Group 10 Final Project: Marching Band Tracker

CS 328, UMass Amherst, Fall 2017

Mary Buckler, Tim Liew, Sophie Manum

1 Inspiration

Our motivation for this project was from Mary being on the UMass Marching Band. In addition to playing the music and moving at the right time, it is essential to have your hands in the right position. It is sometimes difficult to be able to practice on your own since you might not have the drum major or setting a metronome yourself might be difficult, so we created this app to track your motions. It has a pre-set (but that can be changed) set of motions and times that can be changed based on the song you are practicing. This will then keep track of the position you are supposed to be in and classify what position you are actually in and send that data back to the phone. If the actual and expected positions are different, it will send a buzz to let you know that you are in the wrong position and that you need to change to the next one.

2 Data Collection

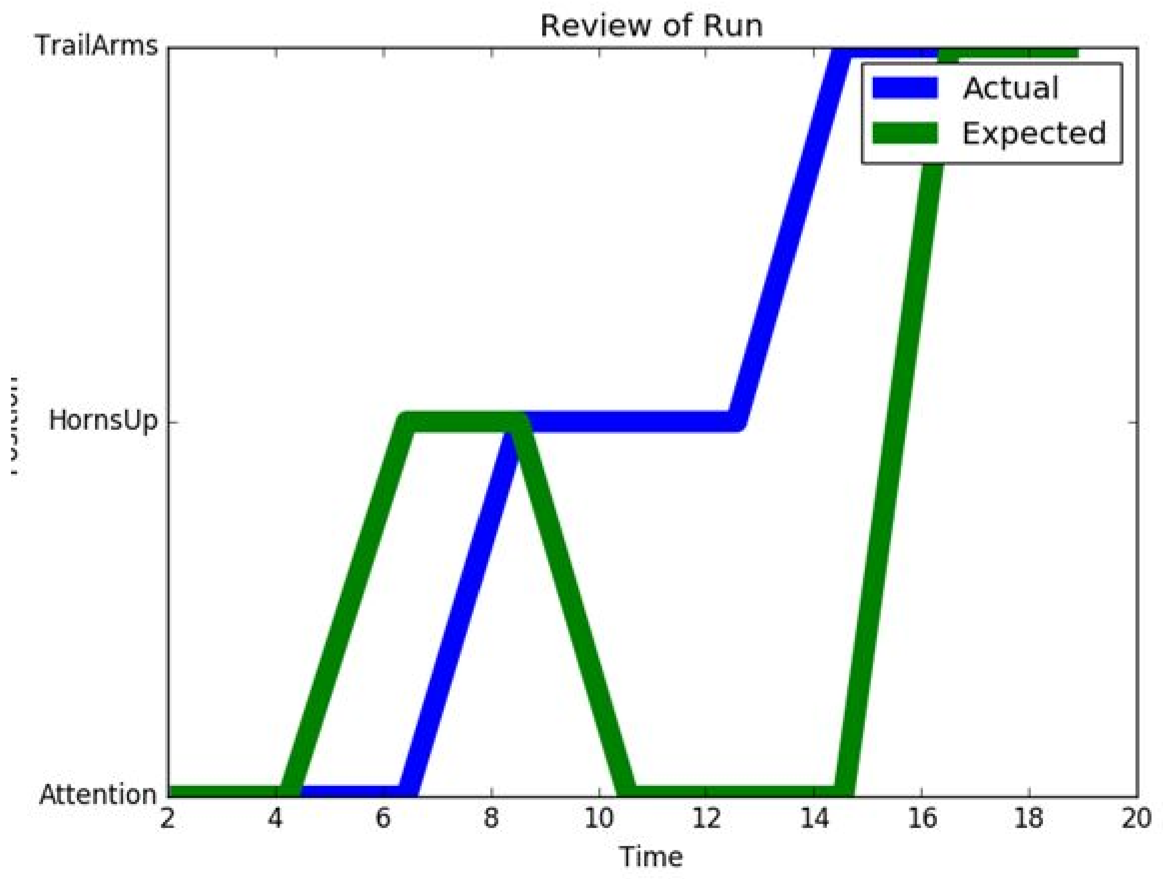
Data collection for this was very simple, since in marching band there really should be no variation from where you are, who you are, your level of physical fitness etc. Every person is required to stay stationary in the same exact position. This enabled us to not need that much data and there was little noise, but still a some because there is never no noise). We took about 3 minutes of each position using the accelerometer in my living room since it would not change much if outside. The one thing we wished we had time for, would be to be able to accommodate for other instruments as well. It would have been easy to accomplish, since we would just need to take data of attention, horns up, and trail arms for the correct positions while using other instruments. Even though Mary is a flutist, she does not know a lot about the positions for other instruments. If we had more time, it would have been great to ask other people from different sections in the marching band if we could collect data from them.

3 Challenges

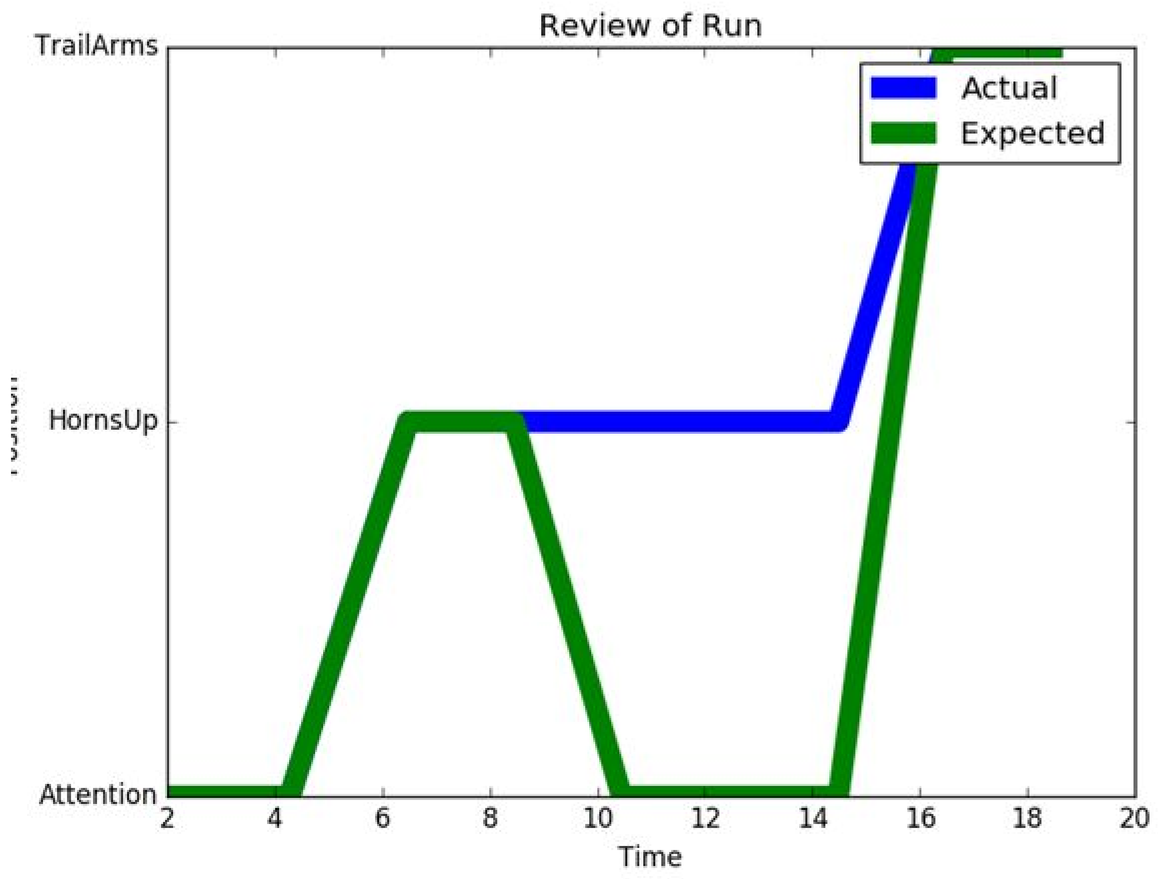
We had many challenges with this project. We had hoped to use the Android Wear since this would be the most practical assuming that most people would probably not want to strap their phone to their arm. The phone might also get in the way of playing an instrument. However, after working with the watch for a very long time, we were unable to get the connection to work. This was due to the fact that we started off on the wrong foot when we were trying to create the Android Wear app. We realized that we did not extend the correct classes that were supposed to connect the app to the server after referring to the starter code from our previous projects. The process to figure out the connection between the app and the server took us a lot of time. Thus, we decided to just stick to using a mobile phone despite having to deal with the awkwardness of strapping the phone to our arm. We originally also had thought we wanted to use the gyroscope for this, however the sample rate was much too low, and we found that the accelerometer worked better anyway. We found that in our final app, the sample rate makes a huge difference. The best case is to have a very high sample rate so the phone can inform you if you are in the wrong position quickly and there is little delay. However sometimes for the run you might only get one sample within that position's time and it is difficult for it to actually be helpful.

4 Final Product

We were able to reuse the connection between the phone and python code from A2.5 which enabled us to focus more on the added functionality to the classification. The classification works the same way just with new positions. To have the set time to be in each position, we took the start time that the python started, and added time onto that to set when the change should be happening. We did this with the time and not the number of samples seen to avoid the issue with the difference in sample rates. Once a sample has been sent from the phone, the python code classifies the position obtained, and checks to see what position it was supposed to be. After that, the expected and actual positions are sent back to the phone and they are appended to list with their timestamp. Once the phone receives the update, the expected and actual positions will be shown on the screen. If the two positions differ (the person is in the wrong position) the phone will buzz to notify the user that they are wrong and have to change their position. Once the sequence is over, the python generates a plot using matplotlib of the time vs expected and actual positions for the user to review their run. Examples of these are shown below.



In this run you can see that the user was correct in the beginning, changed too late from attention to horns up, then never went back to attention, but went early into the trail arms position.



In this run you can see that they did better, however they stayed in the horns up position when they were supposed to have changed to attention, but they met back up by going straight to trail arms.