Penn Project Report
Machine Learning in Real Time Character Recognition

Abstract

Implementation and design of machine learning algorithms to learn characters in real time through IMU and gyro acquisitions on a regular pen.

1. Prototype Design and Programming

This stage consisted mainly of the design of a working prototype to start the data gathering process. The challenge consisted in having a non bulky pen that could stream gyro and IMU data through WiFi to the computer so that the data could then be processed by a more powerful computer.

1.1. Design

We first took all the ideas and components we needed to make this project work, and carried them on to sketches. Once we were pretty sure of the feasibility of the design and that it could contain everything we needed, we stepped into 3D designing a pen that could be easily 3D printed and could be manufactured rather quickly we the resources we have in AddLab. The result of the first version of the pen looks like this:

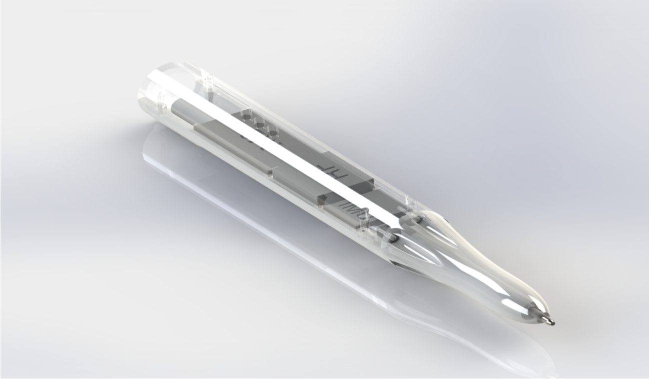

Figure 1. Rendering of the 3D printed model of the penn

As you can see, we have an on-board microprocessor, 3 axis accelerometer, 3 axis gyroscope, magnetometer, and WiFi module. It also has a very sensitive switch that can tel us when the pen is actually writing. On top of that, the pen has 3 LED lights to make a visual recognition of the states the pen was at. We also included in the real version, a switch to turn it on and off, and batteries.

2. Data Processing

2.1. Data Streaming

The first thing we had to do was to get raw data readings from the pen. After a couple of trials and for the sake of time saving, we decided that we would have a better first approach streaming the data out through USB instead of WiFi, to get a higher density of readings (WiFi is slower, saturates faster). Though the pen still has the functionality and all the programming to send the data via WiFi, we are more interested in getting the data out as soon as possible. Once that was done, we could pass to the next big task, data pre-processing.

2.2. Filtering the data

The raw data you get out of this readings is really noisy. It cannot be used in any way like it is. The data needs to be filtered to get the attitude estimation. We use a Kalman Filter to get the attitude estimation, which will give us the orientation of the pen relative to gravity and the page (Which has to be on a flat surface, and cannot be moving). We use this to subtract the gravity components in the vectors that they are appearing, so in this way we prevent drifting on those readings. We also get out of this the rotation of the pen relative to the page, which we will then use for further filtering.

2.3. More filtering... and trajectory tracking

Once we have the pre-filtered data, we can get a cleaner reading of the acceleration of the tip of the pen (relative to the page), which we will use to get a trajectory estimation. We integrate the acceleration twice, while at the same time, high-pass filtering any long time drifting of the readings in both yaw angle and velocities.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2.4. Implicated Dynamics

P: location of the IMU
O: origin of frame G on the page
frame G: SRT fixed to the page with origin O (ground frame)
frame M: SRT fixed to the pen at the IMU with origin at point P (pen frame)
Q: location of the pen tip
$\underset{\sim}{\mathbf{PQ}}$: vector from the IMU origin to the pen tip

The acceleration of the pen tip in the ground frame is given by:

$${}^{G}\underset{\sim}{a}^{Q} = {}^{G}\underset{\sim}{a}^{P} + {}^{M}\underset{\sim}{a}^{Q} + {}^{G}\underset{\sim}{\alpha}^{P} \times \underset{\sim}{\mathrm{PQ}} + 2\,{}^{G}\underset{\sim}{w}^{M} \times {}^{M}\underset{\sim}{v}^{Q} + \ldots$$
$$+ {}^{G}\underset{\sim}{w}^{M} \times \left({}^{G}\underset{\sim}{w}^{M} \times \underset{\sim}{\mathrm{PQ}} \right)$$

Since point Q (the pen tip) is fixed in frame M (the pen frame), ${}^{M}\underset{\sim}{a}^{Q}$ and ${}^{M}\underset{\sim}{v}^{Q}$ are both 0. The above equation simplifies to:

$${}^{G}\underset{\sim}{a}^{Q} = {}^{G}\underset{\sim}{a}^{P} + {}^{G}\underset{\sim}{\alpha}^{P} \times \underset{\sim}{\mathrm{PQ}} + {}^{G}\underset{\sim}{w}^{M} \times \left({}^{G}\underset{\sim}{w}^{M} \times \underset{\sim}{\mathrm{PQ}} \right)$$

The IMU measures accelerations and angular velocities of frame M relative to frame G expressed in frame M. Using the attitude estimate, these measurements can be rotated into frame G to produce ${}^{G}\underset{\sim}{a}^{P}$ and ${}^{G}\underset{\sim}{w}^{M}$. The angular acceleration ${}^{G}\underset{\sim}{\alpha}^{P}$ is obtained by numerically differentiating ${}^{G}\underset{\sim}{w}^{M}$.

The trajectory of the pen tip in the page frame is obtained by twice-integrating ${}^{G}\underset{\sim}{a}^{Q}$.

This does not work because of measurement noise and drift. Differentiation of ${}^{G}\underset{\sim}{w}^{M}$ greatly increases high-frequency noise in the acceleration estimate which must be low-pass filtered. More problematic, double-integration of the acceleration estimate compounds small steady errors which build up to large velocity errors and even larger position errors. This necessitates high-pass filtering of both the velocity and position signals. These high-pass filters are ineffective both at removing low-frequency drift and maintaining good signal quality resulting in unusably noisy and drift-filled position estimates

2.5. Preliminary Results

After some trials and errors, we finally de-bugged the code to get some character images that we could then use with the neural nets that we have already implemented on class, to get some character recognition via class classification. The results from all the estimations give us characters like the one below.

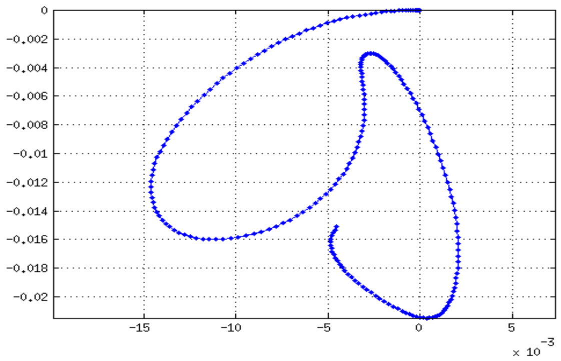

Figure 2. Representation results of the letter ”g”

2.6. Next Steps

After some thinking on the results we are getting and the complexity of the math behind all this, we are presented with two different options. Keep on pushing for this approach to get all the character representations in an image base, or shifting our Machine Learning algorithm to Hidden Markov Models which would potentially require a lot less data pre-processing and a more complicated machine learning implementation.

Acknowledgments

Attitude estimation Kalman Filter and initial approximation provided by **open-source** code from Sebastian Madgwick (July 2012), based on his paper **Estimation of IMU and MARG orientation using a gradient descent algorithm**.
For trajectory reconstruction, consulted Jeen-Shing Wang, Yu-Liang Hsu, and Jiun-Nan Liu’s paper entitled **An Inertial-Measurement-Unit-Based Pen With a Trajectory Reconstruction Algorithm and Its Applications**.