Report Date:02/10/2023
To: ematson@purdue.edu, ahsmith@purdue.edu, lhiday@purdue.edu and lee3450@purdue.edu
From: SWATTER

- Joonki Rhee (rhe9788@kyonggi.ac.kr)
- Gwangwon Kim (tiger6777@kyonggi.ac.kr)
- Hyunjong Jang (20191580@g.dongseo.ac.kr)
- Minseop Shin (20191520@g.dongseo.ac.kr)

Summary

Over the course of this week, We went over my thesis and made revisions based on the suggestions we received the previous week. We were able to complete the methodology section, but the application component has yet to be developed and could not be included in our writing at this time. Our team has completed the writing of the machine learning code and has started creating the program to run on the server.

What SWATTER completed this week:

- Wrote most of the methodology parts
    - Some parts of the application could not be written.
- Some parts of the methodology need to be modified.
    - If the experimental method changes.
- The development process is ongoing and depicted in Figure 1
    - We have written the details of the paper as shown in Figure 2.
- Request for revision of the 2nd thesis to M. Lee.
- Implement Acoustic Drone Detection Application.
- Finalize machine learning code
    - We can train the machine learning models whenever we get the UAV data.
        - This program returns an accuracy table automatically.
    - We finalized code about save and load model.
    - We started making a program for executing on the server.
        - This program will receive arguments about data and model path.
        - This program will return the result of the detection

- Designed packet protocol.
- Made classes of the server program for handling packets.

Things to do by next week

- Writing Experiments and Results
    - The paper will be written as soon as the development is confirmed.
- All parts will be revised through discussion with developers.
    - May continue to change all parts (abstract, etc.).
- Add drone photos, audio data information, etc. to the methodology part.
- We will receive 200 noise data for 5 seconds.
- Models will be tested by UAV and noise data.
    - We will compare accuracy, f1-score of models and select one model.
    - We will compare features of audio data and select features for processing data.
- We will check sklearn-SVM algorithm and libSVM algorithm are the same.
    - It should be the same environment that testing on application and server.

- We will test calling the ML model in the cloud server.
    - The program implemented in C++ will call the ML model with input data and get results from the model.
- We will test the communication between a client and the server.
    - Client: Android application implemented in Kotlin (java)
    - Server: C++ program
- We will find the method to distribute the ML model to smooth service.

Problems or challenges:

- It is difficult to lower the plagiarism rate because there are previous studies with similar experimental methods.
    - Experiment differently.
- There are limitations due to the lack of data collection and experiments.
- Implementation of Feature extraction on Android Environment.
    - Librosa library is the common way to extract features, but it is a Python library.
    - We will use Jlibrosa for JAVA language on Android environment

- Implementation of Support Vector Machine Model on Android Environment.
    - We expect the Support Vector Machine will be the best result.
    - If CNN or NN models are the best, we will use tensorflow lite library.
    - We will use libSVM library for JAVA language on Android Environment.
- Implementation of Audio Data Transfer with Server.

References

Fig. 1. Flow chart of two methods for drone detection

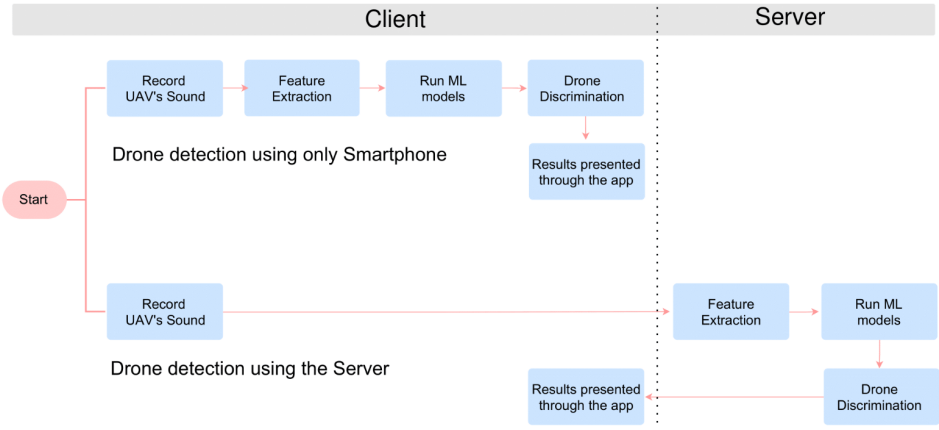
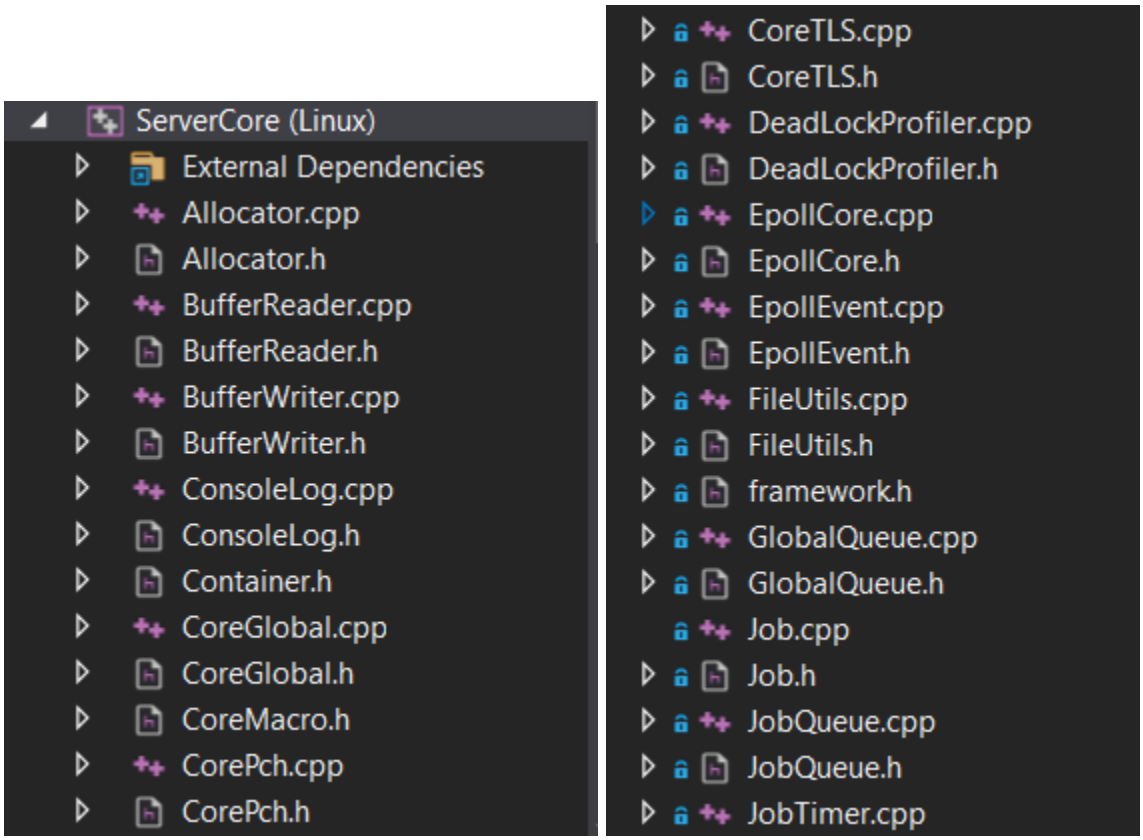


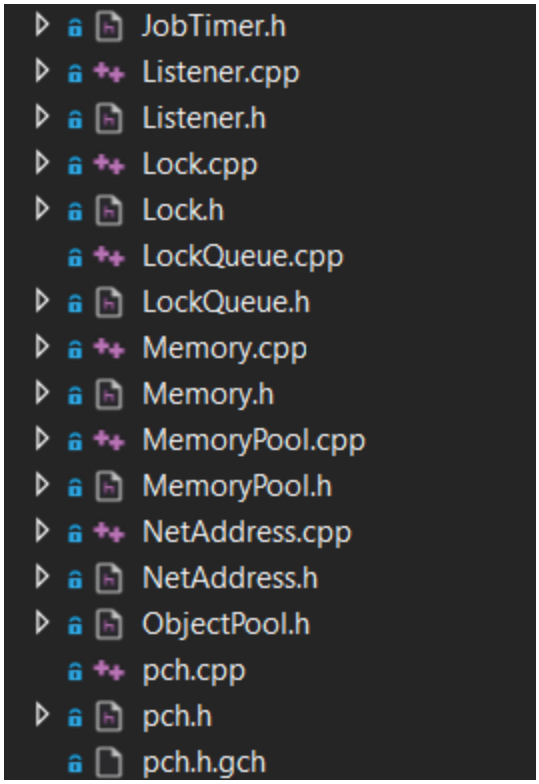
Fig. 2. Drone acoustic feature extraction method

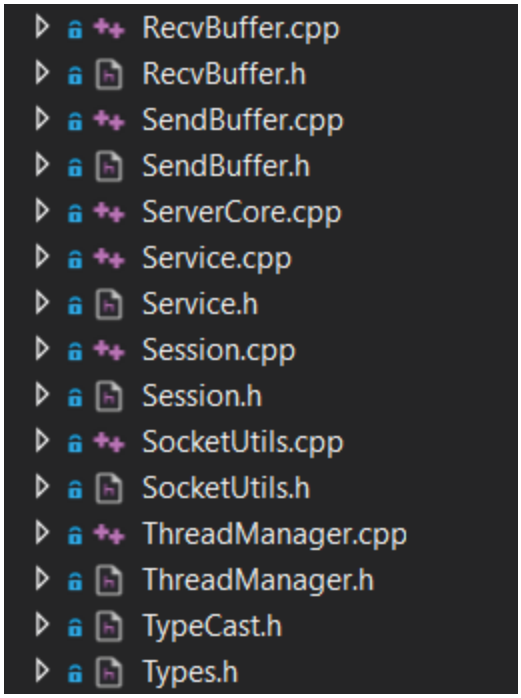
TABLE II
DRONE ACOUSTIC FEATURE EXTRACTION METHOD

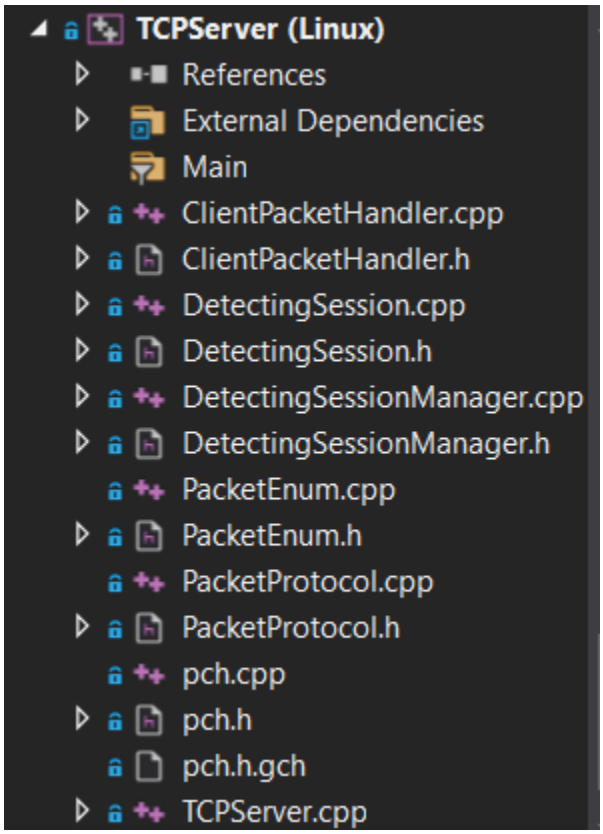
Feature	Shape
chroma_stft	12
chroma_cqt	12
chroma_vqt	12
mel	128
mfcc	40

Fig. 3. Classes of the Server program




JobTimer.h
Listener.cpp
Listener.h
Lock.cpp
Lock.h
LockQueue.cpp
LockQueue.h
Memory.cpp
Memory.h
MemoryPool.cpp
MemoryPool.h
NetAddress.cpp
NetAddress.h
ObjectPool.h
pch.cpp
pch.h
pch.h.gch


RecvBuffer.cpp
RecvBuffer.h
SendBuffer.cpp
SendBuffer.h
ServerCore.cpp
Service.cpp
Service.h
Session.cpp
Session.h
SocketUtils.cpp
SocketUtils.h
ThreadManager.cpp
ThreadManager.h
TypeCast.h
Types.h


TCPServer (Linux)
References
External Dependencies
Main
ClientPacketHandler.cpp
ClientPacketHandler.h
DetectingSession.cpp
DetectingSession.h
DetectingSessionManager.cpp
DetectingSessionManager.h
PacketEnum.cpp
PacketEnum.h
PacketProtocol.cpp
PacketProtocol.h
pch.cpp
pch.h
pch.h.gch
TCPServer.cpp