VOICE CONTROLLED SURGICAL ROBOTIC ARMS

By

Run Zhang

Ximin Chen

Final Report for ECE 445, Senior Design, Spring 2020

TA: Rui Lu

29 May 2020

Abstract

Surgical doctors are always busy during the operation. Usually there would be several assistants helping control some machines. However, it needs time for assistant to response to surgeon orders. In addition, reducing the number of assistant in the operation room can reduce cost and save room in the operation room. In order to free hands of surgeons, we design a real-time voice controlled robotic arm to help them. Our robotic arm is supposed to react once the surgeon speak out the certain instruction name in the pre-defined list. Surgeons do not need to press any button after the model starts and all the actions could be controlled completely by voice. Any noise or undefined instructions would not affect the motion of robotic arm at all.

Contents

1. Introduction

Surgical Doctors need to do many sophisticated operations in their career. And they are supposed to take care of many aspects during an operation. In detail, for instance, the surgeon may need their eyes focused on the screen from a camera inside the patient body while both hands holding scalpels or tweezers during the operations. Under this circumstance, the surgeon would not be able to take any more actions to support his or her operation due to the physical limitation. Usually, there would be several assistants standing by the surgeon, who could help exchange the surgical tools and control the frames of the camera. But during the operation, it needs time for assistant to react as surgeon orders. Also, reducing one assistant in the operation room can cut the cost of operation and save more space near the patient.

In this case, we would like to develop a system which enable the surgical doctors to control the robotic arm through voice order in both Chinese and English. The system architecture is showed as figure 1, the microphone would collect the data which would be record by our program. After several steps of digital signal processing, the sample data would be compared with the order data in the database to do the speech recognition. The order index would be delivered to the microcontroller located in the robotic arm. Finally, the Robot OS in the microcontroller would translate the order into PWM pulse to drive the robotic arm move to the right position with correct angle.

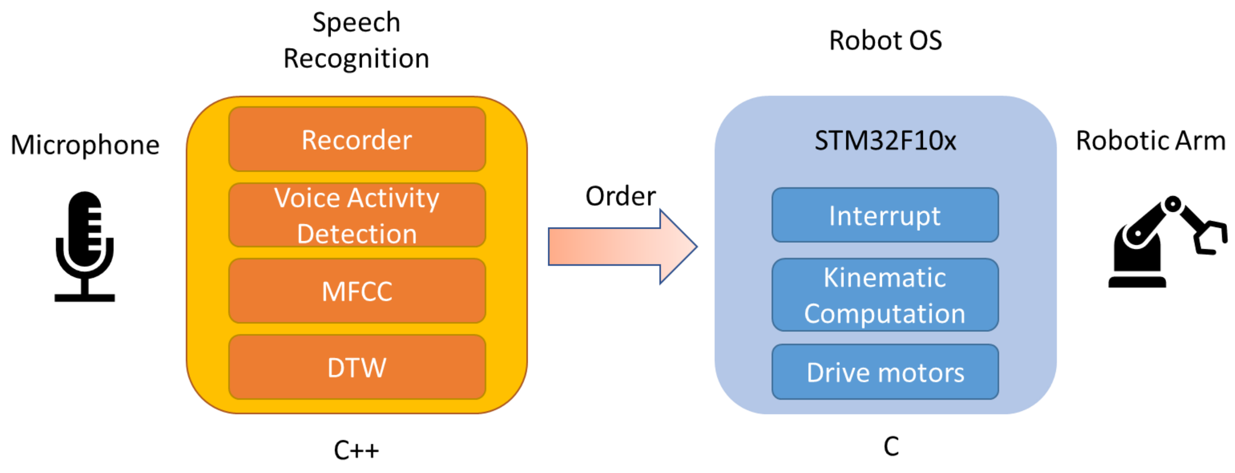


Figure 1 system architecture

2 Design

2.1 Automatic Speech Recognition

Our robotic arm is an application that combines automatic speech recognition with robotic arm control. To be specific, the task of speech recognition algorithm is aimed at receiving and recognizing instructions from the user, the format of which are single words or simple paraphrases (shown in the Table 1 below). And I will introduce how we decide the algorithms used in our program

Table 1 Instruction List

Instructions in English	Instructions in Chinese
Forward	前进
Back	后退
Left	左转
Right	右转
Upward	上升
Down	下降
Raise	抬高
Descend	低头

To achieve the functionality of our robotic arm, we need to develop an algorithm to recognize single-word (or simple-paraphrase) instructions in a high accuracy and low latency. Meanwhile, single-word detection task is not as complex as semantic segmentation for sentences. Deep Learning methods appears to be a good choice to reach high accuracy. However, building convolutional neural network layers is unnecessarily complex in our case. Besides, we need a large-scale training set to train our model. It could a huge workload for only two of us to complete hundreds or thousands of audio copies, and samples from two persons might not be enough.

Here comes another choice of Dynamic Time Warping [1]. The basic idea of DTW is to calculate the similarity between train audio and test audio, detailed information of which would be introduced in the following parts. The advantage of DTW is simple and high accuracy in single word detection. We can avoid the training process of deep learning, which means we could save time in collecting large-scale training dataset and the model training time. And in our previous survey, DTW is incompetent to recognition input audio of instructions quickly and accurately.

But DTW is not available to deal with the original natural audio, which is mixed with environment voice. MFCC [2] is introduced here for audio preprocessing before DTW, which helps reduce the noise. In addition, we use voice activity detection to cut the voice piece out from the real-time record audio.

2.1.1 Recorder through Windows API

In our speech recognition algorithm, we need to use WAV format to read audio input from microphone and store instruction voice data in our computer. WAV is a Resource Interchange File Format (RIFF) bitstream format developed by IBM and Microsoft on PCs. It is mainly used on windows system to help store uncompressed audio data. Fig.2 [3] below is the data structure in WAV file.

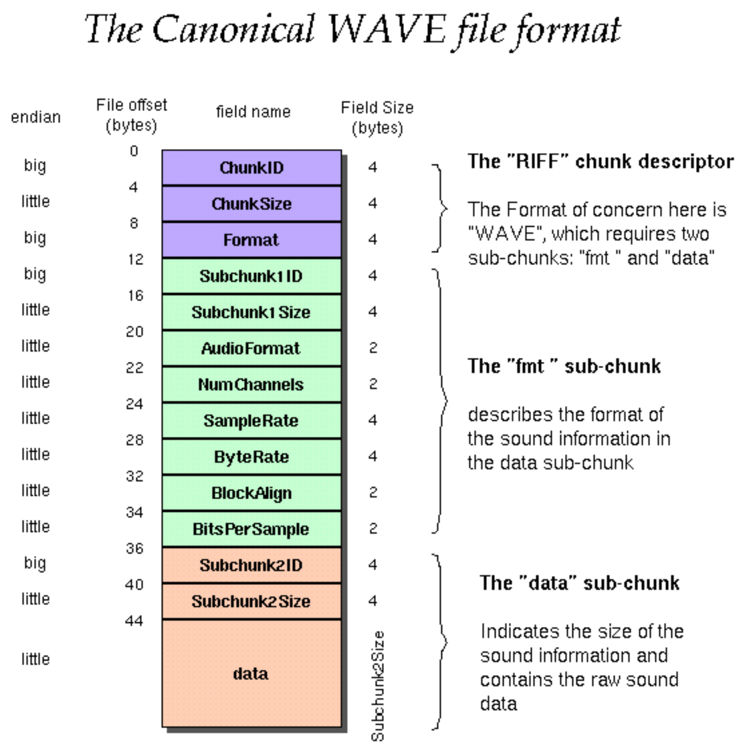


Figure 2 WAV Header Format

According to the figure above, the main concern is to define the audio format in "fmt" sub-chunk and voice data in "data" sub-chunk. The "fmt" sub-chunk records the properties of the audio, including audio format, number of channels, sample rate, byte rate, number of block align and how many bits per sample. And the "data" sub-chunk stores all the audio input in uncompressed pulse code modulation (PCM) format.

Following code shows how to define the header of WAV format and store audio in into the data chunk.

```
#define WAVE_HEAD_LENGTH 44 // bytes length of header
typedef struct
{
    char            chRIFF[4];
    DWORD           dwRIFFLen;
    char            chWAVE[4];
    char            chFMT[4];
    DWORD           dwFMTLen;
    PCMWAVEFORMAT   pwf;
    char            chDATA[4];
    DWORD               dwDATALen;
}WaveHeader;
```

2.1.2 Frame Separation

Voice input could be stored in separate digits, but it does not mean anything unless we listen to a serial of voice signal. Due to its time-variant characteristic, we need to regard audio as a dynamic information combined with time, which means frames of audio should be the minimum unit of speech processing. In order to keep the audio frame stable, usually such the length of certain frame is around 20-40 milliseconds. In addition, we need to set frame shift length less than the frame length. Otherwise it is possible that some important information is separated into two frames, thus making our speech recognition decrease.

To be more specific, for 8kHz recording, we set $FrameLength = 240$ samples, i.e. 30 milliseconds per frame. Then the pointer would move by $\mathrm{FrameShift} = 80$ to obtain the next frame. Finally, the length of output array after framing would be $\frac{\mathrm{InputLength} - \mathrm{FrameLength} + \mathrm{FrameShift}}{\mathrm{FrameShift}}$, of which each element stores $FrameLength = 240$ samples.

2.1.3 Voice Activity Detection (VAD) [4]

Ideally, we just need the exact voice pieces of the instruction. However, we cannot identify the start and end point of our instructions since we are building a real-time speech recognition system. Therefore, we need to develop another algorithm recognition a valid piece of audio and return it to the speech recognition program. Basically, we use a double-check system to identify a valid voice piece, which checks the zero-crossing rate (ZCR) and the amplitude of input audio. Here is an example when I say “great” below printed by MATLAB in Fig.3.

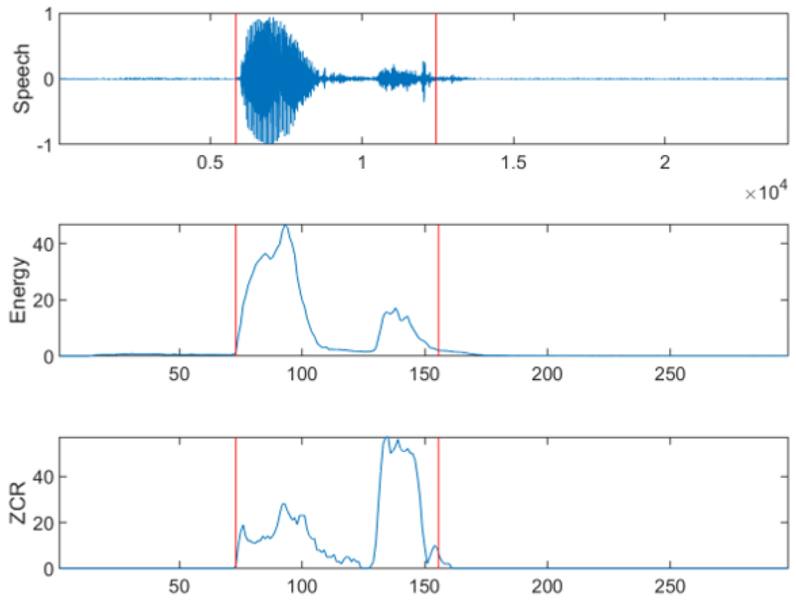


Figure 3 Speech Spectrum with ZCR and AMP

1. Zero-crossing rate and amplitude for check valid voice
   a) ZCR describes the rate of sign changes along a signal, which is an important spectral

characteristic of a signal. The formula of ZCR for a certain frame is shown as below.

$$zcr = \frac{1}{FrameLength} \sum_{i=1}^{FrameLength-1} |sign(x[i]) - sign(x[i-1])| \qquad (1)$$

$$sign(a) = \begin{cases} 0, & a < 0 \\ 1, & a > 0 \end{cases} \qquad (2)$$

During recording, zero-crossing would take place all the time due to the environmental noise, but ZCR for speech is apparently higher than silence time. From the example of "great", we can find that ZCR during speaking time is much higher than other time. Though the data pointer in WAV format prefers `BYTE` type, we use `int16*` in the CPP level (one `int16*` address contains two bytes) which stores signed digits.

b) Another threshold is the amplitude, which reflects the short-time energy for frames during speaking. The calculation for amplitude is more complex than that of ZCR. Before framing process, we need to pre-emphasize the speech audio, which helps compensate the power loss in high frequency area. We can apply a filter on the original audio with coefficients of 1 and 0.9375.

$$x[i] = x[i] - 0.9375x[i-1] \qquad (3)$$

0.9375 is an empirical value to achieve high performance, usually between 0.9-1, which would also be used in the following parts. Then after framing, we need to add window function on the frames before DFT, since DFT prefers periodic signals and, for example, Hanming window could make the short frame more periodic. Finally, by DFT we could estimate the power spectrum.

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N}, 1 \le k \le N \qquad (4)$$

, where $h(n)$ is a window function to enhance the periodicity.

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \qquad (5)$$

2. Minimum word length and maximum silence time
An ideal voice piece consists of many consecutive frames. However, in the example of "great" above, clearly there is an almost silent part within the pronunciation of "great". How does the algorithm know that it needs to continue instead of halting at once? Here we set a value named maximum silence time in out CPP. Once the voice falls in silence, we would start counting the local silence time and the algorithm would accept a short time silence no larger than that threshold. By picking an appropriate value in experiments, we could guarantee that the algorithm would listen to a whole word or even a paraphrase. After detection, the algorithm would remove the tail of that audio piece according to the maximum silence time. On the contrary, if we detect a piece of voice smaller than pre-set minimum word length, the algorithm will regard it as noise and then remove it.

2.1.4 Mel Frequency Cepstrum Coefficient (MFCC)

After framing and VAD, next step before DTW is Mel Frequency Cepstrum Coefficient processing, which is aimed at fetch features from voice data. Particularly, we need to identify important components for language content and meanwhile discard some irrelevant stuff including background noise and emotion of the speakers.

The most popular method would be MFCC (Mel Frequency Cepstral Coefficient) introduced by Davis and Mermelstein in the 1980s. MFCC focuses more on the separate audio fragments according to the given time interval (usually 20-40 milliseconds). And in order to simulate human's cochlea, which cannot identify sounds with close frequency and such effect would be more pronounced as the frequency increases, the Mel filter bank helps ignore some variations in voice frequency as the frequency increases.

Remember MFCC is a separate step after VAD with different functionality, though there are some overlaps on the audio processing part. The whole flow of MFCC is shown as below [5]:

1. Framing spectrum into frames with length of 20-40 milliseconds (same as framing part)
2. Use DFT to estimate power spectrum (same as amp calculation)

$$\mathrm{S_i(k)} = \sum_{\mathrm{n=1}}^{\mathrm{N}} \mathrm{s_i(n)h(n)e^{-j2\pi kn/N}}, 1 \leq k \leq N \tag{4}$$

, where $\mathrm{h(n)}$ is a window function to enhance the periodicity.

$$\mathrm{P_i(k)} = \frac{1}{\mathrm{N}} |\mathrm{S_i(k)}|^2 \tag{5}$$

3. Use Mel filters to divide power spectrum into different bins. In the examples below, (a) in Fig.4 is a complete Mel filterbank with 26 filters which are distributed denser in low frequency so that the model is more sensitive about low frequency data, and less care about the high frequency range.

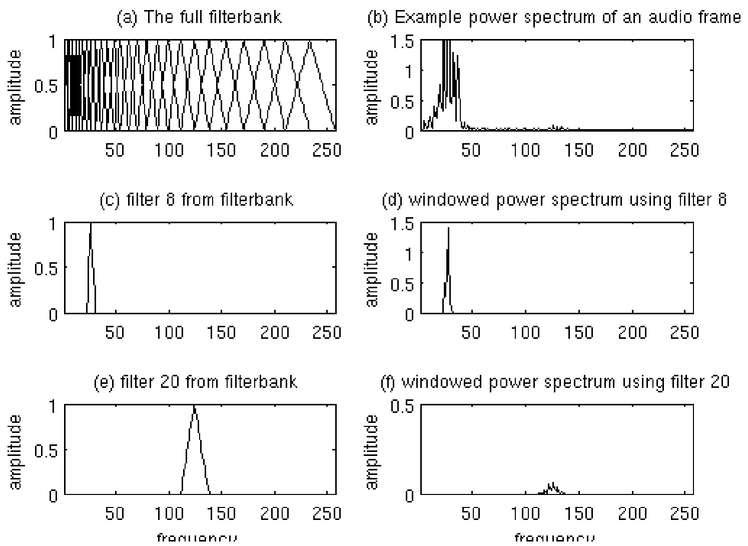


Figure 4 Mel Filter Banks

4. Fetch logarithm on the power spectrum as the recognition of speech are not linearly depend on the power spectrum
5. Use DCT to obtain cepstral coefficients with 26-dimension
6. Additional step: by analysis of first-order or second-order difference, we could increase dynamic change information in our result with an output in 39-dimension.
   The first-order function is shown as below:

$$d_t = \frac{\sum_{n=1}^{2} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{2} n^2} \tag{6}$$

   (second-order uses it twice)

Compared with other methods including LPCC which provide prediction on periodical signal, MFCC is proved to have higher performance for Automatic Speech Recognition.

2.1.5 Dynamic Time Warping (DTW) [1]

According to our survey and literature review, there are several algorithms to recognize the speech content. The most popular algorithms are based on DTW (Dynamic Time Warping) and HMM based on parametric model. Since the DTW algorithm performs good with small vocabulary and easily implemented, whereas the HMM algorithm is more suitable with large vocabulary and needs more time to react. As we all know, people have different habits on the pronunciation of the same word, for example, the tone and length of one syllable might be different. DTW is trying to erase the personal difference between the pronunciation test sample and word model in database. After that, two samples are basically in the same scale and rhythm. In this case, we can directly compare the test sample with the vocabulary models we have.

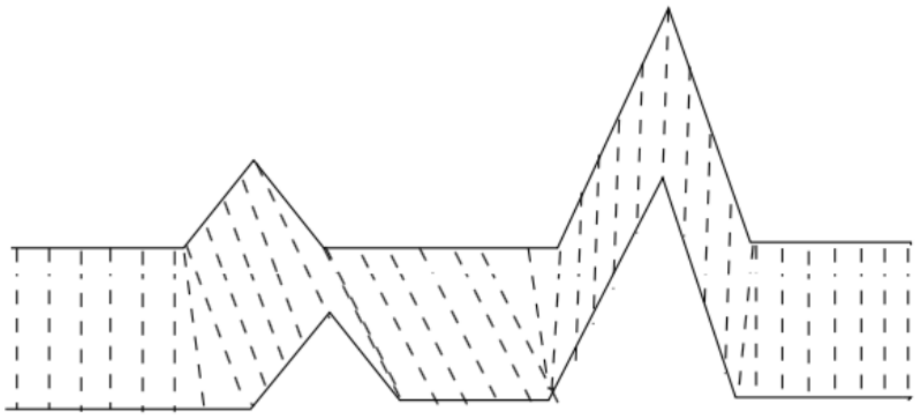


Figure 5 Distance between two similar signal

Assume that we have two arrays (P, Q) of speech data need to compare as Fig.5 [6] shown. One of them is the sample data we get from microphone, and the other is the speech data in our database. In order to find how similar of the two data, the basic idea of DTW algorithm is to find the shortest path from the beginning (time = 0) to the end among to data flow.

We would use the idea of dynamic programming to derive the shortest path like Fig.6 [7].

$$d(P_i, Q_j) = \min\left(d(P_{i-1}, Q_j) + d(i,j), d(P_i, Q_{j-1}) + d(i,j), d(P_{i-1}, Q_{j-1}) + 2d(i,j)\right) \quad (7)$$

After we compare the speech order with all samples, if the lowest distance of those shortest paths is smaller than a threshold value we set. We can determine the shortest one is the matched speech order.

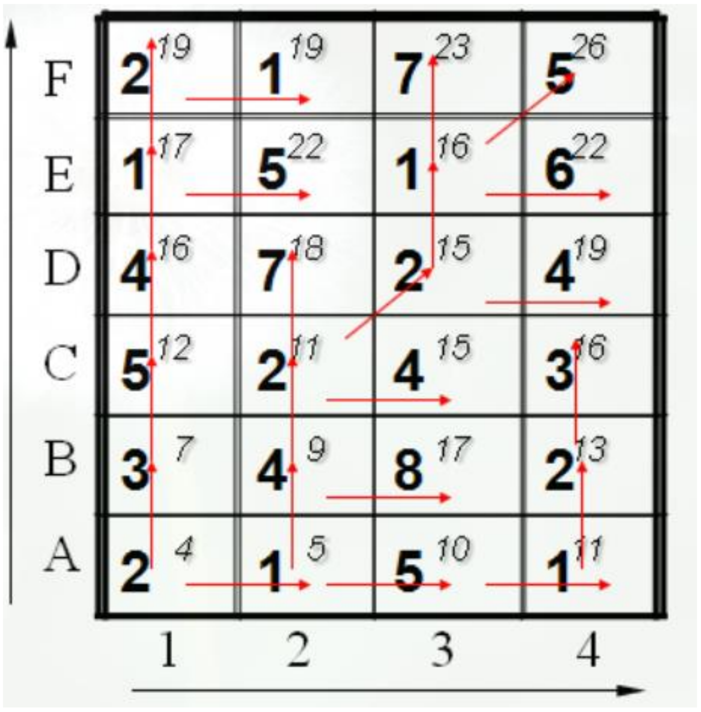


Figure 6 Dynamic Processing

According to the feature we want to implement, the agent only needs to recognize several speech orders such as direction order like 'left' and 'right' and angular orders.

Therefore, we would primarily choose the DTW algorithm to implement the required features. The algorithm would do dynamic time warping on the MFCC (Mel-scale Frequency Cepstral Coefficients) feature of the speech data and check if it matches the speech order we set in the database.

2.2 Robot Arm

The robot arm we choose is designed by a Chinese company LOBOT located in Shenzhen. It is a 6DOF programmable robotic arm. It contains motors, metal skeleton and electronic interface as It showed in figure.7[]. Due to the requirements, we choose the microcontroller STM32 to do part of computation and control the robot arm.

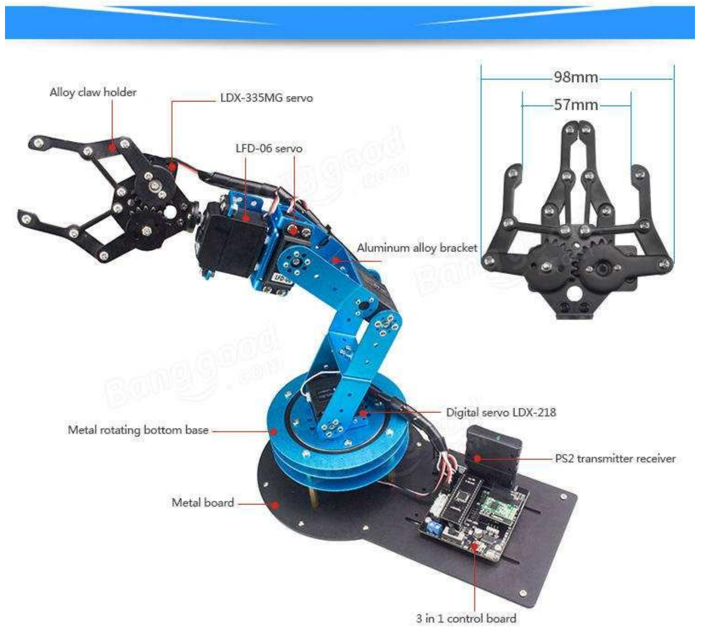
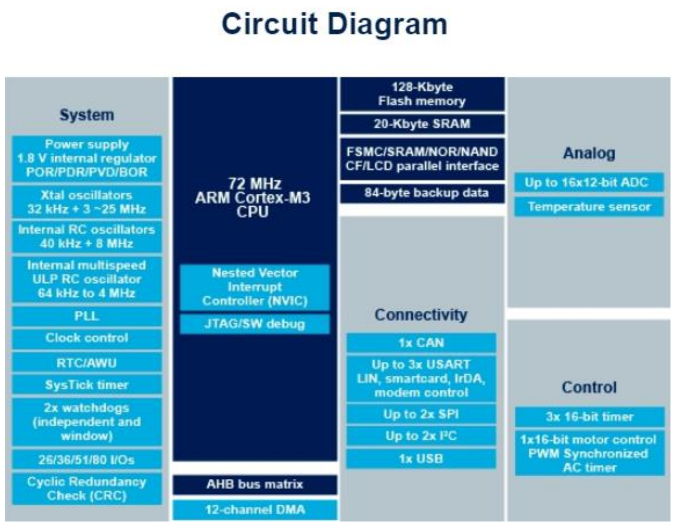


Figure 7 Robotic Arm Structure

2.2.1 STM32

STM32 is a family of 32-bit microcontroller integrated circuits by STMicroelectronics. The version we use is STM32F103RBT6 which contains the ARM Cortex-M3 core. It is set on an extended board that contains PWM motor driver port, power source, USB port and some other modules which we do not need in this



Circuit Diagram
System
Power supply 1.8 V internal regulator POR/PDR/PVD/BOR
Xtal oscillators 32 kHz + 3 ~25 MHz
Internal RC oscillators 40 kHz + 8 MHz
Internal multispeed ULP RC oscillator 64 kHz to 4 MHz
PLL
Clock control
RTC/AWU
SysTick timer
2x watchdogs (independent and window)
26/36/51/80 I/Os
Cyclic Redundancy Check (CRC)
72 MHz ARM Cortex-M3 CPU
Nested Vector Interrupt Controller (NVIC)
JTAG/SW debug
AHB bus matrix
12-channel DMA
128-Kbyte Flash memory
20-Kbyte SRAM
FSMC/SRAM/NOR/NAND CF/LCD parallel interface
84-byte backup data
Connectivity
1x CAN
Up to 3x USART LIN, smartcard, IrDA, modem control
Up to 2x SPI
Up to 2x I²C
1x USB
Analog
Up to 16x12-bit ADC
Temperature sensor
Control
3x 16-bit timer
1x16-bit motor control PWM Synchronized AC timer

Figure 8 STM32 Architecture

project. [8]

2.2.2 Motor and Structure

There are 6 motors installed in the robotic arm. [9] The first and second motor control the angle and open-close mechanism of the clamp on the end effector. The third, fourth and fifth motor decide the position of the clamp in the vertical plain. In the end, the last motor control the direction it faces toward

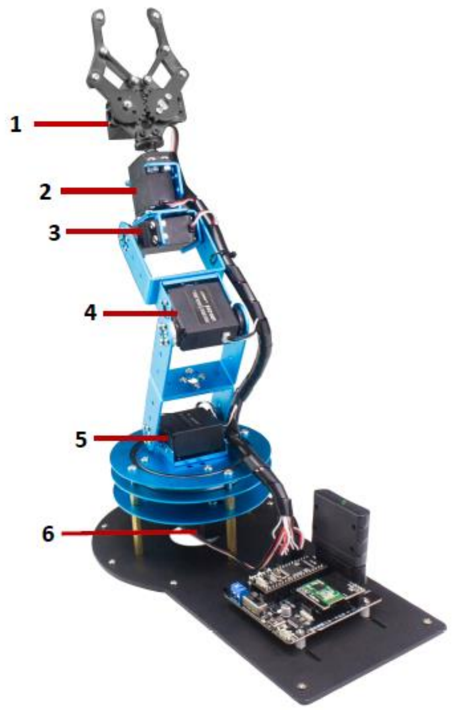


Figure 9 Motor Sets in Robotic Arm

horizontal position.

Each motor is controlled by the PWM voltage signal that generated by the STM32. As it showed in the figure.10 [9] the angle domain of motors is from negative ninety degree to positive ninety degree. The total time period for PWM pulse is 2.5ms, we could adjust the time that control signal stays high to

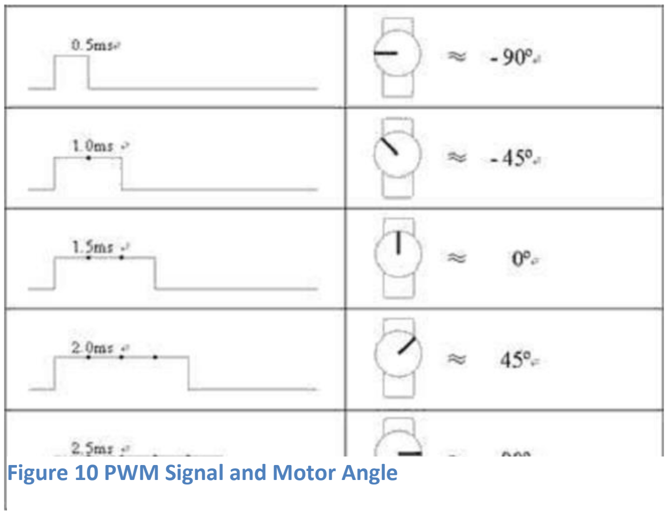


Figure 10 PWM Signal and Motor Angle

change the angle for each motor. In the project, we only use motors 3 to 6 and assign the joint between 2 and 3 as the end effector which represent the position it locates.

2.3 communication

Between speech recognition program in personal computer and control system in microcontroller STM32, we need to build serial communication to load file and transmit control signal.

2.3.1 UART serial port

A universal asynchronous receiver-transmitter (UART) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. According to the figure.11 [10] Under the system, only three signals, Tx (transmitted serial data), Rx (received serial data), and ground, are needed. However, the protocol can provide a rather robust moderate-speed, full-duplex communication since it does not need clock signal transmitted from one device to the other. The only requirment is both device need have accruate internal clock and set the same baud rate. Also, UART transmitted data is organized into packets. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART), an optional parity bit, and 1 or 2 stop bits. In this situation, they can read the data in the buffer by recognizing the start bit and end bit or write data back into the buffer with

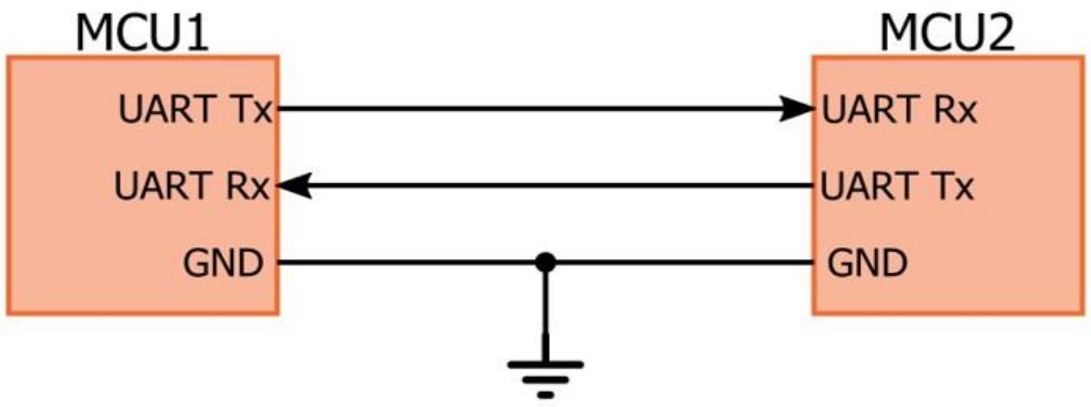


Figure 11 Transimission between two devices under UART

particulalr format.

2.3.2 Serial Communication in PC

There are two parts in this section. The first part is how to load the control signal we design into the STM32. Basically, Keil µVision IDE to build the target file and burn the file into the STM32 through serial port toolbox. The second part is the transmission between the running program and STM32.

In order to send and receive data, we need interact with the hardware interface, communication port (COM). Following the agreement, invoking WinAPI to create information file and relative memory with port's name "COM4" can enable the use of corresponding port. Then, function WriteFile() and ReadFile() could help us to transmit data through it.

```
        hCom = CreateFileA(portname, //portname
                           GENERIC_READ | GENERIC_WRITE, //read write enable
                           0, NULL, //not shared, default
```

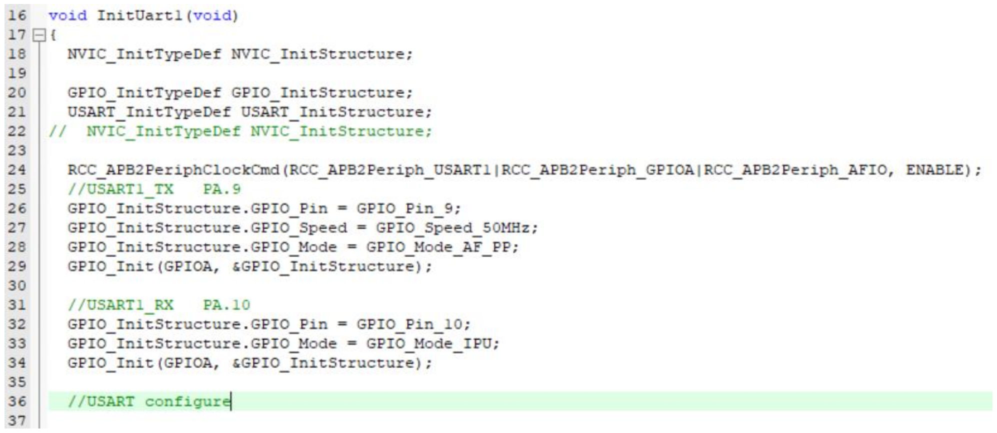
```
                    OPEN_EXISTING, //working with
                    FILE_FLAG_OVERLAPPED, //asynchronous
                    NULL) ;//default
```

2.4 Robot OS

After the generation of order from speech recognition module, the signal would be delivered to the STM32 through the USB cable. The STM32 need to handle the signal and generate corresponding PWM signal to drive the motor into correct positions. Therefore, a simple operating system is required for the task. Before we modify the system on the board, it contains a framework and several libraries in C language for STM32f10x series. What we need to do is to add PC UART handler and motion computation, combined with the main loop.

2.4.1 Interrupt

Interrupt is asynchronous interruption generated by devices which is a high-priority task for the system. When there is no outside interrupt like UART signal and pressing button, the system would run the idle loop which is maintaining the robot arm in the current position. In the main loop, it would call a UART function which deals with the UART signal. When there is an interrupt happened, the system would stop the idle routine loop and start dealing with the interrupt.



```
void InitUart1(void)
{
  NVIC_InitTypeDef NVIC_InitStructure;

  GPIO_InitTypeDef GPIO_InitStructure;
  USART_InitTypeDef USART_InitStructure;
//  NVIC_InitTypeDef NVIC_InitStructure;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1|RCC_APB2Periph_GPIOA|RCC_APB2Periph_AFIO, ENABLE);
  //USART1_TX   PA.9
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  //USART1_RX   PA.10
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  //USART configure

```

Figure 12 Initialization of UART Port

Before handling the interrupt, we need to initialize the UART port in STM32 first. There is a piece code segment in figure.12. As it defined by the library, when the STM32 receive signal from the port, it would generate a interrupt and set a flag function USART_GetITStatus(USART1, USART_IT_RXNE) to true, which would allow UART function dealing with the data in buffer. In our project, we assign “xx” as the start flag and “yy” as the end flag. An index between “1” between “8” representing 8 different orders. After checking the index is valid, the index would be delivered to the position computation model.

2.4.2 Motion Logic

As we discussed in the robot arm part, our robot OS needs to control 4 motors. First, the OS would save the current status (position, angles) of the robot arm in memory. The basic routine is that every time it receives a movement order, it would add a step size in order’s direction and calculate the angles of each motor should assigned. Then, the computation unit call the drivers to drive the motors into the positions.

In addition, for the practical use for surgery camera in the future use, we make the clamp (camera) always look horizontally within limit domain (each motor's working range is -90 to 90 degree). Also, the system would stop the robot at the safety boundary when clamp tries to reach somewhere under the 0 plane in z-axis. However, due to the limit of STM32, it computes trigonometric function slowly, we use look-up table to replace sin() and cos() function with step size of 5 degree. This method significantly increases the speed of computation but loses a little bit accuracy.

3. Design Verification

3.1 Automatic Speech Recognition

3.1.1 Real-Time Speech Recognition Test

Overall, our speech recognition part works in real-time as we expected, which means our algorithm could detect speaking words and send correct instructions to the robotic arm continuously without human control. Below is our test data.

Table 2 Test Result

Order	Test times	Correct times	Accuracy
Down	20	20	100%
Upward	20	19	95%
Forward	20	20	100%
Back	20	19	95%
Left	20	20	100%
Right	20	19	95%
Raise	20	17	85%
Descend	20	17	85%

Our accuracy rate performs well on correct instruction input identification. However, it has problem in avoiding undefined input. Each time the algorithm receives a possible instruction audio array, it would compare it with all other model train audios by DTW. We set 30000 as the maximum input threshold, below which the input audio could be one of the instructions, or undefined voice instead. In the test process, sometimes an invalid input would be recognized as an valid instruction by mistake. Table3 below shows some examples.

Table 3 Wrong Word Identification

Giving Order	Test result
Hello	Down

Teddy	Forward
Vito	Descend

It is hard to calculate the real ratio that how many invalid words among all the words could be wrongly recognized as valid instructions, since the range of words is so large.

3.1.2 Latency Between Software and Hardware Level

The process of DTW could be completed within tens of milliseconds and no apparent delay when we only test software level. However, when we connect robotic arm on the PC, the latency between input audio and output predicted instructions increases. The main difference is that we add a port communication API function for STM32 board. In return it adds some workload to the thread of calculating DTW values.

3.2 Robotic Arm Control

3.2.1 Camera Stability and Motion Limit

In our original design, the end effector needs to be stable so that the camera could provide valid image. Also, the robotic arm should be able to move in a larger space. However, due to the limit of DOF and motor angle limit, there are still some improvement can be done in this area.

3.2.2 Trajectory Planning

In robotics design, trajectory planning is important since it can decide the specific trajectory the end effector would pass in movement. In this case, we can apply a more exquisite control on it. However, we have not added it into our control system for now. It can reach the right position but sometimes it would go through a large area which may cause inconvenience in practical use.

4. Costs

4.1 Parts

Table 4 Parts Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
6DOF Robotic Arm Kit	Hiwonder Technology	121.3	121.3	121.3
SHM1000 Microphone	Philips	12.7	12.7	12.7
Total				**134.0**

4.2 Labor

Our project continued for over 12 weeks. Due to the limitation of working at home during COVID-19 period, we have little progress before May. After we returned campus, we have roughly spend 70 hours per teammates on the project, which cost $25\$ \times 2persons \times 70hrs = 3500\$$.

5. Conclusion

5.1 Accomplishments

Overall we obtain a system with low latency that identify the correct instructions in high accuracy. We have achieved the basic model level of our initial design.

5.2 Uncertainties

A huge uncertainty in our current model is the detection of invalid instructions. It is normal to receive unexpected audio from both human speaking and environmental noise. Sometimes such invalid input would be wrongly identified as correct instructions, which increase the risk possibility of our robotic arm motion

5.3 Ethical considerations

Audio data could be very private for users. In that aspect, we need to keep the stored audio data private and unavailable to other users. The only usage of audio data should be limited within the speech recognition algorithm.

5.4 Future work

5.4.1 Improvement in Speech Recognition

As mentioned above, we need to reduce the possibility that mistake an invalid instruction with a correct one without sacrificing current accuracy on correct input and low latency. Another direction is to expand our dataset. Currently with limited training dataset by our teammates, the algorithm could have a difficult time recognizing speaking from strangers. The pronunciation for a certain word varies among different persons due to their habits and accents. And our current feature extraction algorithm is not strong enough to ignore such differences among persons.

5.4.2 Improvement in Robotic Operating System

Like what we discussed in section 3.2, due to the limit of DOF and motor angle limit, there are still some improvement can be done in the area of precising control. In the future, we could apply more complex system on more advanced robotic arm to get better performance.

References

[1] Berndt, D. J., & Clifford, J. (1994, July). Using dynamic time warping to find patterns in time series. In KDD workshop (Vol. 10, No. 16, pp. 359-370). Available at: https://www.aaai.org/Papers/Workshops/1994/WS-94-03/WS94-03-031.pdf.

[2] Ganchev, T., Fakotakis, N., & Kokkinakis, G. (2005, October). Comparative evaluation of various MFCC implementations on the speaker verification task. In Proceedings of the SPECOM (Vol. 1, No. 2005, pp. 191-194). Available at: https://www.researchgate.net/profile/Todor_Ganchev/publication/228756314_Comparative_evaluation_of_various_MFCC_implementations_on_the_speaker_verification_task/links/0912f50d0a3ac85206000000.pdf.

[3] WAVE PCM soundfile format, web page. Available at: http://soundfile.sapp.org/doc/WaveFormat/.

[4] Sohn, J., Kim, N. S., & Sung, W. (1999). A statistical model-based voice activity detection. IEEE signal processing letters, 6(1), 1-3. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=736233.

[5] Mel Frequency Cepstral Coefficient (MFCC) tutorial, web page. Available at: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[6] Dynamic Time Warping, web page. https://en.wikipedia.org/wiki/Dynamic_time_warping.

[7] Dynamic Time Warping Theory Analysis, web page. Available at: https://blog.csdn.net/huoxingdeshidai6/article/details/89645189.

[8] STM32F103RC, web page. https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32f1-series/stm32f103/stm32f103rb.html

[9] LeArm Hiwonder, web page. https://www.lobot-robot.com/c_detail/18.html

[10] back to basics the universal asynchronous receiver transmitter, web page. https://www.allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart/.

Appendix A Requirement and Verification Table

Table6 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Voice Control a. React rapidly b. High accuracy	1. Speech Recognition a. Work in real time b. High accuracy	a. Y b. Y
2. Camera Movement a. Camera Stable b. Move in correct position c. Move in the shortest path	2. Robot Arm Control a. Motor 3 adjusts the angle towards b. Inverse kinematic c. Trajectory Planning	a. Y b. Y c. N