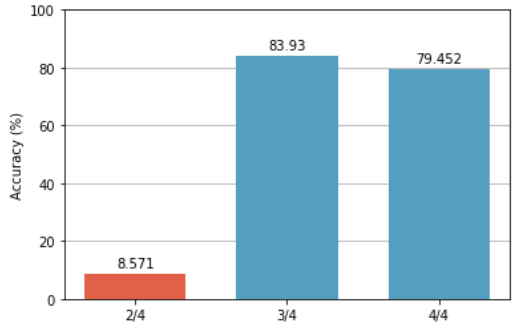


Figure 6: Best accuracy of Time Signature Classifier.

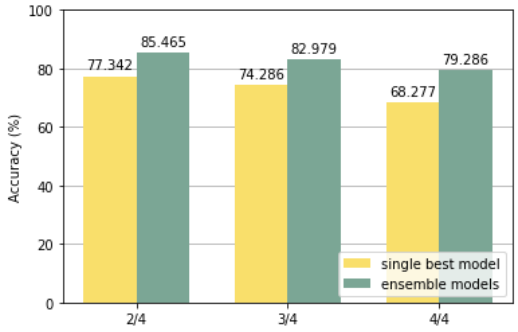


Figure 7: Comparison of the best accuracy achieved by single model and ensemble models (Beat Detector.)

we decide whether to down-sample the data or not to resolve the imbalance between different classes. If down-sampled, we match the number of data for each class to that of the class with the least number of data. In terms of the number of classes, we trained both multi-class and binary classifiers. A multi-class classifier differentiates the order of beat-changing gestures, while a binary classifier treats all beat-changing gestures as one. Thus, a multi-class classifier of time signature $N/4$ has $N+1$ classes, e.g., 1st beat change, 2nd beat change, not changed for 2/4 times signature. We also implement feature engineering of IMU data by calculating mean, standard deviation, and variance of accelerometer and gyroscope values.

For the evaluation of *Time Signature Classifier* and *Beat Detector*, we used three different test data: conducting gestures of each time signature, which lasts one minute each. We used the same window size (266 for *Time Signature Classifier* and 133 for *Beat Detector*), but in the test phase, non-overlapping data was fed to the classifier. We divided train, validation, and test data so that no instance is overlapped.

5.2 Performance of Time Signature Classifier

Figure 6 reports the best accuracy achieved by *Time Signature Classifier* on test dataset. In the latter two cases, the accuracy is fairly high. Time signature 2/4, however, has an extremely low accuracy of 8.571%. From the analysis, we recognized that the model was highly overfitted to 4-beats data. It is a reasonable result because conducting gesture of 2/4 is similar to gesture sequence of 2-3-4 in time signature 4/4 (Figure 4). Possible solutions would be increasing the window size or collecting more data to prevent overfitting. We leave this as future work.

5.3 Performance of Beat Detector

Figure 7 shows the accuracy of *Beat Detector*, accompanied by the comparison with the single best model. Overall, we could observe that ensemble learning approaches were effective in terms of accuracy, achieving an average accuracy of 82.577%.

5.4 Performance of Musical Dynamics Classifier

In BeatSaver, the relative trajectories are matter in order to estimate the dynamics. Therefore, we visualize the estimated trajectories in three-dimension space to investigate relativity. Before visualization, we merged trajectories referring to the beat type from *Time Signature Classifier*. The number of trajectories which are merged is two for 2/4 beat type, three for 3/4 beat type, and four for 4/4 beat type, respectively. Figure 8 illustrates the visualized trajectories. From the visualized trajectories, we observed that (1) the trajectories are similar to each other, and (2) there is a difference in the length of trajectories between different dynamics.

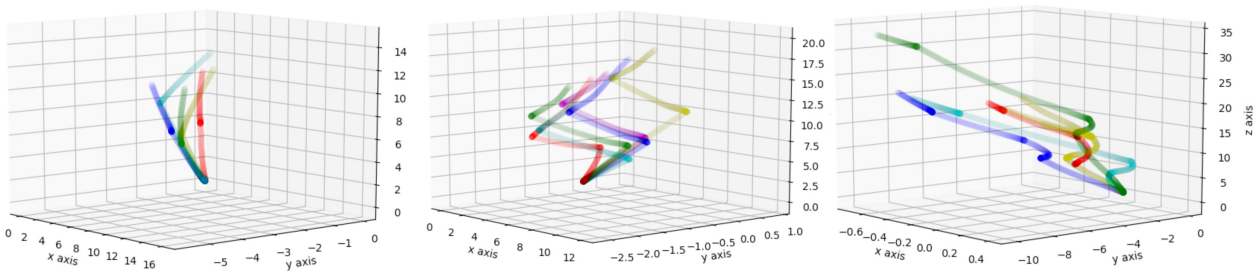
6 BeatSaver: Limitation and Discussion

6.1 Diversity of Time Signatures

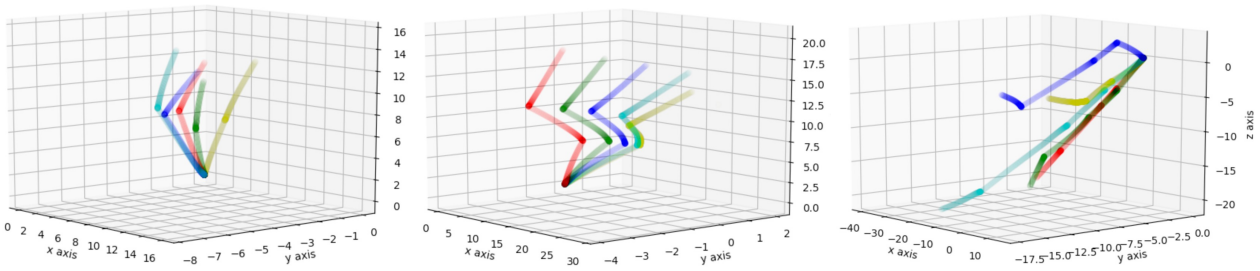
In the project, we only considered three time signatures: 2/4, 3/4, and 4/4. The limitation of the BeatSaver is that the number of required *Beat Detector* increases as the diversity of time signatures increases. We believe that with an increased amount of data, we will be able to combine *Time Signature Classifier* and *Beat Detector* into one classification model that can extract beats from any conducting gestures.

6.2 Diversity of Musical Dynamics Expressions

Each conductor has a different conducting style, and each musical dynamics is expressed in a different volume of conducting gesture [9]. The number of conductors in the dataset is limited to two, and their conducting styles were similar to each other. Therefore, the threshold-based musical dynamics classifier showed quite a reliable performance. However, as the diversity of conductors increases in the real-world, the performance of threshold-based will be significantly lower. Therefore, it will be an alternative to display relative dynamics (e.g., the same and the louder) using sequential models such as LSTM [12] and Bayesian model [23]. Showing the relative dynamics to one of the previous phases will be a breakthrough to mitigate the diversity of conductors in the real-world.



(a) 2/4 beat in *piano* dynamics (b) 3/4 beat in *piano* dynamics (c) 4/4 beat in *piano* dynamics



(d) 2/4 beat in *forte* dynamics (e) 3/4 beat in *forte* dynamics (f) 4/4 beat in *forte* dynamics

Figure 8: The results of visualizing five trajectories with each musical dynamics in a three dimensional space. Each trajectory is randomly selected from the dataset and is identified in a different color.

6.3 Different Conducting Styles

We did not consider personalization in this project, yet the conducting gestures vary depending on different conductors [9]. Applying fine-tuning technologies on the base model would be required for the BeatSaver to be deployed in the wild.

6.4 Conclusion

In this project, we built a lightweight and practical conducting recording system, BeatSaver, that is attachable to a conducting baton. BeatSaver infers (1) time signature, (2) beat, and (3) musical dynamics achieves by collecting and analyzing the IMU sensor data. BeatSaver achieves To our best knowledge, BeatSaver is the first conducting recording system based on only IMU sensor data. We believe BeatSaver can provide a better practicing experience for musicians by recording conducting details while not hindering conductors' conducting experience. We hope further works will deal with the technical challenges we present in this paper.

Acknowledgments

This work was supervised by our awesome professor Song Min Kim and his wonderful teaching assistants. Thank you for the great course :) Also, special thanks to Yewon's younger sister, Yejin, who generously let us attach Arduino sensor to her precious magic wand!

Availability

The code of BeatSaver is available at GitHub. The demo video is also available at YouTube.

References

[1] Arduino. Arduino arduino nano 33 ble sense. https://store-usa.arduino.cc/products/arduino-nano-33-ble-sense, 2021. Accessed: 2021-12-19.

[2] Chase D Carthen, Richard Kelley, Cris Ruggieri, Sergiu M Dascalu, Justice Colby, and Frederick C Harris. Muse: A music conducting recognition system. In *Information Technology-New Generations*, pages 363–369. Springer, 2018.

[3] Fahn Chin-Shyurng, Shih-En Lee, and Meng-Luen Wu. Real-time musical conducting gesture recognition based on a dynamic time warping classifier using a single-depth camera. *Applied Sciences*, 9(3):528, 2019.

[4] Universita Degli Studi Di Torino DBios. BORIS behavioral observation research interactive software. http://www.boris.unito.it, 2021. Accessed: 2021-12-18.

[5] Muhammad Ehatisham-ul Haq, Muhammad Awais Azam, Usman Naeem, Yasar Amin, and Jonathan Loo.

Continuous authentication of smartphone users based on activity pattern recognition using passive mobile sensing. *Journal of Network and Computer Applications*, 109:24–35, 2018.

[6] Joyce Horn Fonteles and Maria Andréia Formico Rodrigues. User experience in a kinect-based conducting system for visualization of musical structure. *Entertainment Computing*, 37:100399, 2021.

[7] Taesik Gong, Yeonsu Kim, Ryuhaerang Choi, Jinwoo Shin, and Sung-Ju Lee. Adapting to unknown conditions in learning-based mobile sensing. *IEEE Transactions on Mobile Computing*, 2021.

[8] Taesik Gong, Yeonsu Kim, Jinwoo Shin, and Sung-Ju Lee. Metasense: few-shot adaptation to untrained conditions in deep mobile sensing. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 110–123, 2019.

[9] Robert Nathan Grechesky. *An analysis of nonverbal and verbal conducting behaviors and their relationship to expressive musical performance (aesthetic, gestural)*. The University of Wisconsin-Madison, 1985.

[10] Yunsu Ha and Shinichi Yuta. Trajectory tracking control for navigation of self-contained mobile inverse pendulum. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, volume 3, pages 1875–1882. IEEE, 1994.

[11] Mohammed Mehedi Hassan, Md Zia Uddin, Amr Mohamed, and Ahmad Almogren. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, 81:307–313, 2018.

[12] Dilip Chakravarthy Kavarthapu and Kaushik Mitra. Hand gesture sequence recognition using inertial motion units (imus). In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 953–957. IEEE, 2017.

[13] Mikkel Baun Kjærgaard, Sourav Bhattacharya, Henrik Blunck, and Petteri Nurmi. Energy-efficient trajectory tracking for mobile devices. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 307–320, 2011.

[14] Eric Lee, Henning Kiel, Saskia Dedenbach, Ingo Grüll, Thorsten Karrer, Marius Wolf, and Jan Borchers. isymphony: an adaptive interactive orchestral conducting system for digital audio and video streams. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 259–262, 2006.

[15] Jingwei Liu, Fanghui Cai, Longfei Wu, Rong Sun, Liehuang Zhu, and Xiaojiang Du. Epda: Enhancing privacy-preserving data authentication for mobile crowd sensing. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[16] Sumit Majumder and M Jamal Deen. Smartphone sensors for health monitoring and diagnosis. *Sensors*, 19(9):2164, 2019.

[17] Maryam Naseer Malik, Muhammad Awais Azam, Muhammad Ehatisham-Ul-Haq, Waleed Ejaz, and Asra Khalid. Adlauth: Passive authentication based on activity of daily living using heterogeneous sensing in smart cities. *Sensors*, 19(11):2466, 2019.

[18] Brendan Tran Morris and Mohan Manubhai Trivedi. A survey of vision-based trajectory learning and analysis for surveillance. *IEEE transactions on circuits and systems for video technology*, 18(8):1114–1127, 2008.

[19] Universal Orlando. Wand interactive hermione granger wand. https://shop.universalorlando.com/shop/harry-potter/collectibles/interactive-wands/interactive-hermione-granger-wand-1279649, 2020. Accessed: 2021-12-18.

[20] José Ramón Padilla-López, Alexandros Andre Chaaraoui, and Francisco Flórez-Revuelta. Visual privacy protection methods: A survey. *Expert Systems with Applications*, 42(9):4177–4195, 2015.

[21] Damião Ribeiro de Almeida, Cláudio de Souza Baptista, Fabio Gomes de Andrade, and Amilcar Soares. A survey on big data for trajectory analytics. *ISPRS International Journal of Geo-Information*, 9(2):88, 2020.

[22] Bharath Sudharsan, Dineshkumar Sundaram, John G Breslin, and Muhammad Intizar Ali. Avoid touching your face: A hand-to-face 3d motion dataset (covidaway) and trained models for smartwatches. In *10th International Conference on the Internet of Things Companion*, pages 1–9, 2020.

[23] Heung-Il Suk, Bong-Kee Sin, and Seong-Whan Lee. Hand gesture recognition based on dynamic bayesian network framework. *Pattern recognition*, 43(9):3059–3072, 2010.

[24] Shaohua Wan, Lianyong Qi, Xiaolong Xu, Chao Tong, and Zonghua Gu. Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications*, 25(2):743–755, 2020.

[25] Zhiqiang Xing and Demoz Gebre-Egziabher. Modeling and bounding low cost inertial sensor errors. In *Proceedings of IEEE/ION PLANS 2008*, pages 1122–1132, 2008.

[26] Peide Zhu, Hao Zhou, Shumin Cao, Panlong Yang, and Shuangshuang Xue. Control with gestures: A hand gesture recognition system using off-the-shelf smartwatch. In *2018 4th International Conference on Big Data Computing and Communications (BIGCOM)*, pages 72–77. IEEE, 2018.