Table of Content

[1.0 Introduction 1](#_Toc42608694)

[2.0 System Description 3](#_Toc42608695)

[2.1 Open Pose 3](#_Toc42608696)

[2.2 Face Recognition 4](#_Toc42608697)

[2.3 SVM 5](#_Toc42608698)

[2.3.1 What is SVM 5](#_Toc42608699)

[2.3.2 Data Pre-processing 6](#_Toc42608700)

[2.3.3 Result 8](#_Toc42608701)

[2.4 MQTT 10](#_Toc42608702)

[2.5 UI 11](#_Toc42608703)

[3.0 Algorithm 13](#_Toc42608704)

[4.0 System Diagram 14](#_Toc42608705)

[4.1 System Flow Diagram 14](#_Toc42608706)

[4.2 Data Processing Diagram 15](#_Toc42608707)

[4.3 Class Diagram 16](#_Toc42608708)

[5.0 Performance 17](#_Toc42608709)

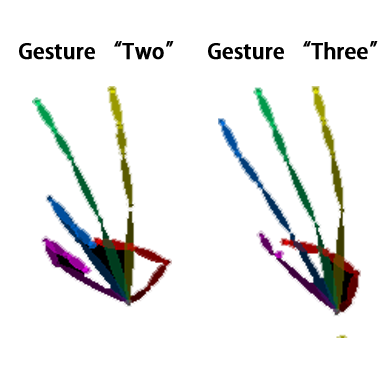
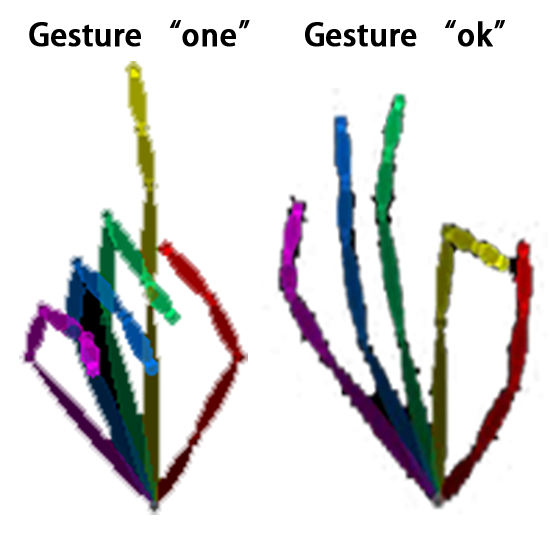
[6.0 Installation 17](#_Toc42608710)

[7.0 Conclusion 18](#_Toc42608711)

[8.0 Reference 18](#_Toc42608712)

# Introduction

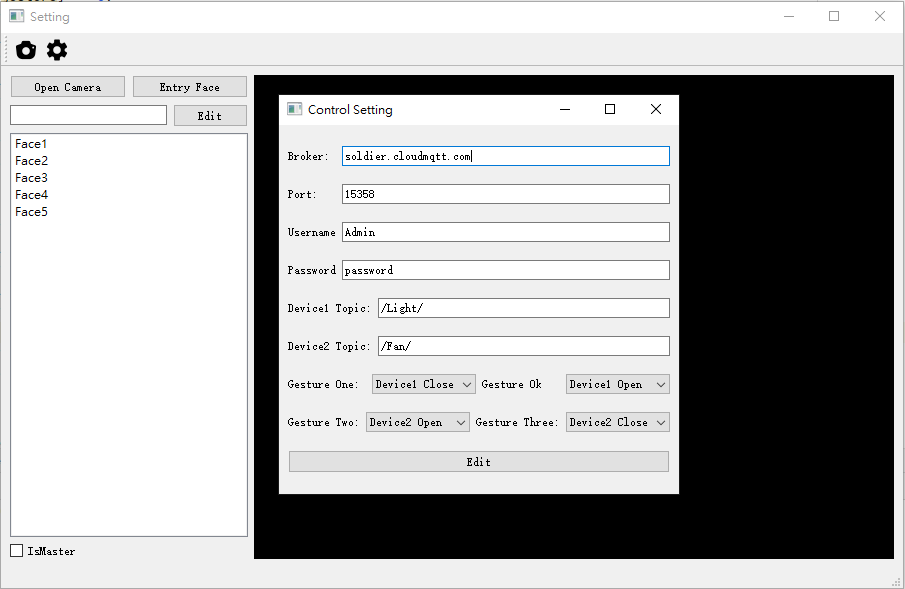
The programming language of the project is Python. Open pose, face recognition model was used in this project. In this project, users can use gestures to achieve the opening and closing actions to control the appliances. To control the appliances, user needs to raise his right arm and make a gesture for 3 seconds in front of camera. In order to recognize gestures, we use open pose model to obtain key points of the hand, and then use SVM to predict hand gestures. Finally, it could control home appliances by the result.



In the case of multiple people, the face recognition function is also added to identify the skeleton of different users. Users can set face which user need as master to gain priority control. If master and other user control the appliances at the same time, the system will execute the master command. In the situation that somebody is controlling the appliances and master is not, the system will execute somebody’s command.



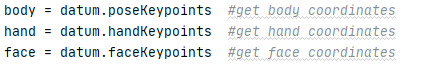
Of course, user could also set the face and control information by UI window. User could entry the face information by this window. In addition, the appliance is controlled by MQTT, users can also change the MQTT server information to control different devices. And also, it could change the function of gestures in this window.

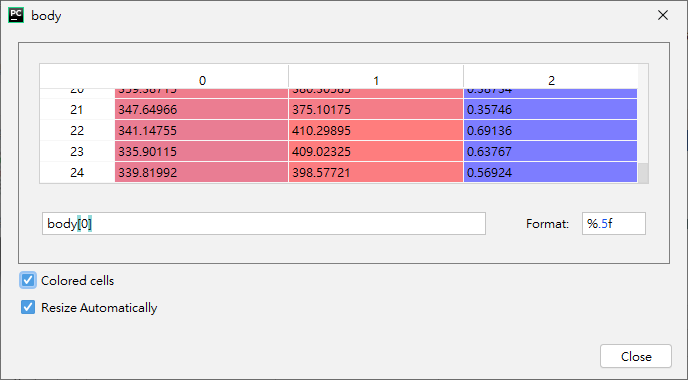


# System Description

## Open Pose

The project uses the open pose [1] model to obtain the skeleton of the human body. This model is created by Carnegie Mellon University. It could achieve 2D real-time multi-person key point detection. Not only the body, it can also recognize the skeleton of the hand and face. With the skeleton of the hand, gesture recognition becomes simple. But at the same time, open pose model has certain requirements for computer hardware such a GPU with sufficient performance. In this project, I used the python API feature of the model and export the skeleton coordinates as an array. At last, use SVM to predict with the hand skeleton and get the result.

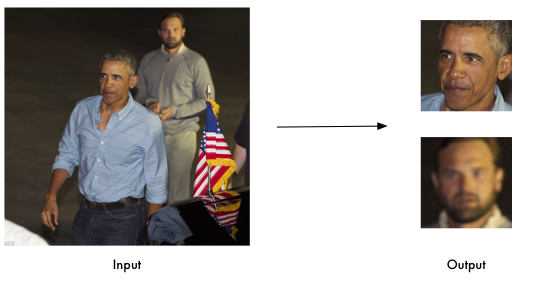


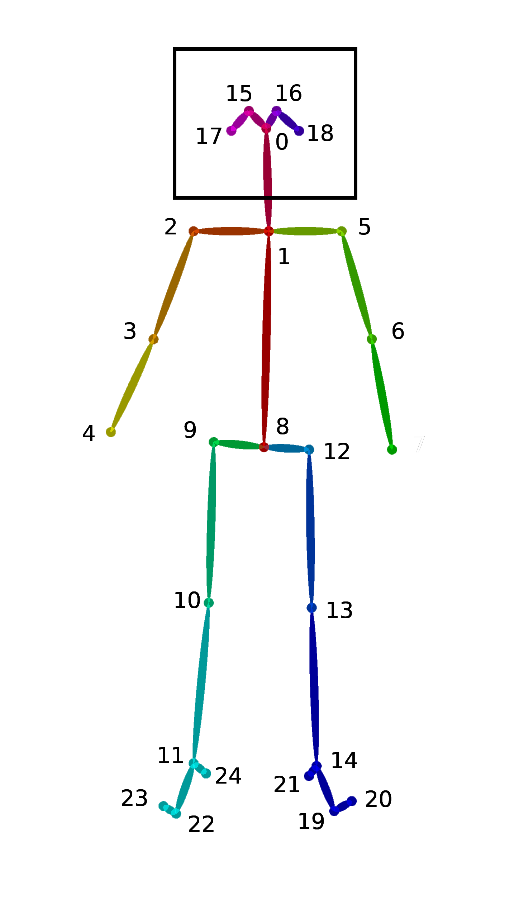


Body[i] mean different people in the frame. Column [0] is x axis, Column [1] is y axis, Column [2] is confidence.

## Face Recognition

Face recognition [2] model can find all the faces that appear in a picture. First, it can read the pre-photographed picture, and then record the face information in the picture. When there is a new frame, find all the faces in this frame and compare with the exist face. Finally, return the recognition result.



As I said before, face recognition is to identify the owners of different skeletons, but how to bind the face and skeleton together? The method is to find the position of the head point0, 15, 16, 17, 18. Then use OpenCV’s ‘boundingRect’ method to get the frame of the head. Finally, pass the head frame to face recognition model and return the result.

## SVM

### What is SVM

Support vector machines so called as SVM is a supervised learning algorithm which can be used for classification and regression problems as support vector classification (SVC) and support vector regression (SVR). It is used for smaller dataset as it takes too long to process. In this project, we will be focusing on SVC.

This project will use the polynomial as the SVM kernel. At present, the mainstream kernels used by SVM are Polynomial, linear and rbf. After testing, the best performance in this project is Polynomial. In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.

|  |  |  |  |
| --- | --- | --- | --- |
|  | rbf | linear | poly |
| Time | 0.03s | 0.0012s | 0.00099s |
| Precision | 0.69 | 1.00 | 1.00 |

### Data Pre-processing

Data preprocessing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis. Often, data preprocessing is the most important phase of a machine learning project, especially in computational biology. [3]

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data preprocessing includes cleaning, Instance selection, normalization, transformation, feature extraction and selection, etc. The product of data preprocessing is the final training set.

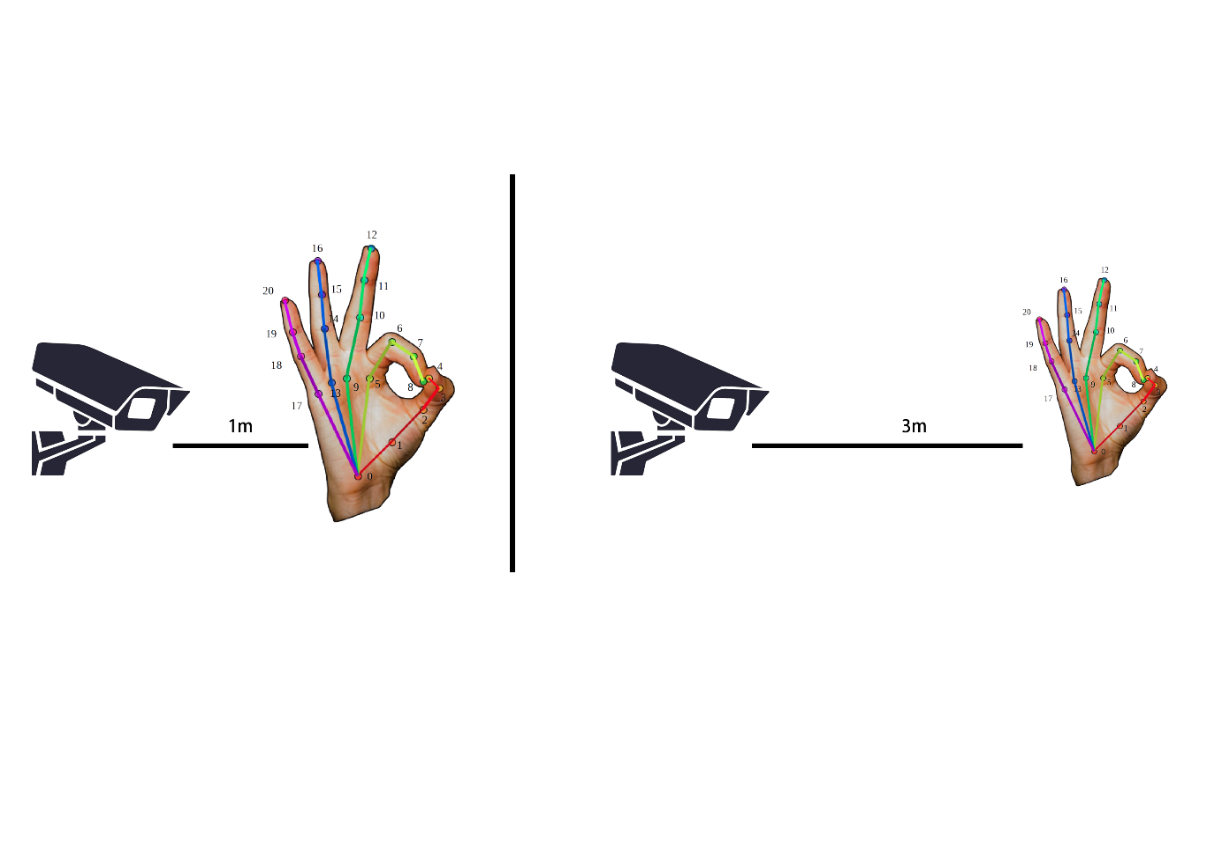
In this project, what we get from the open pose model is the human skeleton coordinates, we should consider how to let the SVM to predict the results effectively. Because the coordinates are used to predict, the coordinates obtained by the user's gestures at different angles or different positions will be different.

coordinates

|  |  |  |
| --- | --- | --- |
| point | x axis | y axis |
| 0 | 321.333 | 244.331 |
| 1 | 412.798 | 253.221 |
| … | … | … |

Therefore, it is necessary to consider that the SVM model can predict the correct gesture result when the user is at different angles and positions.

Considering data processing, we need to think about factors that affect the prediction results from several directions. The first is the position. Because the position of the hand is uncertain, the coordinates of the hand cannot be used as the feature to SVM training. We can consider that using vectors as features. Then we can imagine that the vector contains direction and size. When the user is far from the camera, the size of the vector is smaller; when the user is close to the camera, the size of vector is larger, so the vector cannot be used as a feature of the SVM.



Then we infer from the above that we can only use the direction as the feature to training SVM. Because this condition is not affected by factors such as the position of the hand and the distance from the camera. The data that can possess this characteristic is the unit vector. We can calculate the unit vector from the coordinates of the skeleton of the hand.

And we can get the unit vector of the skeleton through this formula.



According to the order of key points of hand, we can divide into 20 vectors.

|  |  |  |  |
| --- | --- | --- | --- |
| Point to Pair | | | |
| point0, ponit1 | point1, ponit2 | point2, ponit3 | point3, ponit4 |
| point0, ponit5 | point5, ponit6 | point6, ponit7 | point7, ponit8 |
| point0, ponit9 | point9, ponit10 | point10, ponit11 | point11, ponit12 |
| point0, ponit13 | point13, ponit14 | point14, ponit15 | point15, ponit16 |
| point0, point17 | point17, ponit18 | point18, ponit19 | point19, ponit20 |

### Result

When we get the data, after data cleaning, pre-processing and wrangling, the first step we do is to feed it to an outstanding model and of course, get output in probabilities. How in the hell can we measure the effectiveness of our model? Better the effectiveness, better the performance and that’s exactly what we want. And it is where the Confusion matrix [4] comes into the limelight. Confusion Matrix is a performance measurement for machine learning classification.



But the situation in the picture above is the case of only two classes. In this project, there are 4 gestures. The total data size is 442, and 70% of it is used as training data. There are 136 testing data and the data will be randomly shuffled. So, the Confusion Matrix for this project should be as follows

|  |  |  |  |
| --- | --- | --- | --- |
| “OK” gesture | Actual Values | | |
| Predicted Values |  | Positive | Negative |
| Positive | 37 | 0 |
| Negative | 0 | 96 |

In the “OK” gesture, set the result of the gesture with “OK” label as positive, and other as negative. We can calculate the Recall rate of “OK” gesture based on these results.

OK Recall = TP / (TP + FN) = 37 / (37 + 0) = 1.00

OK Precision = TP / (TP + FP) = 37 / (37 + 0) = 1.00

|  |  |  |  |
| --- | --- | --- | --- |
| “ONE” gesture | Actual Values | | |
| Predicted Values |  | Positive | Negative |
| Positive | 33 | 0 |
| Negative | 0 | 100 |

In the “ONE” gesture, set the result of the gesture with “ONE” label as positive, and other as negative. We can calculate the Recall rate of “ONE” gesture based on these results.

ONE Recall = TP / (TP + FN) = 33 / (33 + 0) = 1.00

ONE Precision = TP / (TP + FP) = 33 / (33 + 0) = 1.00

|  |  |  |  |
| --- | --- | --- | --- |
| “TWO” gesture | Actual Values | | |
| Predicted Values |  | Positive | Negative |
| Positive | 27 | 0 |
| Negative | 0 | 106 |

In the “TWO” gesture, set the result of the gesture with “TWO” label as positive, and other as negative. We can calculate the Recall rate of “TWO” gesture based on these results.

TWO Recall = TP / (TP + FN) = 27 / (27 + 0) = 1.00

TWO Precision = TP / (TP + FP) = 27 / (27 + 0) = 1.00

|  |  |  |  |
| --- | --- | --- | --- |
| “THREE” gesture | Actual Values | | |
| Predicted Values |  | Positive | Negative |
| Positive | 36 | 0 |
| Negative | 0 | 97 |

In the “THREE” gesture, set the result of the gesture with “THREE” label as positive, and other as negative. We can calculate the Recall rate of “THREE” gesture based on these results.

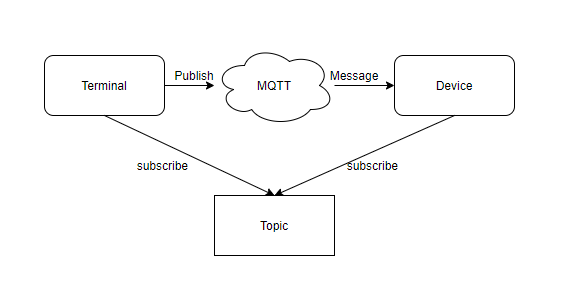
THREE Recall = TP / (TP + FN) = 36 / (36 + 0) = 1.00

THREE Precision = TP / (TP + FP) = 36 / (36 + 0) = 1.00

## MQTT

MQTT is an open OASIS and ISO standard lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, lossless, bi-directional connections can support MQTT. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited.

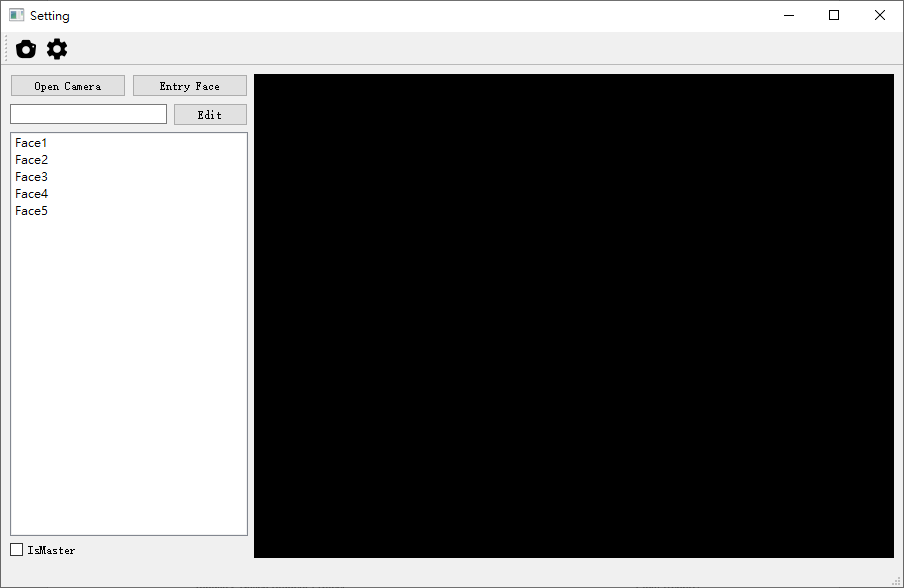
In this project, MQTT is mainly used to communicate between the terminal and the device. Because MQTT saves power and you it can be used when there is network. MQTT is widely used in the Internet of Things. The example applied to this project is that different devices subscribe to different topics. When users make different gestures, they will push different message to different topics, and the device will take corresponding actions after receiving the message.

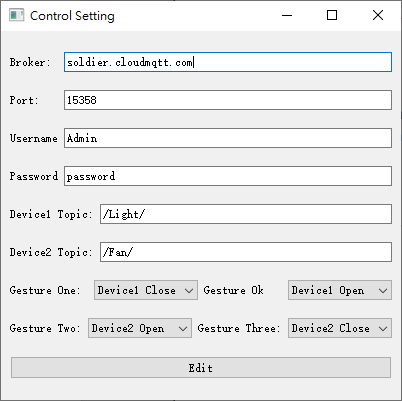


## UI

The UI of this system is made using PyQt. PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt supports Microsoft Windows as well as various flavors of UNIX, including Linux and MacOS. Since the main programming language of this project is python, the language used by PyQt is also python. Pyqt could coincide with this project

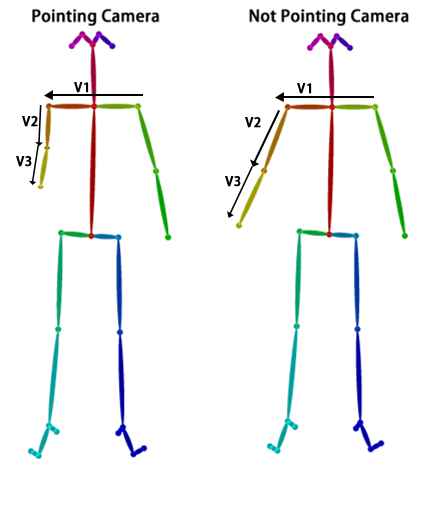
Since all the key information in this project such as face name, master permissions, MQTT server and hand gesture information are set through the configuration file. The main function of the UI is to modify the configuration file.



In this window, the user can select the list on the left to modify or enter face information. And the input box above is to enter the name of the face. When the user clicks "open camera", the black area on the right will display the current screen. When user turn on the camera, press the "entry face" button to enter the face information which in the current frame. At this time, the user needs to click the button again to confirm the entry. Before the user presses the button for the second time, the status bar will show whether the face contained in the current frame matches the one just entered to avoid the failure of the entry.

And this window can modify the information of MQTT server and the function of different gestures.

# Algorithm

As mentioned earlier, users need to raise their arms to trigger a 3-second timer. So how does the system identify whether the user raised his arm?

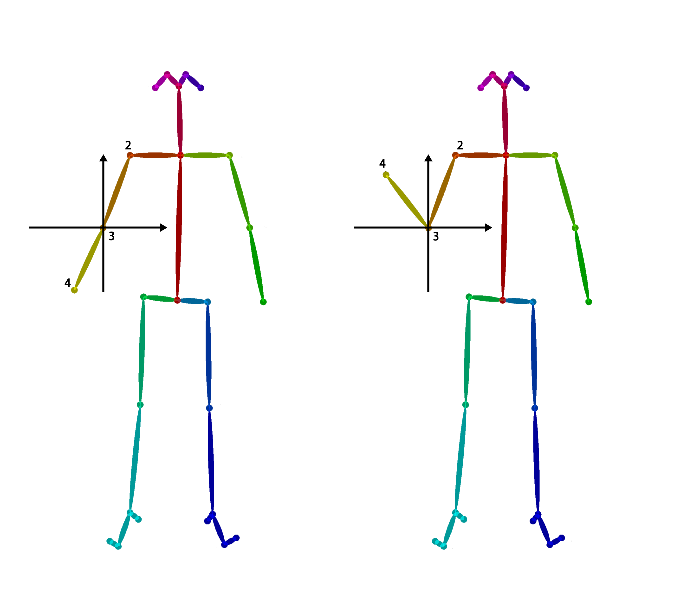
In Smart Home Control v1.0, the principle is that when the length of the arm is less than K times the length of the shoulder, the user is pointing at the camera, which triggers the control program.

Smart Home Control v1.0

The formula is that |V2| + |V3| < K|V1|.

But in Smart Home Control v2.0, which is this project. The skeleton of the hand cannot be recognized when the user points to the camera. Because open pose uses the wrist as a reference point to recognize gestures. When the user points to the camera, the wrist is blocked and the hand is naturally not recognized.

Down

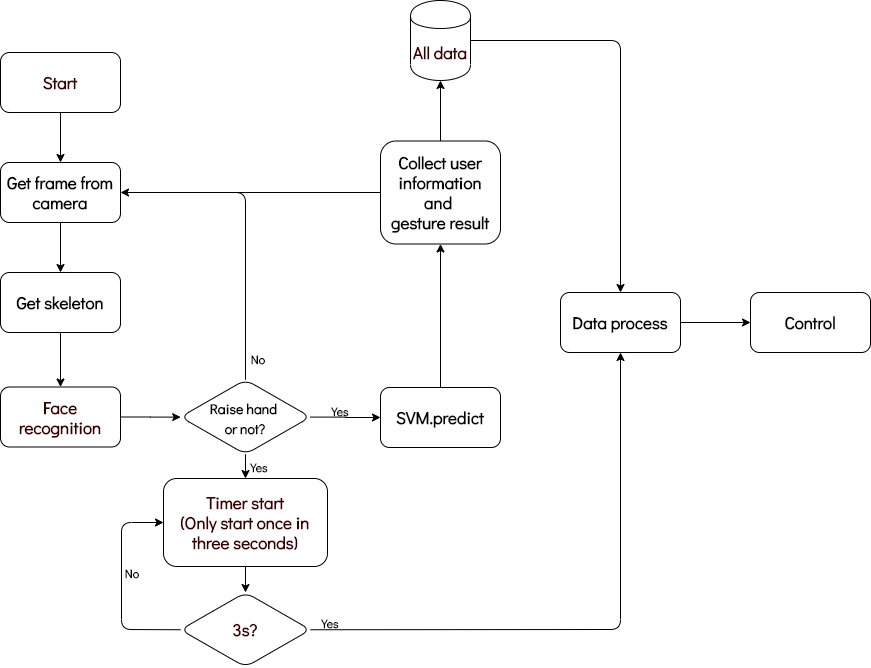
We can use another way to trigger the control program is to raise the arm. Make a rectangular coordinate system with point 3 as the origin. If point 2 is in the first quadrant and point 4 is in the second quadrant, the user raises his arm. By judging the quadrant where the skeleton of the hand is located, it can be recognized whether the user raises the arm.

Up

Smart Home Control v2.0

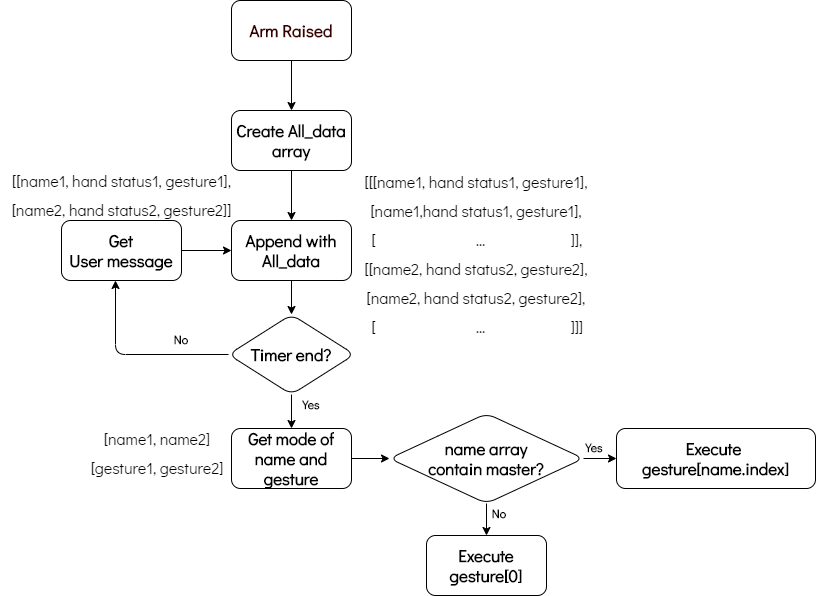
# System Diagram

## System Flow Diagram

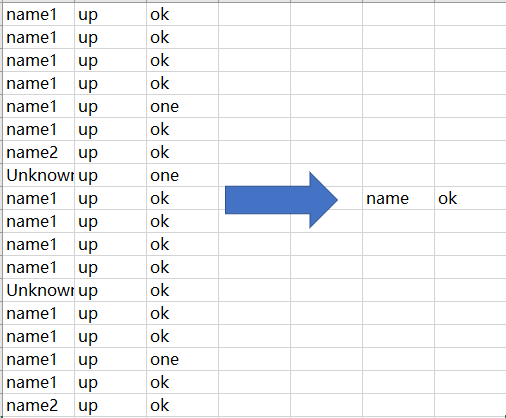
****

When the system starts, it will get the frame from camera first. After that, open pose python API will export the body and hand skeleton coordinates in the frame. Then, it will get the head frame base on the head skeleton and recognize the face in this frame. Next, the system will identify whether the user has raised his arm as a trigger for a 3-second timer. If user hasn’t raised his arm up, system will continue to update the new frame until user raise his arm up. On the other hand, if user raise his arm up, the 3-second timer will be triggered as a new thread. At the same time, SVM will also predict the result with incoming hand data which is converted into unit vectors. The result and face information will be collected. About the “All data”, it will store user information for each frame. It will be clear if user did not raise his arm. While the 3-second timer is end, system will process all data collected in 3 seconds. Finally, system will perform action by processed data.

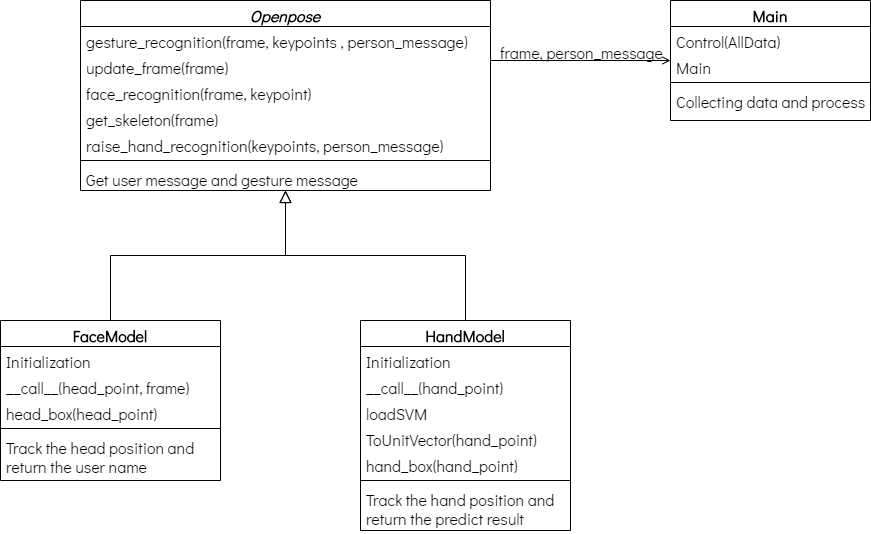
## Data Processing Diagram



The data processing flow begins when the user raises his arm. First, create an array call All data to store user information for each frame. User information for each frame will be appended to all data. It is worth mentioning that all data is a three-dimensional array. The number of the first dimension depends on the number of users in a frame; the number of the second dimension depends on the amount of data collected before the end of the timer; the third dimension is the user name, user arm status and gesture. After 3-seconds timer end, take all the data of the first user in the first dimension, and then get a 2-dimensional array. In theory, the usernames in the first column are the same. In order to avoid recognition errors in individual frames, we will take the mode of the column. For the same reason, we will also take the mode of the third column which contain gestures. Then continue to loop this action until the data calculation in all data is completed. The number of cycles is determined according to the size of the first dimension of all data.



## Class Diagram



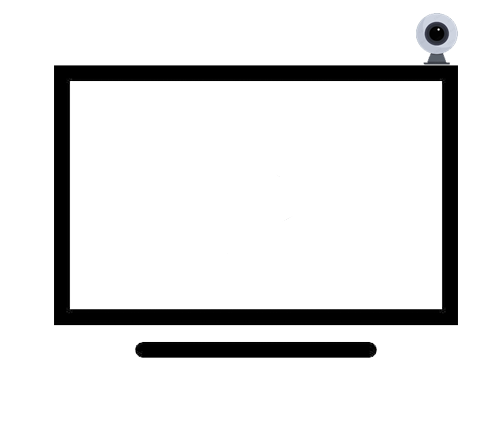
As can be seen from the diagram, this project consists of 4 Classes. They are composed of Face Model, Open pose, Hand Model and Main. In Face Model, the Initialization is to encode the face image of user. When the system calls this model, it will compare with current frame’s face. The head box function is used to find the head and extract head frame. Hand Model is to calculate unit vectors and track hand position, and use SVM to predict hand gesture results. Open pose model is responsible to get skeleton of body and hand. And also get user message and gesture result from Face Model and Hand Model. Finally, Main function will show the processed frame and process data.

# Performance

This project is run under win10 and NVidia graphics card. Under this platform, when a person is in front of the camera, the FPS can reach a maximum of 10. When 2 people are in the same frame, the FPS can reach a maximum of 7. In the case of 3 people, the FPS can reach a maximum of 4.

Mention again, the user needs to hold a gesture for 3 seconds to control the home appliance. It means that in the case of 3 people in the same frame, at least 12 sets of data can also be collected. We can know from 3.3.3 that the Recall rate of each gesture is as high as 1. Except in extreme cases, such as open pose model cannot recognize hand skeleton, 12 sets of data are enough to get correct results.

# Installation

There are several conditions to consider for the installation location. First, it needs to cover the places where users often move. Because the camera needs to take a picture of the user in front of it to be recognized normally. Second, try to install it in a place with sufficient brightness. If the camera does not have night vision, the system will also be difficult to identify. In summary, I think most people can be installed on a TV.

# Conclusion

Overall, the project achieved the expected results. Judging from the results, gesture control of home appliances is completely feasible. We can imagine the usage scenario. In a noisy environment, users use gesture control to perform better than voice control. B But this does not mean that voice control can replace machine vision control. On the contrary, machine vision cannot replace voice control. I think that the two different technologies should compete and complement each other through the market.

In this IBSP project, I used technologies that I had never been exposed to before, such as the use of SVM and some models. It also made me deeply understand that in the field of information technology, technology is changing with each passing day and we have to keep learning.

# Reference

[1] Open Pose model, by Carnegie Mellon University. Available at: <https://github.com/CMU-Perceptual-Computing-Lab/openpose>

[2] Face Recognition, MIT License. Available at: <https://github.com/ageitgey/face_recognition>

[3] Data pre-processing, Pyle, D., 1999. Data Preparation for Data Mining. Morgan Kaufmann Publishers, Los Altos, California.

[4] Confusion matrix, Lily Su, <https://medium.com/@lily_su/confusion-matrix-roc-auc-and-imbalanced-classes-in-logistic-regression-5c7ead3deefc>