Task1

1. How the android application works

This android application has five activities, which are as follows.

Activity name	Function
Mainactivity	Plot the main view; Register button listening event; Call camera and show the camera stream on the screen; Redraw the bundle boxes;
SettingsActivity	Plot the SettingsActivity view;
SettingsServer	Set server's IP and port
settingsAbout	Show Application about information
StatsActivity	Show performance information of this system

Mainactivity is the main activity of this application. By using Intent class, we can change to different activities. Mainactivity also override seven callback methods of life cycle of activity, such as onCreate(), onStart(), onPause(), onStop(), onDestory(), onRestart().

Constants class saves lots of parameters. Other classes can import it and use final parameters globally.

ARManager class needs to tackle remote communication and data saving tasks, by using ReceivingTask, TransmissionTask, and DatabaseHelper class.

Detected class is an information format of recognized object. Any information of recognized objects will be saved to an instance of Detected class.

The resolution of my phone is vertical and the preset resolution of this android application is horizontal. So, I rewrite the onDraw function, which will plot the buddle boxes on the SurfaceView. When ReceivingTask function receive data from server, the sub thread will tell main thread to update UI, because the sub thread cannot update the UI directly. In a sub thread, we must asynchronous message processing mechanism to achieve this goal, and Handler class is a good choice.

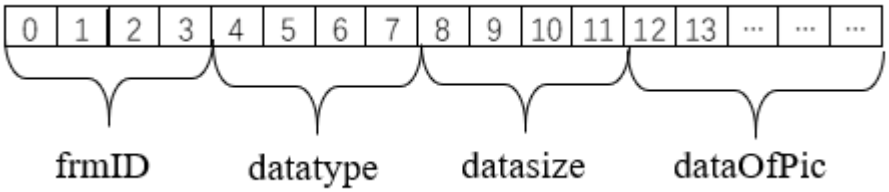
This application also uses multithread technology. In ARManager class, the application creates two threads. The first one is an instance of TransmissionTask class, which is used to send data to server alone. The second one is an instance of ReceivingTask class, which can receive data from server alone. This synchronous message processing mechanism can reduce waiting time.

2. About my AR server

I choose python language to build my AR server, because I can use yolov5 model easily in python language. Besides, it is easier to build a project in a short time.

a. Receiving data by UDP.

The client will send a java byte array to server by using UDP. The first thing we need to do is to receive data by UDP. The java byte array's format is as follows.



We can use python socket to get byte data. And by using this format showing as the figure, we can get frame ID, datatype, datasize, data of picture. Then, we need to transfer these byte data, including frame ID, datatype and datasize to int, and transfer data of pictures to unit8. We use Numpy and OpenCV-python to finish this transformation. And besides, the resolution of my phone is vertical and the preset resolution of this android application is horizontal, so I need the transpose and flip operation. But if you devices' resolution is horizontal, you will not need transpose or flip operation.

b. Using yolov5 model

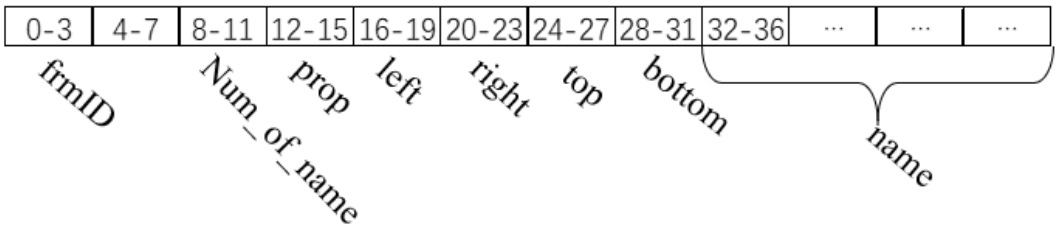
Then, we introduce yolov5 to our code, by using "torch.hub.load()". We need input data of pictures to yolov5 model, and receive the results of recognition.

The results and transfer to Pandas format. The results include many parameters, such as the location, confidence, and name of an object.

c. Send data to android application

After getting results, we need to build a python bytearry, which will be used to prepare for a sending data format.

The receiving format of android application is as follows.



Except the name of this object, other parameters, such as frame ID, num of name, all use 4 bytes to save. So, we need to transfer these int or float parameters to 4bytes. However, the confidence parameter is float type, we need to change it to int. Otherwise, this confidence parameter cannot save only use 4bytes. And the android application's receiving code:

detected[i].prob =
(float)(ByteBuffer.wrap(tmp).order(ByteOrder.LITTLE_ENDIAN).getFloat());

must change to:

detected[i].prob =
(float)(ByteBuffer.wrap(tmp).order(ByteOrder.LITTLE_ENDIAN).getInt());

Otherwise, the android application will have Exception.

d. Save the results

We need to save the recognized picture with buddle boxes and other information, such as location, confidence information on our server. The yolov5 model have help us save recognized pictures with buddle boxes easily. And also, we recode other information to database. Sqlite3 is a good choice.

In my program, we set 2 flags, including “save_results_pic” and “save_results_sql”. If we set “save_results_pic = True”， the recognized picture with buddle boxes will be save at “./run/hub”. If we set “save_results_sql = True”, the other information will be save to a Sqlite3 database.

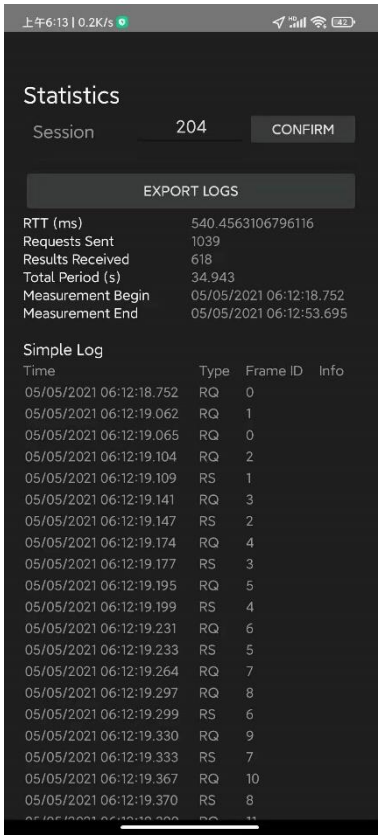
e. Using multithreading

Saving results on server will have lots of I/O operation, which will waste lots of time if we use only one thread. Therefore, I create two other threads. The first one is used to save recognized pictures with buddle boxes alone. The second one is used to execute SQL operation alone.

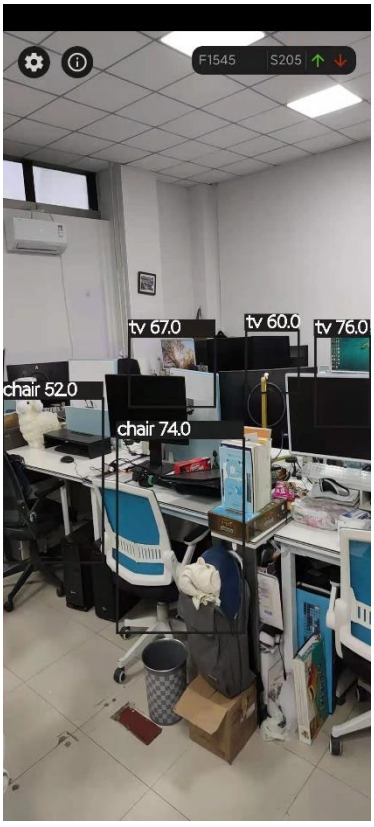
3. Perform analysis on the system

a. Performance

I test my system over a local WIFI access point. I set “save_results_sql = True” and “save_results_pic = True”.



Statistics results



recognized example 1

recognized example 2

From the figure, we can see the speed is 29.73 frame per second, and the RTT is 540.45ms, which is a little high. Packet loss rate is 40.52%. I think, maybe saving bundle boxes and executing SQL use lots of time.

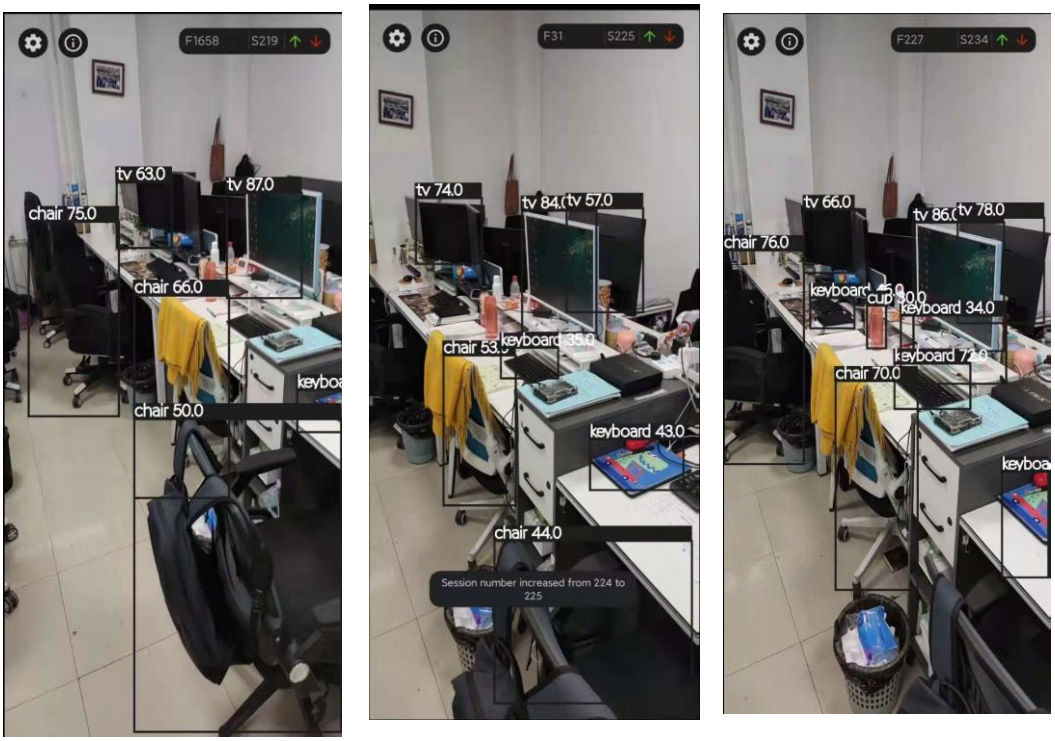
What about other cases, if we don't need to save picture with bundle boxes or need to recode other information to database? The results is as follows.

flags	RTT (ms)	Packet loss rate	Frame speed (/s)
save_results_pic=True save_results_sql = True	540.45	40.52%	29.73
save_results_pic=False save_results_sql = True	367.39	13.26%	29.83
save_results_pic=True save_results_sql = False	463.87	34.97%	29.79
save_results_pic=False save_results_sql = False	278.74	2.65%	29.79

We can conclude that saving pictures with bundle boxes needs lots of time, if we only recode other information, such as location, confidence, the TRR will reduce 32.04%。

b. Threshold comparison

In this system, we use pretrained yolov5x. So the results should be good. From the figure above, this system can analyse most of objects. In this system, if confidence is less than 0.5, we will not show the bundle boxes of them. If we reduce this threshold, the system will show more objects. (The original size of bytes buffer is 512, which can show no more than 5 recognized results. If we want to show more recognized results on the SurfaceView, we must change the final parameter REG_SIZE of Constants class)



Threshold = 0.5 Threshold = 0.3 Threshold = 0.1

We can find, with the decrease of Threshold, the number of recognized objects will increase.

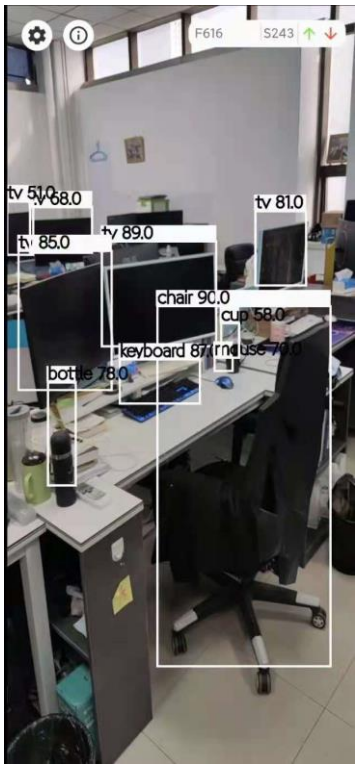
c. Resolution comparison

The original resolution of send picture is (1920/7)*(1080/7)=274*154. If we increase the resolution, what about the results.

We increase the resolution of send pictures, and observe the number of recognized objects. We can find, when we increase the resolution of send picture, the number of recognized objects will increase.

274*154

384*216



640*360