**Multimedia software based on Python and PyQt5**

**Developer： 马东徽**

**Contact：** [**18700580583/2507441744@qq.com**](mailto:18700580583/2507441744@qq.com)

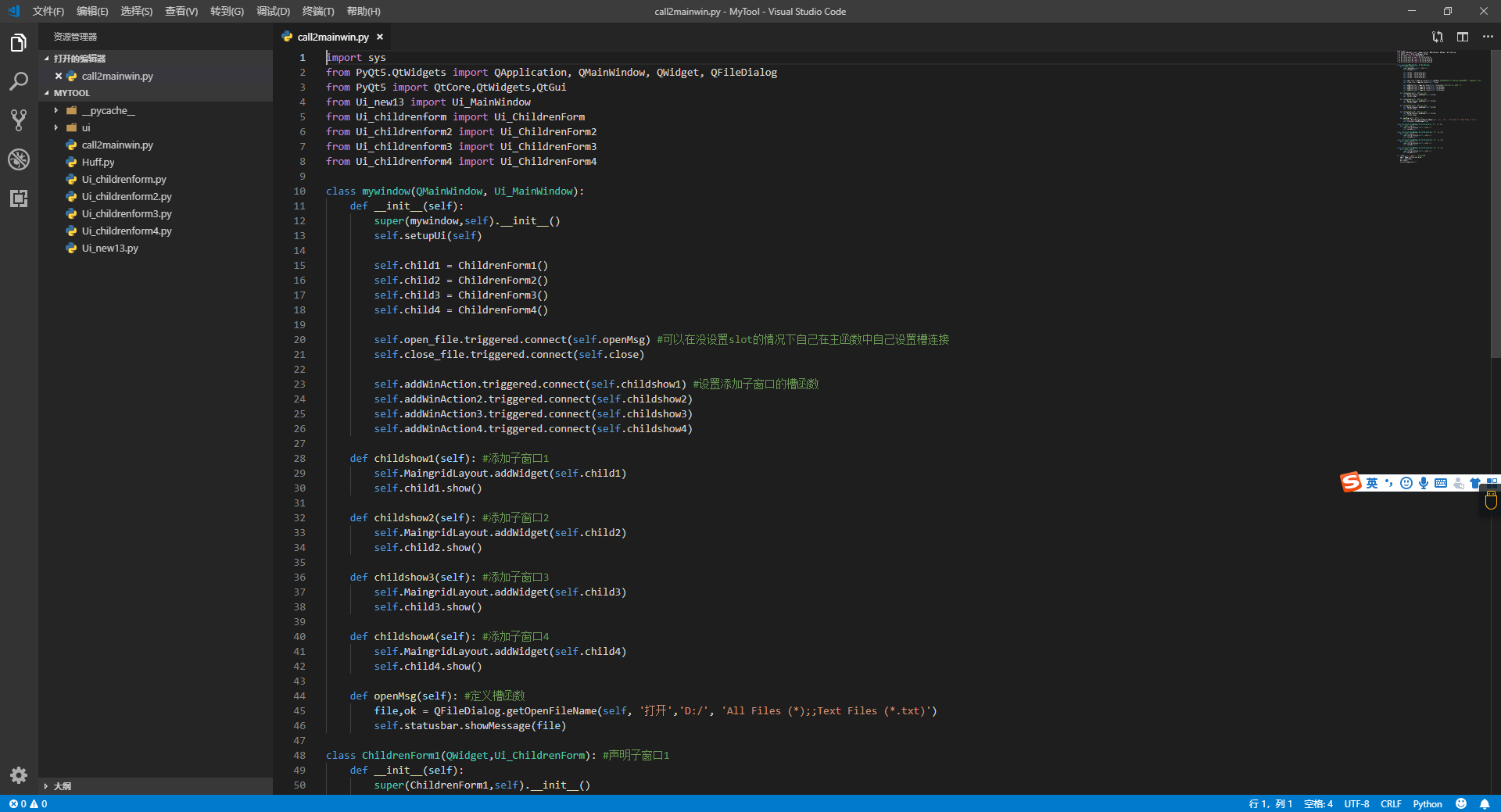
**ABSTRACT**

This project using python and is based on PyQt5 platform to complete a multi-functional toolbox. With the function about image processing, frame fetching by video, audio processing and text compression.

The image processing part includes canny edge detection, histogram equalization and threshold segmentation. The frame fetching part of video can play the file selected by the user, and automatically extract the frame of video according to a certain interval, and save it in the folder. You can also view the extracted pictures in the software. The audio processing part can play audio files in wav format, flip audio files, convert wav files into MP3 format and store, realize audio clipping and splicing, adjust local volume, etc. The data compression part can compress user's text files in three different ways, including repetition suppression, pattern replacement and Hoffman coding.

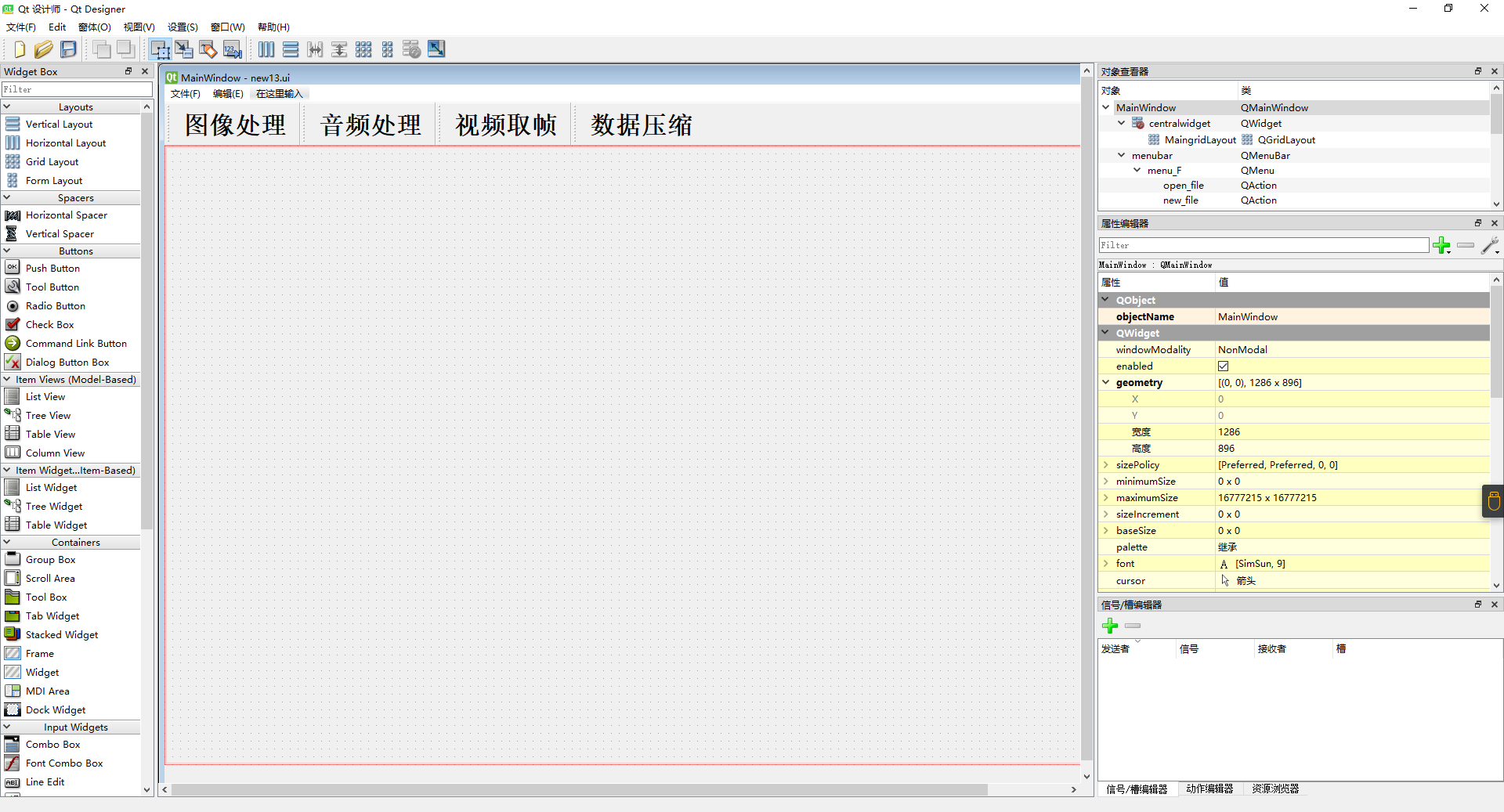
1. **Development preparation and tool selection**
2. Code Writing

Visual Studio Code (VScode) is a humanized, user-friendly IDE. Can automatically complete, report errors, alignment. And it has rich extensions. For example, Chinese package extension, icon extension and so on. It greatly improves the efficiency of code writing.



1. UI design

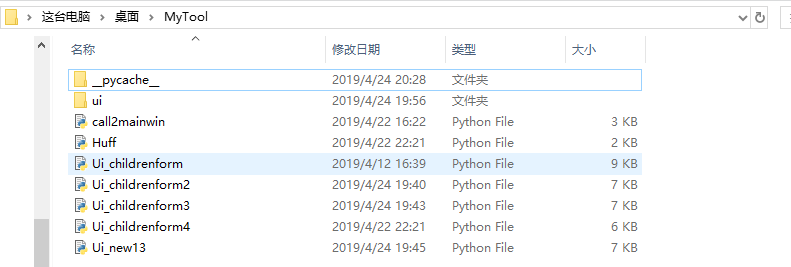
QT is an excellent tool for GUI development. PyQt5 is a cross-platform toolkit for creating GUI applications. It integrates Python and Qt libraries. In other words, PyQt5 allows APIs in the Qt library to be invoked using the Python language

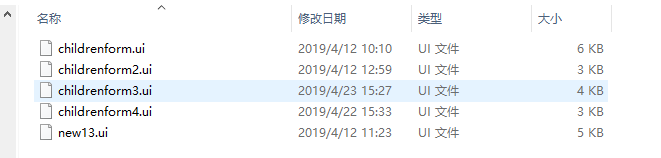


1. Knowledge preparation

Because I have a certain python foundation, before I was learning machine learning related content, I would like to take this opportunity to continue to consolidate Python level, but there is no relevant development experience of PyQt5, so I learned about PyQt5 development from online and related books. General process

1. **Project file composition**

The whole software consists of five parts. The four sub-windows implement the four functions of the software respectively, and the main window is responsible for accommodating four sub-windows. The project includes 7 program files and 5 ui files. 



Ui file (like **childrenform.ui** and so on) is the initial design by QtDesigner what is QT's GUI interface design tool. After compiling to Python file with vscode, modify the file inside the file according to the project needs, and get the program file of the child window. (like the **Ui\_childrenform.py**)

1. **Function realization**

First of all, run the python file **call2mainwin.py**

1. **Image Process**

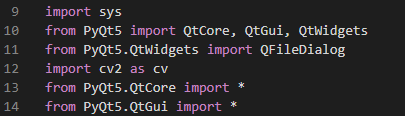
Image processing mainly includes converting RGB images into grayscale images and performing Canny edge detection, histogram equalization, threshold segmentation, etc.

The Python file about image processing is **Ui\_childrenform.py**



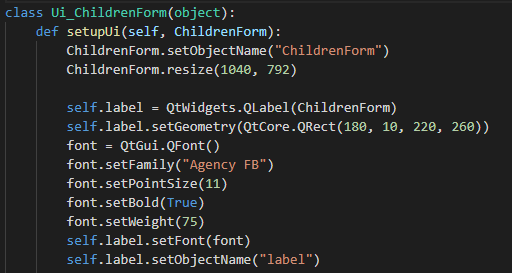
**Figure3.1 Image processing window**

First of all, we need to import some of the necessary components of PyQt5(like QtCore, QtGui, QWidget). Here we import the computer vision library **OpenCV2** for image processing



**Figure3.2 Module and library import**

Followed by a class related to a series of widgets(**控件**) in the GUI subwindow



**Figure3.3 Widget size, coordinate font, etc**.

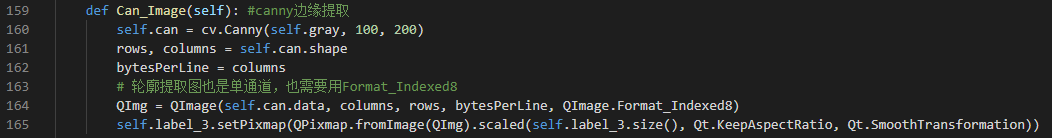
In this, we also define the slot functions associated with those controls.

For example:

We define a slot function(**槽函数**) **Can\_Image** for the button which we call It cannybutton, when we clicked the cannybutton, the signal will be send, then the function will be execute to realize Canny edge detection

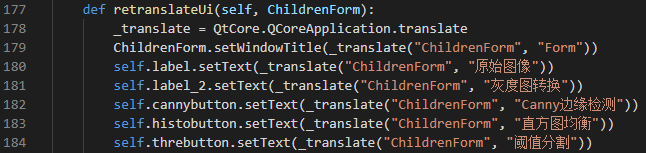
So we must define slot function later:

Here, we use Qlable to display the processed image. Qt.keepAspectRatio and Qt.SmoothTransformation is to adapt the processed image to the size of the label and to make the image not distorted after scaling



**Figure 3.4 Slot function of cannybutton—Can\_Image**

Last part is about the tag of the button, also the name of the lables or buttons



**Figure3.5 Tag of labels or buttons**

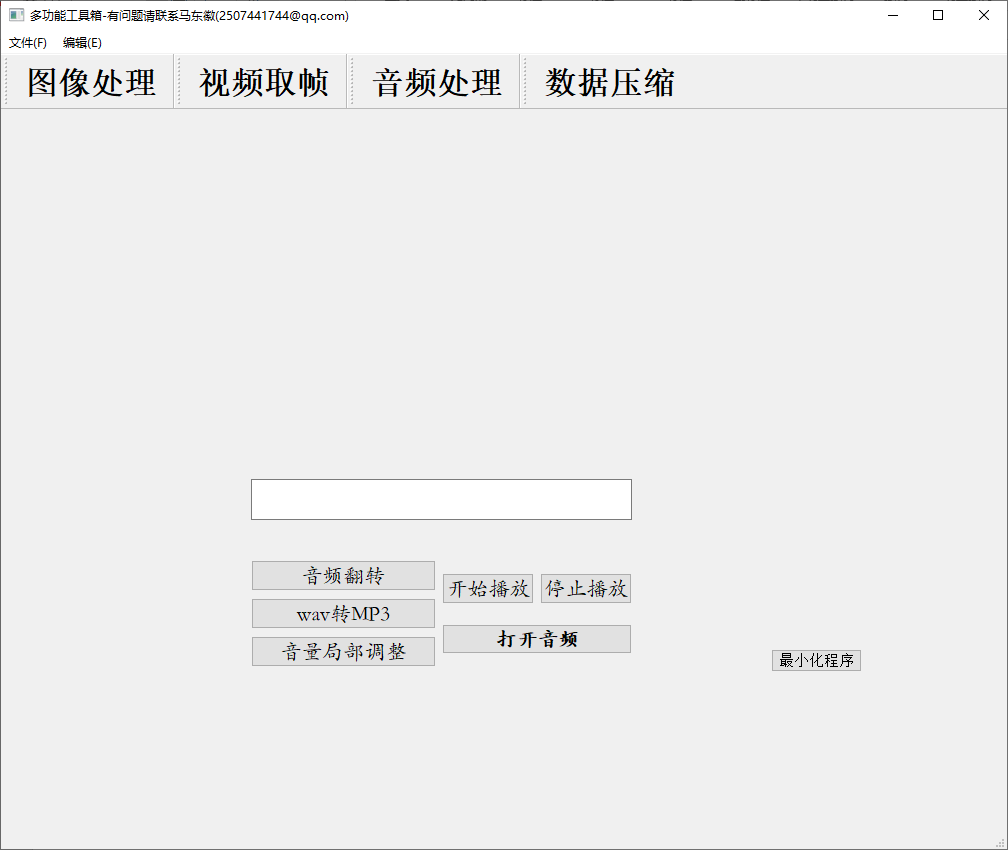
And the final process result is shown as:



**Figure3.6 Final result of image processing**

1. **Audio Process**

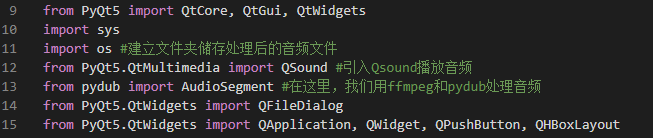
Audio process mainly about play the wav file that user choose, format conversion from wav to MP3 and store it, cut and splice the audio and adjust the volume locally. All the processed files will be stored in a folder. (The path is **D:\audioprocess\**)



**Figure 3.7 Audio process child window**

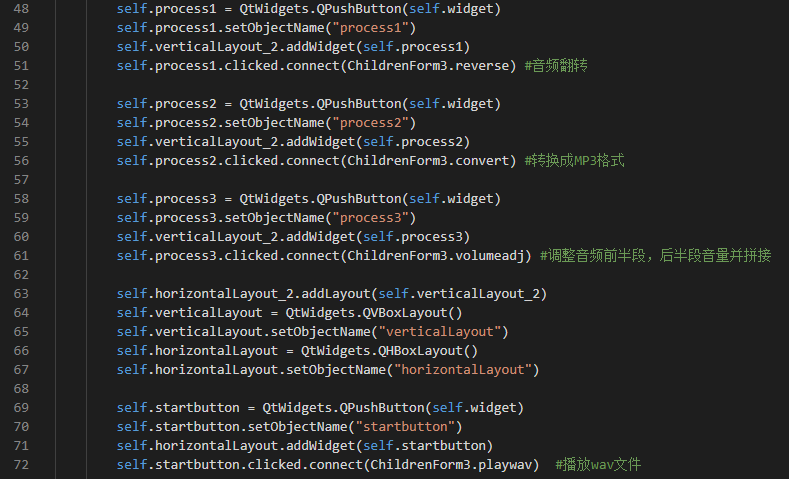
The Python file mainly about image processing is **Ui\_childrenform3.py**

First of all, we import the necessary part of Qt (import the QFileDialog to open the files that user want to load). Particularly, we import the module **os** to make a folder to store our processed files. And we import **Qsound** to play the .wav files. Here we introduce cross-platform tools **ffmpeg** and **pydub** to process the .wav files.



**Figure3.8 Module and library import of audio processing**

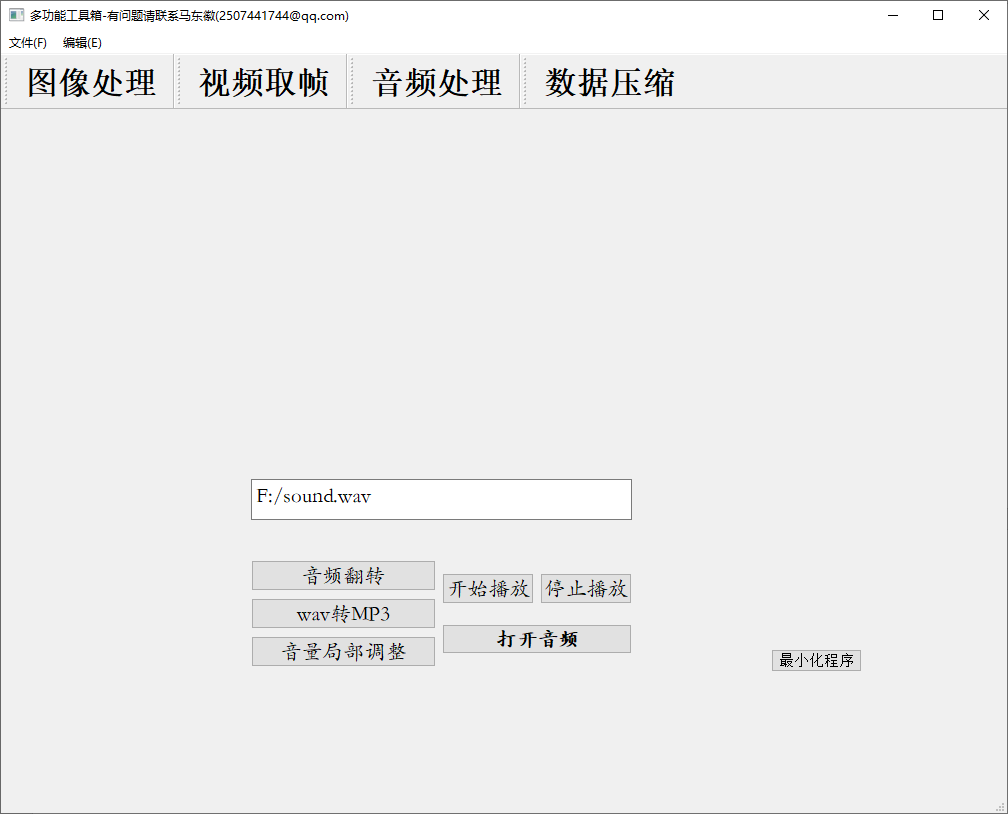
And also we set the slot functions for the buttons in the child window. (process1-reverse/process2-convert/process3-volumeadj/startbutton-playwav/stopbutton-stopwav).



**Figure3.9 Set slot functions for buttons**

And all of these slot functions, you can see in the code file **Ui\_childrenform3.py**. Include playwav(), stopwav(), openfile(), reverse(), convert(), volumeadj(), I have necessary comments after the necessary lines.

And the final process results is shown like this:



**Figure3.10 Final results of audio processing**

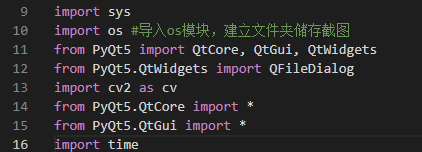
1. **Extract video frames**

Video processing is mainly about the video play and extract frames. And we can also play the frames image on the label. The python file that mainly about video processing is **Ui\_childrenform2.py**



**Figure3.11 Child window of video processing**

Also, we import modules and libraries, we import **OpenCV** and **QThread** to play video, import time to set the video sampling time interval, we import os to make a folder (The path is **D:\vedioshut\**) to store the frame capture.





**Figure3.12 Module and library import of vedio processing**

Like the image, audio processing ,we set the slot functions for the buttons (PlayVedio, FreVedio, Display\_Img, close)



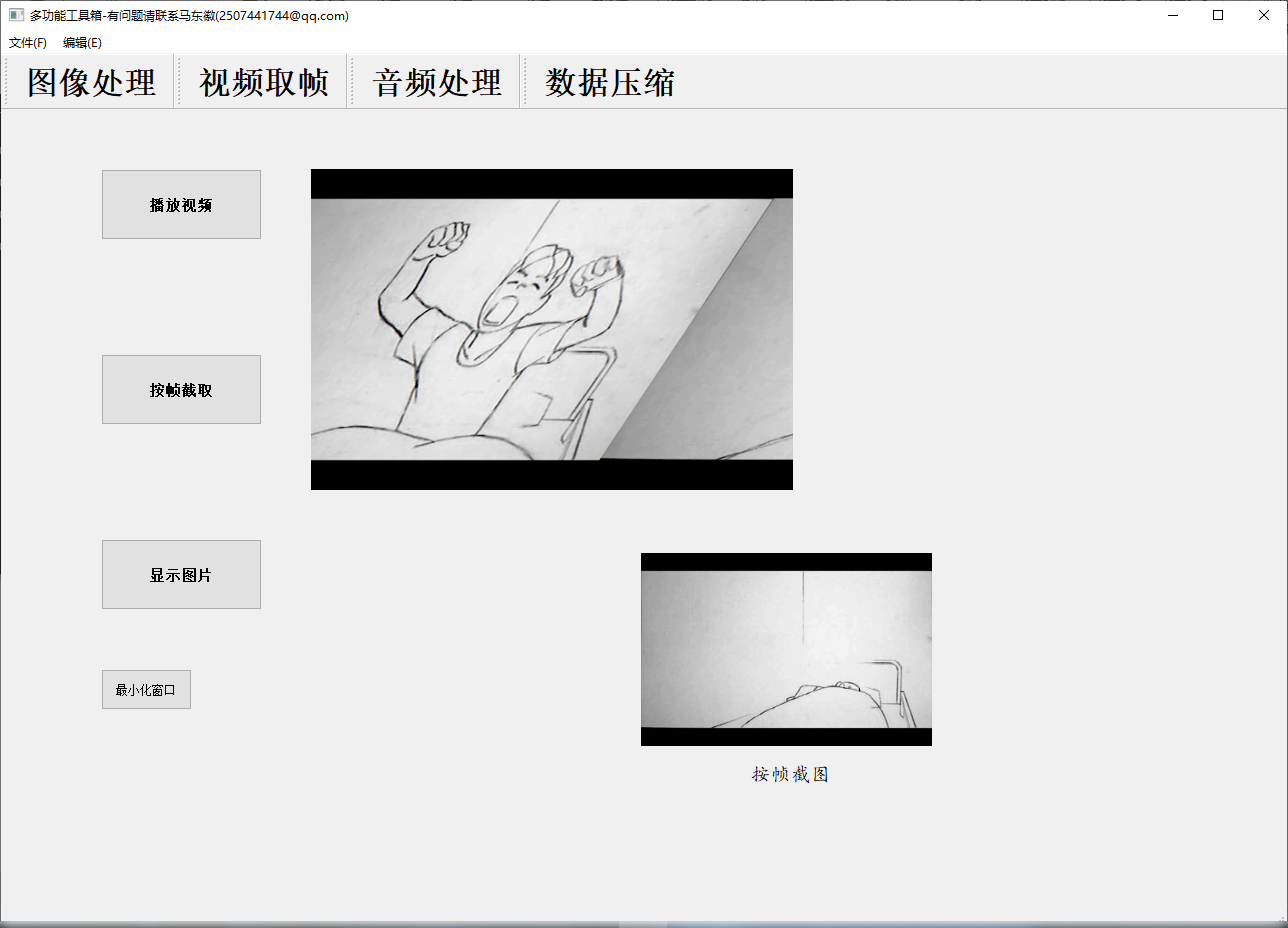






**Figure3.12 Slot functions for the different buttons**

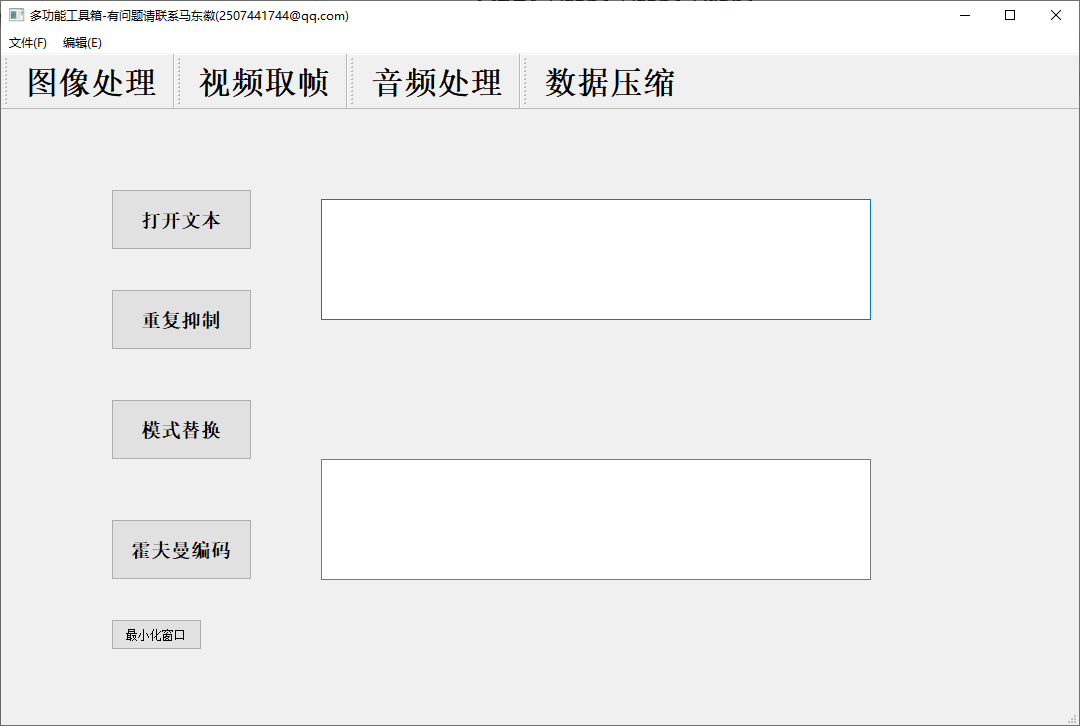
All of these slot function that I define can be seen in the file Ui\_childrenform2.py. And I also made some comments where necessary, and the final result is shown:



**Figure3.13 Final results of video processing**

1. Text compression

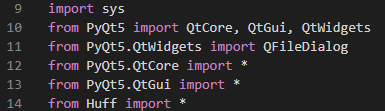
Text compression is mainly about string compression, about the string compression, this project mainly realized three compression methods: **Repetitive Sequence Suppression**(重复序列抑制), **Pattern Substitution**(模式替换) and **Huffman Coding**(霍夫曼编码).



**Figure3.14 Child window of text-compression**

The python files about text-compression are **Huff.py** and **Ui\_childrenfrom4.py,** Huff.py is the algorithm of Huffman coding, about how to generate a Huffman tree, and Ui\_childrenform4.py is about the child window of text compression and the other compress algorithm.

Because we need to use the Huffman algorithm, so we need to import all the classes and functions of Huff.py into Ui\_childrenform.py:



**Figure3.15 Import Huff into Ui\_childrenform**

We also set the slot functions for those buttons:





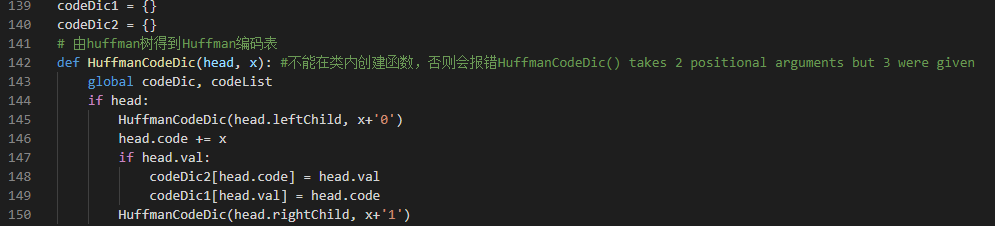






**Figure3.16 Slot functions for buttons in child window**

And for the slot functions(loadtxt(), compress\_1(), compress\_2(), compress\_3() and close), you can also see in the code files, I also do some necessary comments after codes. Specially, I define a function named **HuffmanCodeDic()**. Its function is to generate Huffman code through the Huffman tree obtained by the Huffman algorithm:



**Figure3.17 Huffman coding algorithm**

The final result of this text compression are:



**Figure3.18 Repetitive Sequence Suppression(重复序列抑制)**



**Figure3.18 Pattern Substitution(模式替换)**



**Figure3.18 Huffman coding(霍夫曼编码)**