# Multi-modality intelligent gloves with feedback functions for metaverse

Feng Miaoyun

## Introduction

This project mainly requires that we should design an intelligent glove based on triboelectric nanogenerators (TENG) and bio-mimetic feedback to realize wearable human-machine interface (HMI) for metaverse, with the help of Internet of things (IoT) and artificial intelligence (AI).

With this requirement, my intended goal is that:

(1) Design a glove which can capture users' hand movements and show them in a cooperative virtual piano game.

(2) If users played a wrong tune, the glove would give them a feedback.

(3) The system can recognize users' basic sign language for games.

What I have done is that:

(1) Designed and improved the sensors (contact-separation & bending/stretching degrees) up to the fifth edition, using PDMS, EGaIn and Eco-flex 00-30.

(2) Designed the whole glove, which has 19 sensors in total.

(3) Modified signal amplification and filtering circuit, and differentiation is achieved. The PCB passed the tests.

(4) Designed a microcontroller based on STM32F4 to replace Arduino because it has more ADC channels, and some extra functional modules are added.

(5) Analyzed the collected signals with Matlab to design voltage integration algorithms.

(6) Learned ML and Python for data analysis systematically. The original signals are processed through 6 steps for reprocessing and are plotted, then specially enlarged for datasets. Five models are tested and the result is satisfactory.

(7) Designed a model of my own hand with Blender, whose action parameters can be obtained from the processed sensor data.

(8) Learned Unity 3D, and started to design the cooperative piano game.

The rest of the report explains what I did in detail and proposes future plans accordingly.

## System Flow

As shown in Fig.1:

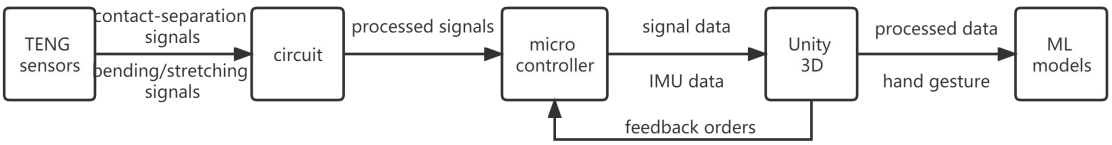


Fig.1 System Flow of the whole system

TENG sensors generate contact-separation signals or bending/ stretching signals. The signals go through amplification and filtering circuit, and then collected by ADC channels of the microcontroller. The microcontroller sends the collected data and IMU data to the

computer through wireless serial port. Unity 3D reads the data and transform it into Euler angles of the finger knuckles or hand position and attitude, which is shown in the player scene as a whole hand. Unity also calls the python scripts as scheduled to recognize the hand gesture. Then, according to the interaction in the game, Unity sends orders back to the microcontroller through wireless serial port. The microcontroller gives feedback to the player through ERM vibrators.

## TENG Sensors

The first edition of TENG sensor for monitoring the bending degree of finger knuckles is shown in Fig.2. It is inspired by *<<Soft Modular Glove with Multimodal Sensing and Augmented Haptic Feedback Enabled by Materials' Multifunctionalities>>*. The first layer is a 12mm*12mm cuboid which is 2mm thick with a cylindrical digged-out 10 mm in diameter and 1 mm in depth, it is made of Eco-flex 00-30. The second layer is a 11mm*11mm nickel fabric with a 15mm tail to connect to the wire. The third layer is a Eco-flex 12mm*12mm cuboid. Combine them together and apply a coat of Eco-flex to seal.

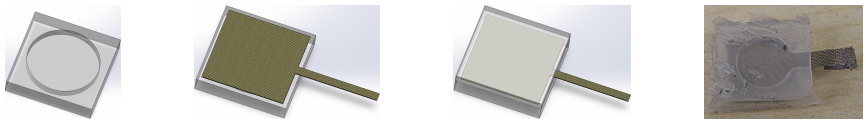

Fig.2 TENG sensor version 1

The first edition can only detect contact-and-separation. The second edition is inspired by *<<Augmented tactile-perception and haptic-feedback rings as human-machine interfaces aiming for immersive interactions>>*. It's pretty much the same as the first edition on the whole, but some tapered protrusions 1 mm high are added to the first layer. If bent, the contact area will get larger, which can be reflected in the change in the current signal.

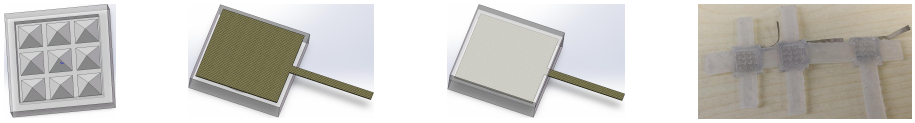

Fig.3 TENG sensor version 2

When testing version 2, I found that the nickel fabric may deviate from its original position because it's poor extensibility. If you use the sensor many times, the nickel fabric may even curls up.So I plan to change it into some stretchable positive materials. In the third edition, that is EGaIn/Eco-flex mixture. It is inspired by *<<A stretchable self-powered triboelectric tactile sensor with EGaIn alloy electrode for ultra-low-pressure detection>>*. I changed the material of the third layer into PDMS, because it is more electronegative.

Fig.4 TENG sensor version 3

When using version 3 to design the whole glove, I found that a round sensor is not easy to bend when finger knuckles bend. So the forth edition changed it to long strip shape (8mm* 30mm).

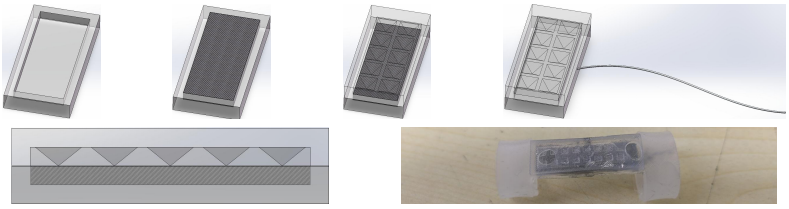

Fig.5 TENG sensor version 4

When testing version 4, I found that I made a big mistake. The EGaIn/Eco-flex mixture should serve as electrode instead of triboelectric positive layer. When collecting signals, I originally thought that the sensor should fit the knuckle to synchronize deformation. But actually I followed this paper:*<<Self powered glove-based intuitive interface for diversified control applications in real/cyber space>>*.

Now I'm designing version 5, trying to solve the former problem. I plan to make it out before next semester starts.

Fig.6 shows TENG sensors for monitor stretching. The first 2 pictures are the first version, and the last ones are the second. For the first version, clamp the sensor between two fingers, if fingers spread or pressed together, the different layers of the sensor will contact and separate, thus generating signals. But it may limit finger movement and the measuring range is quite limited. The second version is inspired by *<<A Soft Sensor-Based Three-Dimensional (3D) Finger Motion Measurement System>>*. It doesn't use PDMS because it has poor stretchability. By stretching the sensor, it will generate satisfactory signal.

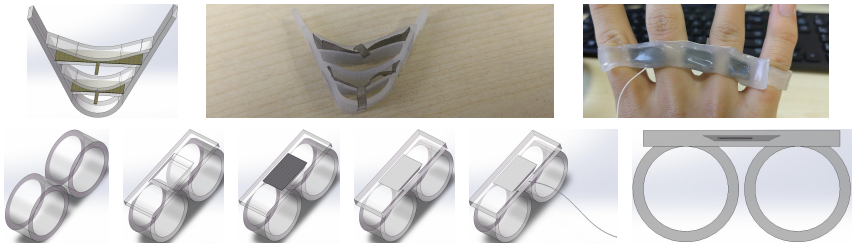

Fig.6 TENG sensor for monitor stretching

For the whole hand, there are 5 contact-and-separation sensors located at distal phalanges (DP)/distal phalanges (DP); 8 bending sensors located at proximal interphalangeal (PIP) and Metacarpophalangeal (MCP) of forefinger, middlefinger, ringfinger and littlefinger; 1 bending sensor located at interphalangeal (IP) of thumb; 1 bending sensor located at the palm side to monitor carpometacarpal (CMC) of thumb; 4 stretching sensors located between fingers. Figure.7 illustrates this, in which read means on the back of the hand side, blue means on the front of the hand side, circle means contact-and-separation monitor, elliptic means bending monitor, rectangle means stretching monitor.

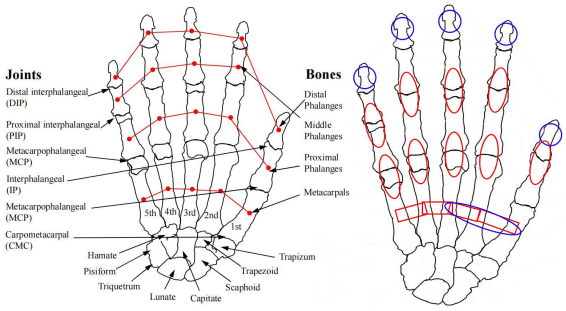


Fig.7 locations of sensors

## Amplification and Filtering Circuit

I modified the circuit mentioned in *<<Augmented ..... interactions>>* in the following aspects:

(1) Resistor values for voltage bias. For contact-and-separation, the amplitude of the positive and negative pulses is different, the ratio of the two resistor is 1:2; For bending/ stretching, it is 1:1.

(2) The way of amplification. In-phase amplification can not reduce the signal, but we should reduce the pulse of contact-and-separation in order not to burn out the ADC points. So I choose inverting amplification. So, amplify the bending/stretching signal and reduce the contact-and-separate signal, differentiating the modules.

(3) Added power supply filtering.

(4) Modified the cutoff frequency of the low-pass filter.

(5) Added feedback signal amplification circuit, using triode.

(6) Sockets that match the PCB of the microcontroller are designed to connect the two PCBs.

In Fig.8, the first picture is the original circuit, the second is my modified circuit, and the last, the 3D view of the PCB. The PCB is tested useful.

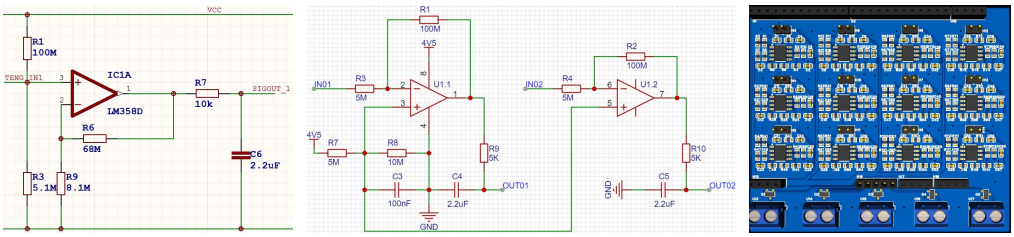


Fig.8 Diagrams of the amplification and filtering circuit

## Microcontroller

Fig.9 shows the schematic of the microcontroller.

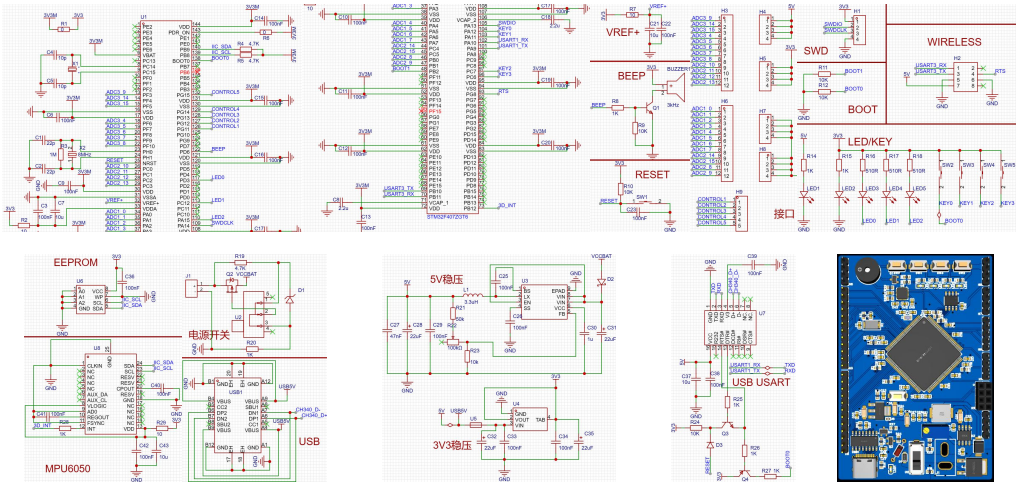


Fig.9 Schematic of the microcontroller

STM32F407ZGT6 has 24 ADC channels, some of which are used for other functions, and the remaining ones are enough for 19 sensors. In this microcontroller board, a 7.2 volt lithium battery serves as the power supply, going through the power switch, it will then be regulated to 5 volt, and then be regulated to 3.3 volt. Another kind of power supply

is through the USB port of a computer, it is mainly used for serial communication and downloading programs. The wireless part is used to send message of signals to the remote computer. The beep, led and key part is used for human-machine interface. The MPU6050 is used to monitor the basic location and attitude of the hand. The reference voltage of ADC is set as 3.3 volt. It means the max voltage of the circuit processed signal is 3.3 volt and the base voltage is 2.2 volt (3.3 *(2/(1+2))).

Fig.10 shows the flowchart of the microcontroller.

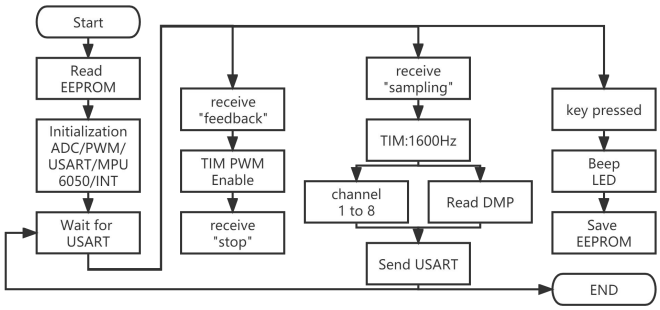


Fig.10 Flowchart of the microcontroller

Read EEPROM to get the settings recorded and initialize every modules. Wait for the computer to send wireless serial port signals. If receive feedback, let the ERM vibrators vibrate according to different frequencies. If receive sampling, it will collect the signals from 19 channels respectively and read from the DMP of the MPU6050, then send these messages to the computer. If key pressed, it will beep, and change the settings according to the key pressed and save it to the EEPROM.

I haven't tested the microcontroller board. I will do this and improve it in the winter vacation.

## Voltage Integration Algorithm

To analyse the signals, I firstly designed a software low-pass filter through Matlab, then plot the original and processed signal. In Fig.11, the upper one is the original signal and the lower one is the software low-pass filter processed signal. The first picture shows the signal generated by contact-and-separation, the second one shows the signal of bending, and it's not amplified and filtered by the signal processing circuit. This third one is that goes through the signal processing circuit. The last one shows the signal of sensors being stretched, and it's not processed.

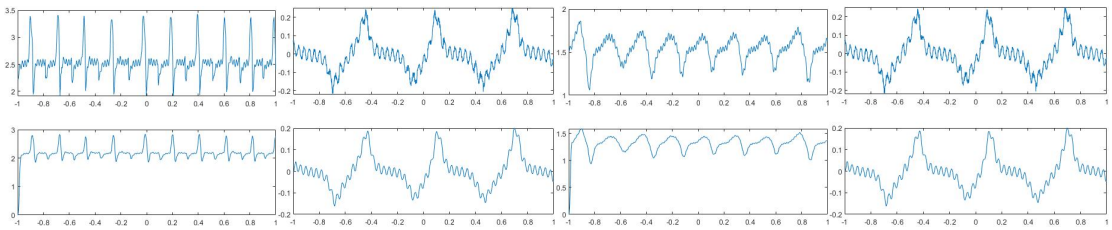

Fig.11 Matlab analysis for original signals

Below is the algorithm of processing contact-and-separation signals:

$$\text{Out} = 1: \qquad \left(V_{\text{signal}} >= V^{+}_{\text{threshold}}\right)$$

$$\text{Out} = 0: \qquad \left(V^{-}_{\text{threshold}} < V_{\text{signal}} < V^{+}_{\text{threshold}}\right)$$

$$\text{Out} = -1: \qquad \left(V_{\text{signal}} <= V^{-}_{\text{threshold}}\right)$$

We should try different $V^{\pm}_{\text{threshold}}$ to avoid wrong judgement.

Then is the algorithm of processing bending/stretching signals (inspired by *<<Haptic-feedback smart glove as a creative human-machine interface (HMI) for virtual/augmented reality applications>>*): Let the signals go through the software low-pass filter (translate it into C#), then, get V zero through a large number of signal data.

$$\text{Vzero} = \frac{\sum_{0}^{T} V_{\text{signal\_test}}}{T}$$

Bend the finger to the maximal angle $A_{\max}$, which is known from *<<The anatomy of the dorsal aponeurosis of the human finger and its functional>>*, then get $V_{\text{signal\_max}}$.

$$V_{\text{integration\_max}} = \sum_{t_1}^{t_2}\left[\left(V_{\text{signal\_max}} - V_{\text{zero}}\right) \bullet \Delta t\right] \propto A_{\max},$$

where $V^{t_1}_{\text{signal\_max}} = V^{t_2}_{\text{signal\_max}} = V_{\text{zero}}$, for $t_1 < t < t_2$, $V_{\text{signal\_max}} > V_{\text{zero}}$

Integrate the voltage signal, due to their linear relationship, the degree of bending and stretching can be obtained from the integral value. Then, accumulate the small angles get from integration, we can get the degree our finger joints bend or stretch.

$$V_{\text{integration}} = \left(V_{\text{signal}} - V_{\text{zero}}\right) \bullet \Delta t$$

$$\Delta_{\text{angle}} = \frac{V_{\text{integration}}}{V_{\text{integration\_max}}} \bullet A_{\max}, \quad \text{Angle} += \Delta_{\text{angle}}$$

There is still a problem: Though I do not do any movement, charge still accumulates or dissipates over time, resulting to the integrated signal having a linear offset with a slowly changing slope. As to the slowly changing slope, we should write a program that continuously detects the current slope and eliminates it. I will do this after I make out the whole glove and collect enough data.

## Machine Learning

I plan to use the original signals of changing from one gesture to another, and the corresponding voltage integration signals to recognize the hand gesture. This is inspired by *<<AI enabled sign language recognition and VR space bidirectional communication using triboelectric smart glove>>*.

After doing an action, at this moment my gesture is gesture A, it's probably that gesture A is transformed from gesture B, also it's possible that gesture A is transformed from gesture C. If I have 10 gestures to recognize, actually I should classify 90 different actions (n*(n-1)).

For testing the idea, I used 4 TENG sensors to collect the data at the same time, which

are respectively tied to the Proximal interphalangeal (PIP) of forefinger, middle-finger, ring-finger and little-finger. After going through an operational amplifier and a first-order low-pass filter, the current signals are changed into much larger voltage signals. I collected signals of 4 gestures: one, two, three, four, and they are labeled as: 0, 1, 2, 3. Added the gesture fist, I got 4 actions for each gesture, so the total actions' number is 16. We collected 2 times of them, with the last time failed, getting a dataset of 31 CSV files.

Fig.12 illustrates the pipeline of signal processing.

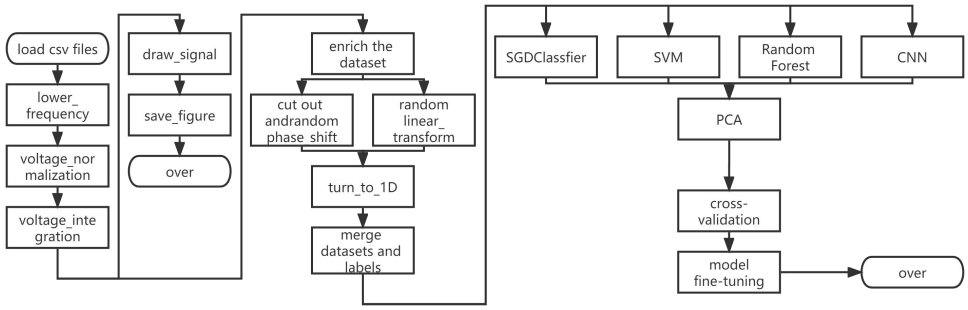


Fig.12 Flowchart of the python scripts

(1) Load data from CSV files, get an array of dataframe.

(2) The collecting frequency is 400Hz, much higher than need. Decrease the frequency to 200Hz for plotting and 50Hz for learning. This can greatly decrease the number of features.

(3) Normalize the data. Considering that different sensors have different voltage output range, it is reasonable to unify them into a unified scope.

(4) For data of 200Hz, plot them out to get an intuitive understanding. A sample plot is shown in Fig.13.

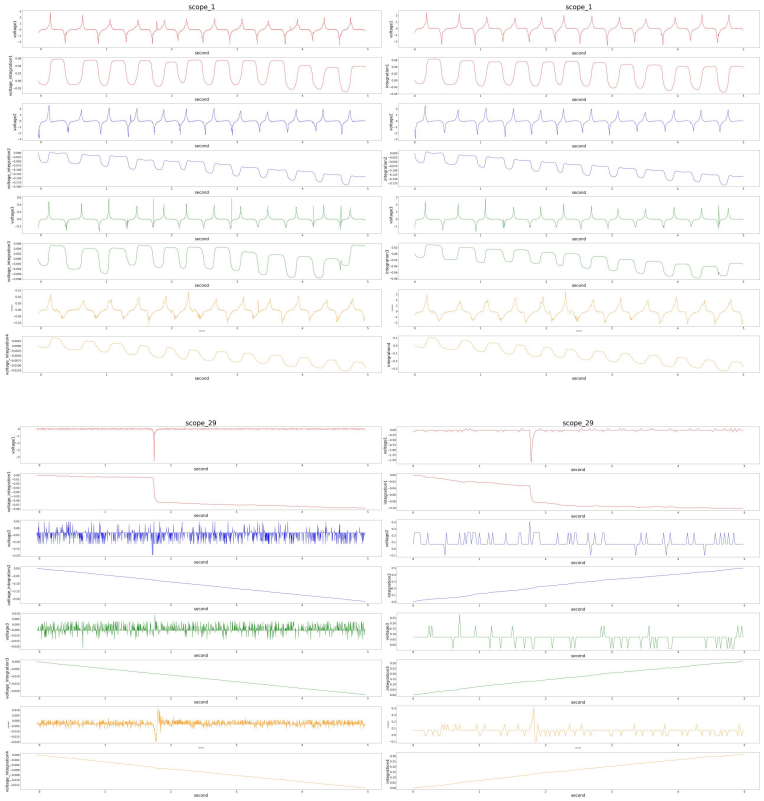

Fig.13 Plots of original (left) and processed (right) signals

(5) For data of 50Hz, cut them down. These data are collect in 5 seconds, much longer than an action costs, so just select the useful pieces, making the system responses more quickly.

(6) To enlarge the dataset, copy the cut-up data with Gaussian noise added.

(7) We learn from the relevant papers of TENG that, The angle a finger joint bends is positively correlated to the integrated value of the voltage signal. So integrate the data and serve them as new features.

(8) We found that due to charge accumulation and dissipation, the integrated data signal has a linear bias with slowly changing slop. To further enlarge the dataset, we come out an idea that we can change the slop of the linear bias and add it to the integrated signal.

(9) Turn the processed dataframe to 1 dimension, get a 768-feature array.

(10) It is worth mentioning that, different actions cost different time, so we should extend the shorter data to be as-long-as the longest one, which can also enlarge the dataset because we introduced random phase shift while extending the shorter data.

I tried the following models: SGD Classifier, Logistic Regression, MultinomialNB, K Neighbors Classifier, Decision Tree, Random Forest Classifier, Gradient Boosting Classifier and SVM, and tuned some parameter. The best result is 0.99692, seems over-fitting. I will go on collect more data and more gestures in the coming semester, and tune the models.

I also plan to make use of machine learning to adjust some baseline according to the linear regression, because different people have different sizes and shapes of hand, even that, the signals are different the different times you wear the same glove. I will do this after I realize the former idea.

## Blender modeling and Unity 3D

I modeled my own hand in Blender, a software for modeling. Then, deal with it in Unity 3D, setting up parent-child relationships.

Simply speaking, the angles my finger joints bend is in correspondence to the Euler angle of the finger joints created in the hand model. In Fig.14, the Euler angle of the finger joints has already been modified.

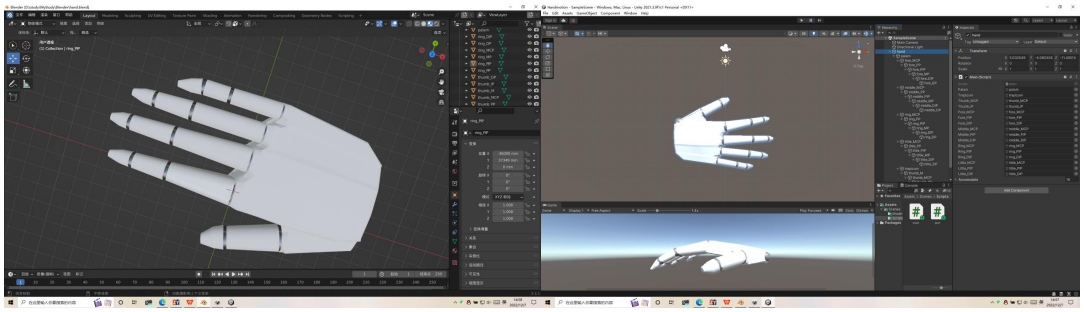

Fig.14 The model of my hand and editing in the Unity 3D

Also, the DMP data controls the movement and attitude of the palm. By reading serial port data through C#, we can control the hand to move. Its flowchart is shown in Fig.15.

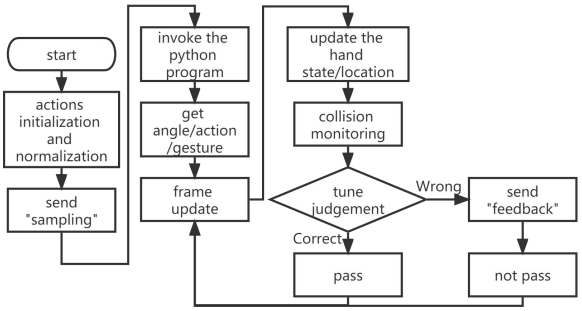


Fig.15 Flowchart of Unity 3D

The unity 3D get the first-hand signal, it will deal with it with the algorithm mentioned above. At the same time, after accumulating to the specified number of signals, it will call the python program to do the classification. With these results, unity 3D will update the frame to show them in the players' horizon.

To design the piano game, I need to model a piano and do collision monitoring, if collision occurs, the sound will play. And if you play a wrong tune, unity 3D will ask the microcontroller to give you a feedback.

For multiplayer performance, the unity 3D should equip 2 functions: The first one is that it can read from 2 or more different serial ports and deal with the data separately. The second one is that it can get messages coming from another game running on another computer through communication protocols.

How to design the unity game still needs careful consideration. I will do this at the last part of the project.

## Future Plan

(1) Design and test sensor version 5, if satisfactory, design the whole glove and try to improve it's stability.

(2) Improve the PCB of microcontroller, testing it's all functional modules, including HMI design.

(3) Code on the microcontroller, firstly test it on a development board, then my board.

(4) Collect enough data of the new glove, at least 10 gestures (90 actions), and each at least 20 times.

(5) Process these original data, improving the preprocessing program.

(6) Try different ML and CNN models and tune them.

(7) Model the piano.

(8) Redesign the feedback module, re-select the ERM vibrator.

(9) Code and test the communication between Unity 3D and microcontroller

(10) Improve the voltage integration algorithm.

(11) Code on the collision detect and test it.

(12) Try using double serial ports on the computer and some communication protocols to realize multi-players.

(13) Complete the whole Unity 3D project.

(14) Read more papers and reviews.