MULTI-MODAL THERMAL POINT CLOUD SYSTEM USER GUIDE

Table of content

Multi-modal Thermal Point Cloud System User Guide

- 1 General Notes
  - 1.1 Intended use
  - 1.2 System composition
- 2 Calibration and alignment
  - 2.1 multi-modal sensors calibration
  - 2.2 multi-modal sensors fusion
    - 2.2.1 2D fusion
    - 2.2.2 3D fusion
    - 2.2.4 Point Cloud Result Save and Registration
- 3 Thermal Point cloud GUI Software Usage
  - 3.1 Environment setup
  - 3.2 Software UI description
  - 3.3 ROS Read TPCD Data
- 4 Trouble shooting

1 GENERAL NOTES

1.1 Intended use

The multi-modal thermal point cloud system based on the emitted infrared energy of objects, to detect the depth and temperature of observing object, combining frames in visible light, to construct the thermal point cloud.

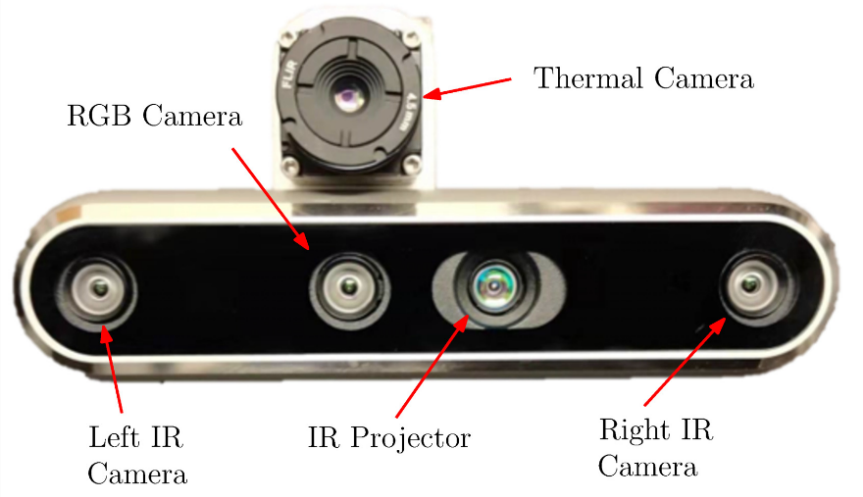
1.2 System composition

- Hardware

> The hardware system consists of two parts, one is a sensing part consisting of multimodal sensors and another is a thermal chessboard part for the calibration of the multimodal sensor system.

1. Multimodal sensors system

The sonsor system consists of a depth camera and a thermal camera. In our approach, we use a Intel **Realsense D455** depth camera and a **FLIR Boson 320** thermal camera.

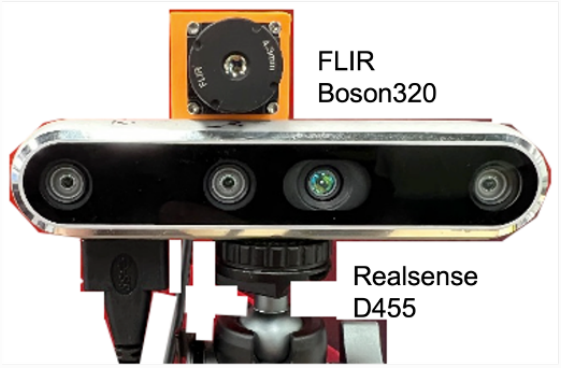


with the parameters:

Sensor Model	FoV	Resolution	Frame rate	Wavelength
FLIR Boson320	52deg.	320×256	60 FPS	$8 - 14\mu$m
Realsense D455	87deg.	640×480	30 FPS	$380 - 800$ nm

(1)

Use the aluminum frame shown below on the left for fixing, or the 3D printed frame shown on the right for fixing are both possible.



Except keeping the camera firmly fixed, the only thing you need to pay attention to is to adjust the orientation of the depth and thermal cameras so that their FOV centers have the maximum overlap as the below figure.

- Software

We give the following directory tree for the software:

```
.
├── CMakeLists.txt
├── calib
├── calibration
├── include
├── qt_develop
├── ros
├── savedpcd
├── src
├── tree.md
└── video
```

CMakeLists.txt: Cross-platform compilation of programs is managed through Cmake, this cmakelists is used for the pointcloud generation C++ program.

calib: calibration parameters for test

calibration: calibration tools' scripts folder

include: Headers folder

src: C++ program source code folder

qt_develop: The QT GUI program folder

ros: The ROS package for reading the thermal point information we select in the qt software

savedpcd: folder to save the current point cloud and script for registration

video: Demo videos of the system

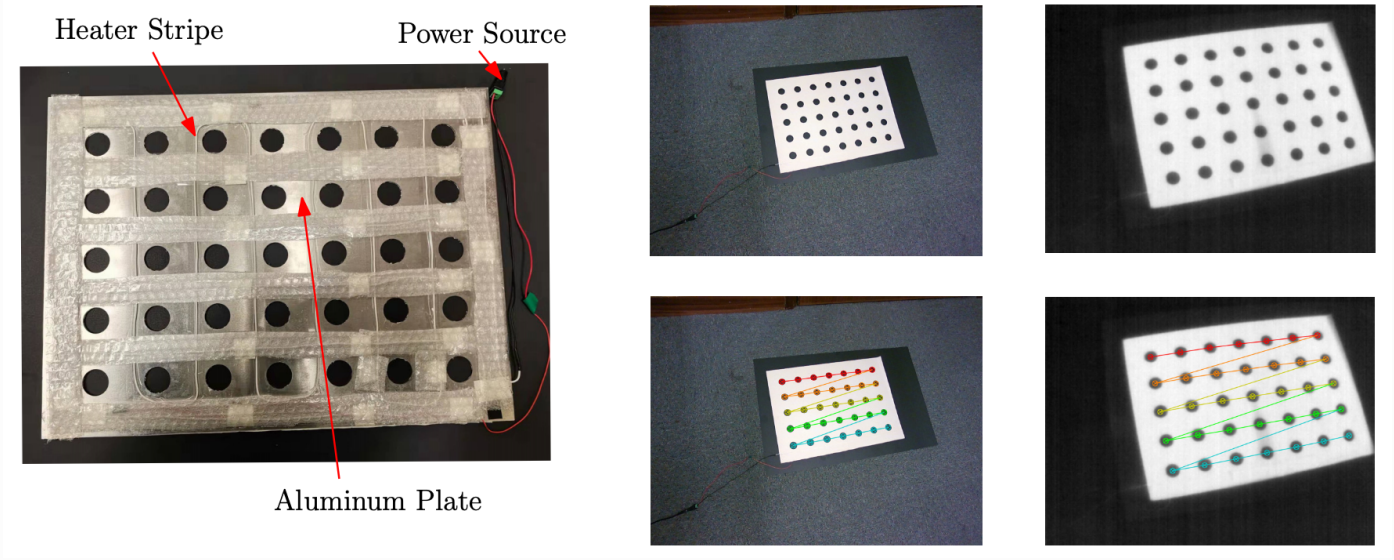
The specific environmental requirements and software configuration will be explained in detail in the corresponding section later.

2 CALIBRATION AND ALIGNMENT

2.1 multi-modal sensors calibration

- Hardware preparation

We use the self-made calibration tool to calibrate the whole system. The tool is shown below:



The main part of the calibration tool is a rectangular aluminum plate with even-distributed circle holes on it and covered with white paper on the front. Heater strip is placed at the back of the plate. Heater strip will heat aluminum plate to make sufficient contrast for the thermal camera.

- Software

We use a python script for calibration.

The script is `thermal_pointcloud/calibration/get_img_d455_boson.py`

To run this script, you should firstly configure the running python environment. We recommand to use `Anaconda` as environment and package management tool.

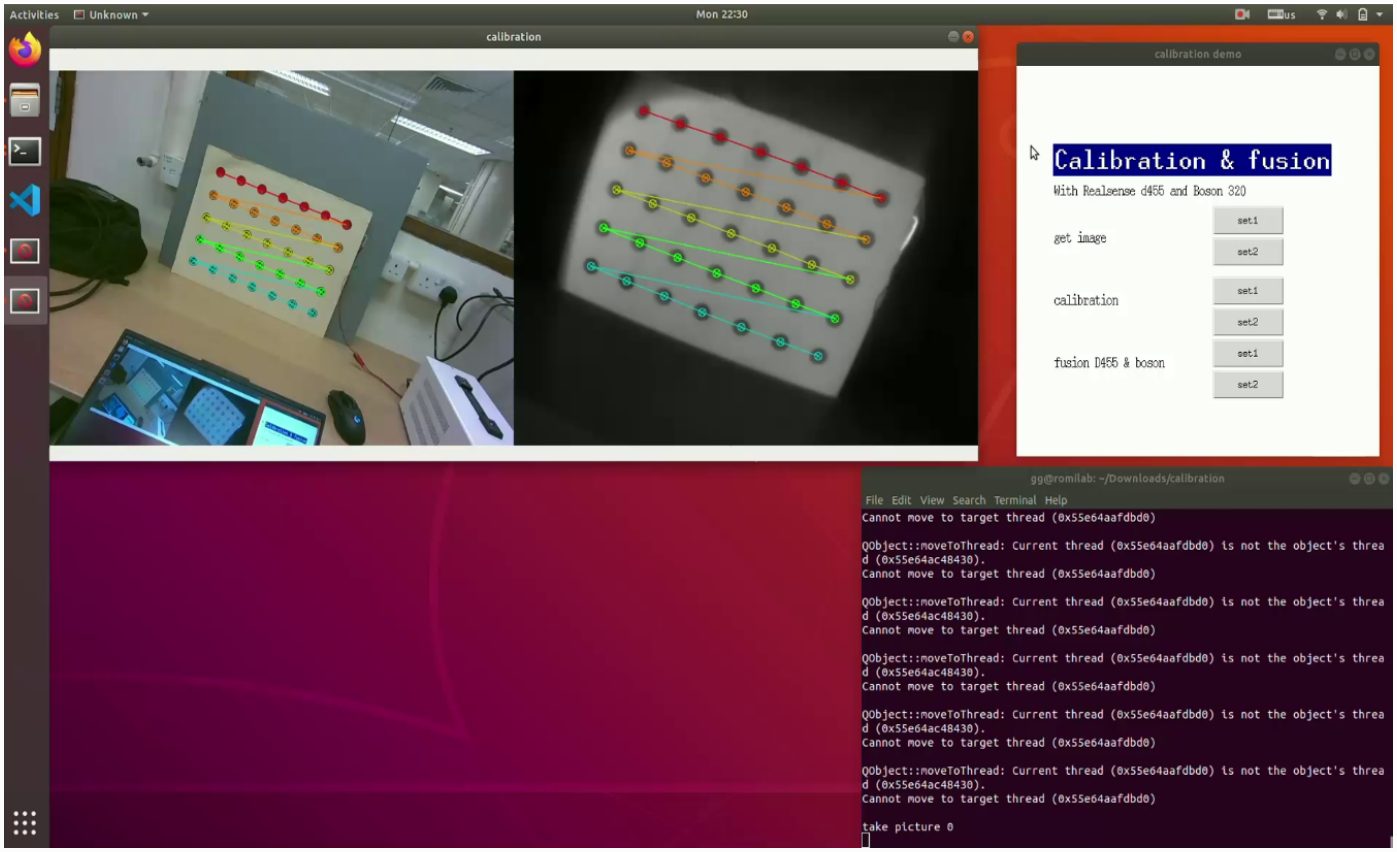
```
# Download miniconda3
cd ~
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-
x86_64.sh
sha256sum Miniconda3-latest-Linux-x86_64.sh
# Verify that the output matches the one online
sh Miniconda3-latest-Linux-x86_64.sh
#clean
rm ~/Miniconda3-latest-Linux-x86_64.sh
```

Create packages:

```
# Create env
conda create --name thermalpcd python=3.8
source ~/.bashrc
# activate env
source activate
conda activate thermalpcd
# essential pkgs
pip install pyrealsense2
pip install flirpy
pip install opencv-python
apt install v4l-utils
```

After the environment set up, you can use the script to calibrate. Also, you can use the `GUI.py` to calibrate two groups of parameters.

The calibration program is running as below:

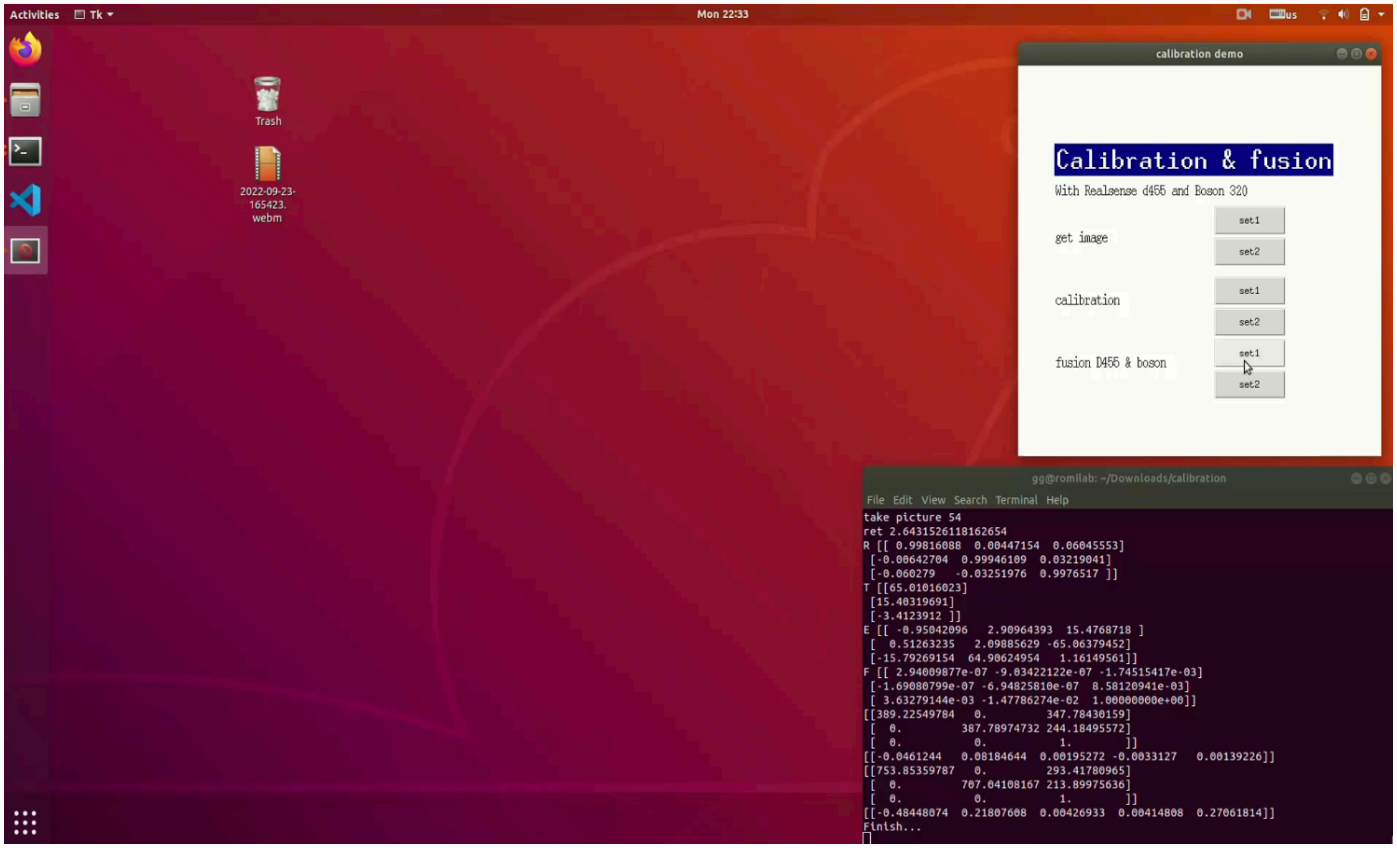


After selecting the calibration window in the upper left corner of the figure with the mouse, whenever you adjust the camera's pose, keep the camera pose fixed and press the `s` key on the keyboard. If detected enough feature points on both RGB and thermal image of the chessboard, the program would return the sentence "Take picture `num` ", otherwise, it will return a prompt telling that it is not detected feature points.

In our calibration progress, we require taking 55 pairs in total, after that, press `q` to quit the get calibration image program. **Note that the picture series start from zero, so we should finish the calibration after the prompt reports the 54th pictures has taken.**

> This progress costs about 5 minutes.

After getting image procedure, we then run the script `thermal_pointcloud/calibration/fusion_D455_boson.py` for getting the calibration parameters, or use the `gui.py`. The program will load the output of the former step and output the calibration parameters in the `.xml` format files in the folder, also shows in the terminal for examining as the following figure.



And the generated `.xml` calibration files are listed:

`boson_realsense_RGB_intrinsic.xml`

`boson_realsense_RGB_distortion.xml`

`boson_realsense_Thermal_intrinsic.xml`

`boson_realsense_Thermal_distortion.xml`

`boson_realsense_Ralative_rotation_matrix.xml`

`boson_realsense_Ralative_translation_matrix.xml`

In the same folder with the scripts.

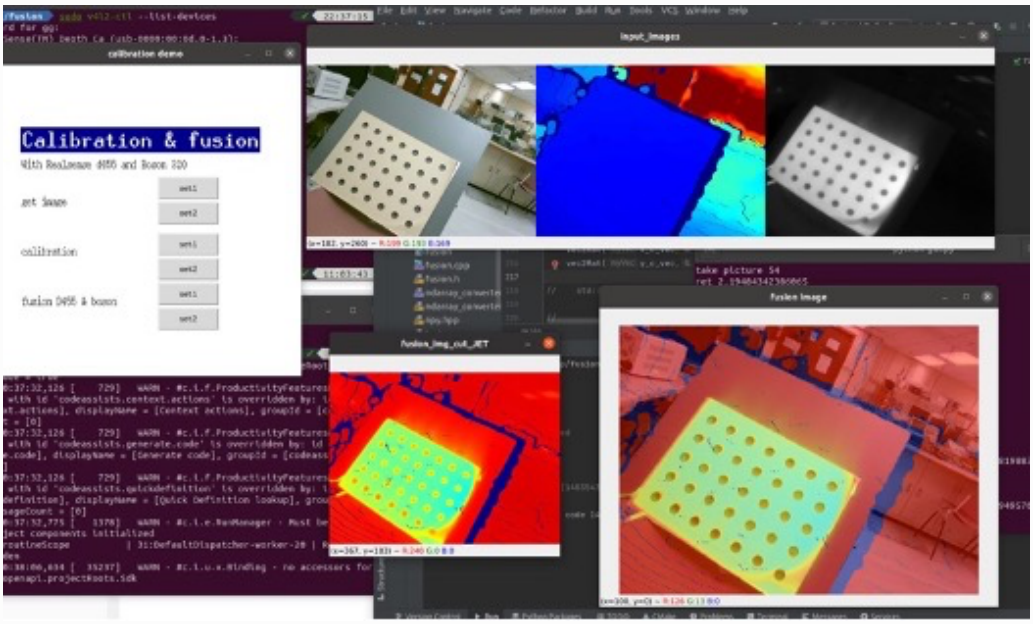
2.2 multi-modal sensors fusion

> After calibration, we will give the fusion results here both in 2D (python) and 3D (python and C++) in realtime.

2.2.1 2D fusion

Also in the `thermalpcd` conda environment we run the script `thermal_pointcloud/calibration/fusion_D455_boson.py` , or use the `gui.py` . The program will load the calibration files in folder `test_data` or `test_data_2` .

The output is shown below:



The above thre images in a row show the RGB, depth and the thermal output.
The lower two images from left to right show the alignment result from thermal to RGB plate and the fusion result of RGB, depth, thermal cameras.

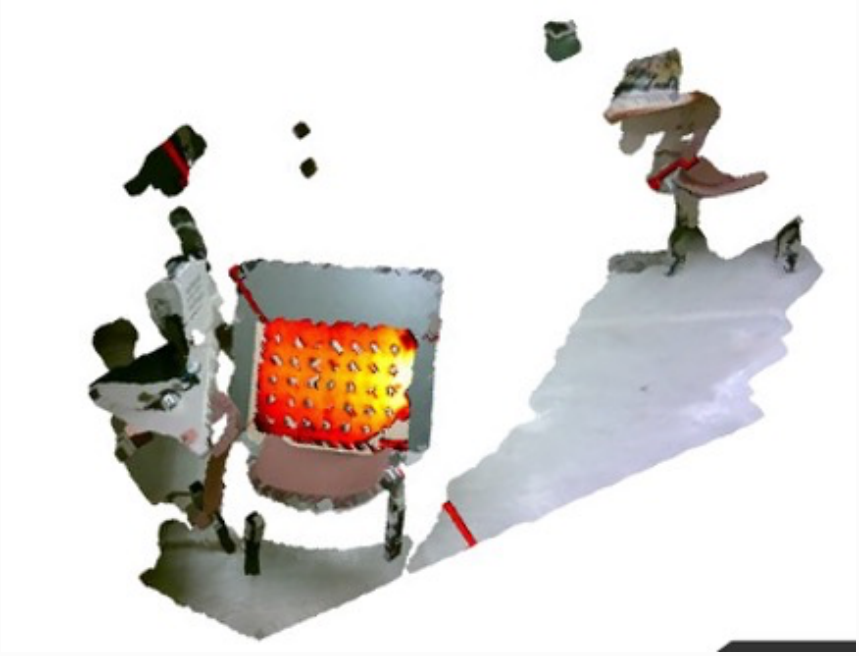
2.2.2 3D fusion

- 3D fusion in python

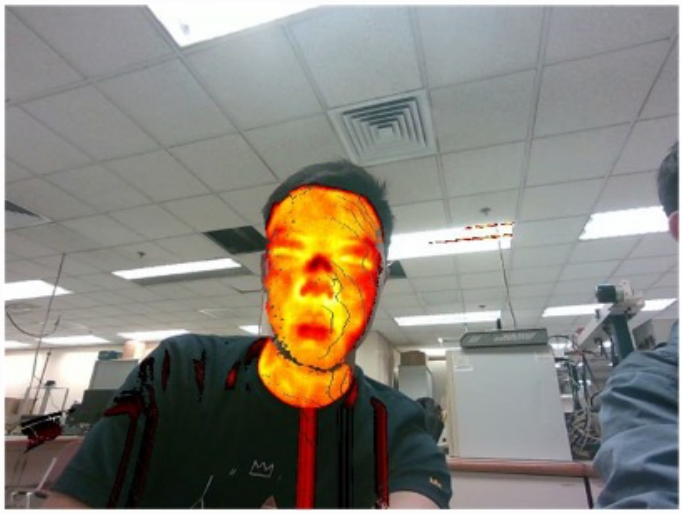
First install the package for visualizing the point clouds in python

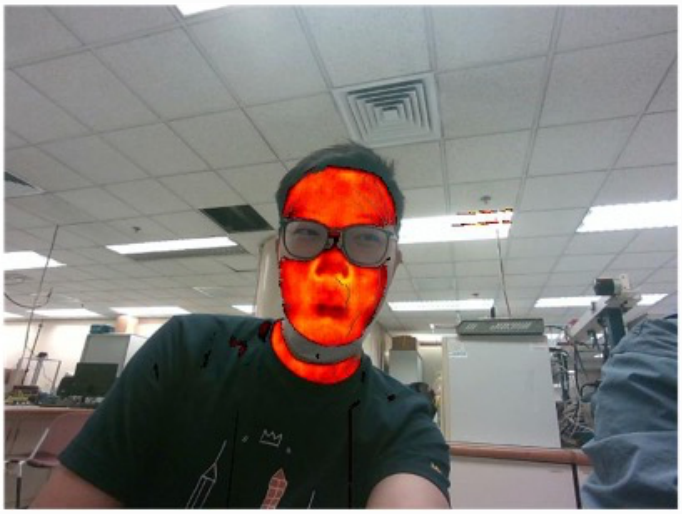
```
pip install open3d
```

Then, run the script `thermal_pointcloud/calibration/HOT.py` , then we can see the fusion thermal pointcloud result in python open3d:



Also, we can run the script `thermal_pointcloud/calibration/take_photo.py` to save the intermediate results of the RGB and thermal fusion image for check:





- 3D fusion in C++

The C++ environment requires the following libraries with the specific version:

- CMake version>=2.8
- Realsense2 SDK
- PCL 1.11
- Opencv 3.4.x

1. The realsense SDK can be find in https://github.com/IntelRealSense/librealsense, and please install it with the instruction. You can use the realsense-viewer to check the installation.
2. The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing. You can find the ver 1.11 source code in the link https://github.com/PointCloudLibrary/pcl/releases to build and install it.
3. OpenCV 3.4.13 source code can be found in https://opencv.org/releases/

After installing the above essential library, We next navigate to the root of the code folder, and edit the `CMakeLists.txt`

```
# Link libraries ------------------------------------------------------------
target_link_libraries(fusion
        "/home/gg/librealsense/build/librealsense2.so"
        ${PCL_LIBRARIES}
        ${OpenCV_LIBS}
)
```

Change the `librealsense.so` file path to yours.

Then, check boson's device port with `v4l2-ctl`

```
v4l2-ctl -d /dev/video- --all
```

According the result, change the port in `fusion_new.cpp`

```
// Video device by default
sprintf(video, "/dev/video8");
sprintf(thermal_sensor_name, "Boson_320");
```

Move the calibration files generated in 2.1 to `thermal_pointcloud/calib` folder, the detailed files are shown as:

Then, navigate to the root folder.

```
mkdir build
cd build
cmake ..
make
```

Then, the C++ visualizing program is built, you can see the exacutable file `fusion` in the `build` folder. Run it with

```
./fusion
```

It shows the real-time thermal pointcloud of the multi-modal system. You can use your mouse to toogle the perspect of the view and zoom in/out.

In file `src/fusion_new.cpp` , you can also set the filter on/off by changing this part of code's comment.

```
        pcl_generator(cloud,cut_img,cut_depth);

        // add filter
        pass.setInputCloud(cloud);
        pass.setFilterFieldName("y");
        pass.setFilterLimits(0.4,0.9);
        pass.setFilterFieldName("x");
        pass.setFilterLimits(-0.8,0.65);
        pass.filter(*cloud_filtered);

        viewer->removeAllPointClouds();
        viewer->addPointCloud(cloud,"Cloud Viewer");
        viewer->updatePointCloud(cloud,"Cloud Viewer");

        // visualize pass through filter results
//        viewer->addPointCloud(cloud_filtered,"Cloud Viewer");
//        viewer->updatePointCloud(cloud_filtered,"Cloud Viewer");
```

Also, in this part of code, the thermal point cloud saved in the memory of pointer `cloud`.

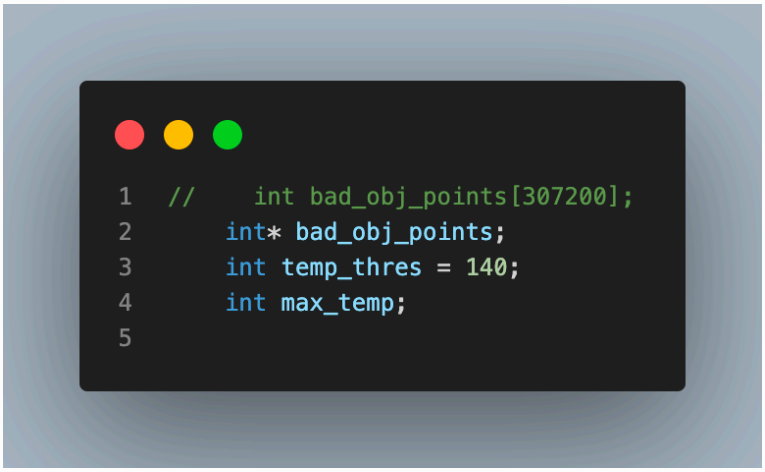
Additionally, there are some configurable parameters for point cloud generation:

1. Fusion Point Cloud Size

This depends on the overlap region size of the depth camera and RGB camera. It's decided the maximum overlap pixels size of them. With the fixed frame designed by us for Realsense D455 and FLIR Boson 320, it is set to 390*280. You can resize this for your cameras' settings.

2. Temperature threshold

This is a threshold set for the visualization for the thermal fusion point cloud. It controls the visualization selection of RGB and Colormap. You can reset through changing `temp_thres`:

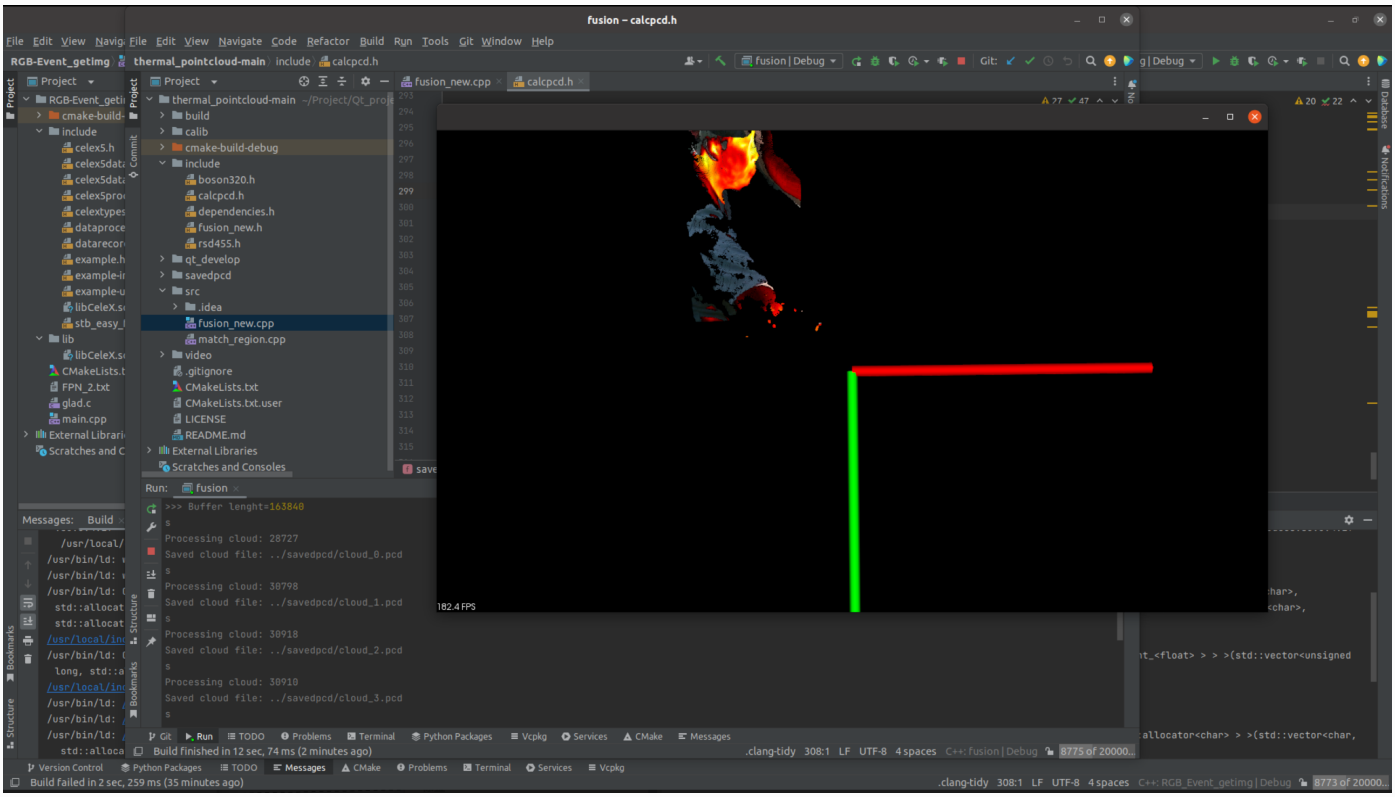


The value of this threshold varies linearly with temperature.

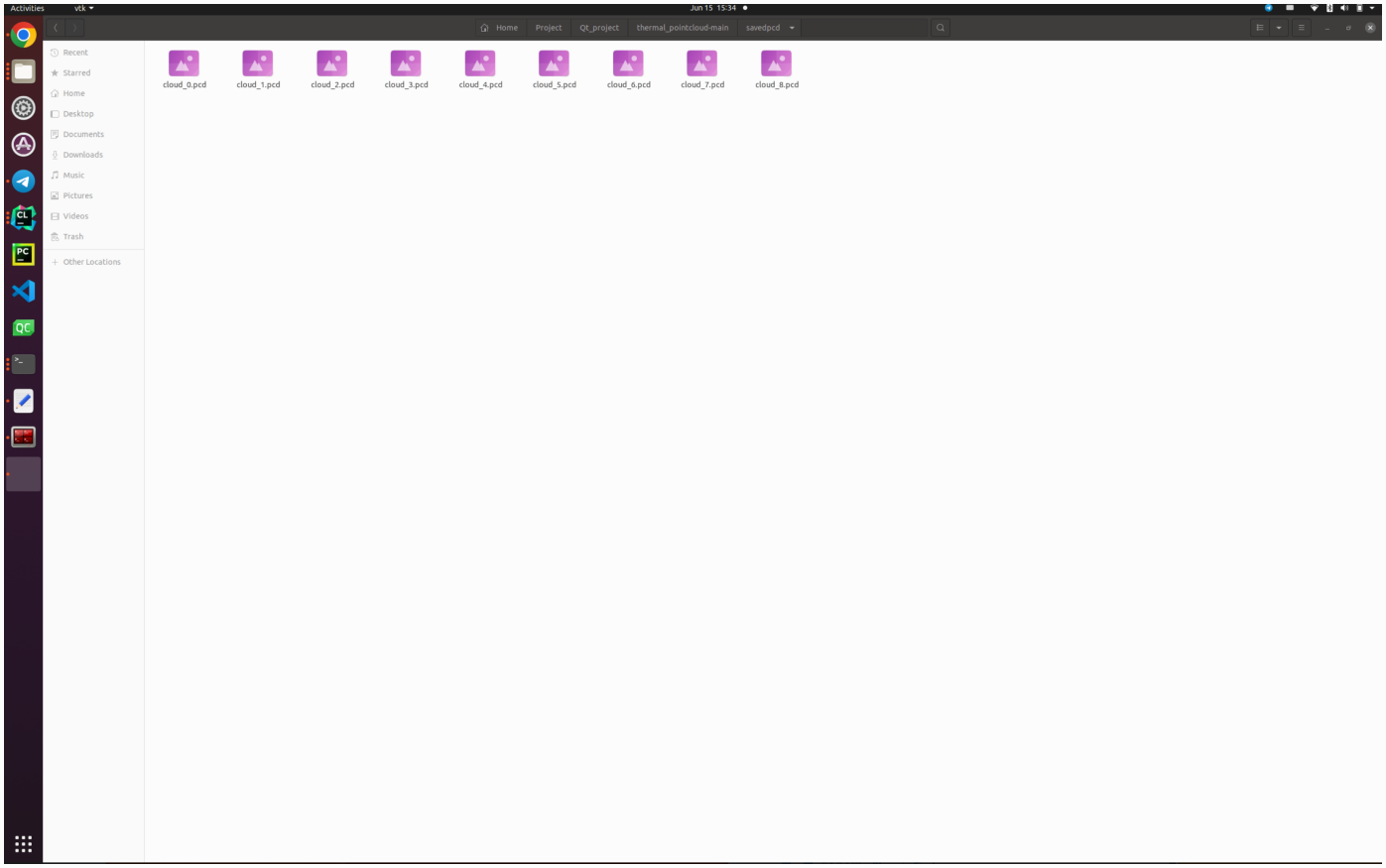
2.2.4 Point Cloud Result Save and Registration

In this part, we use the C++ program to save the intermediate thermal point cloud result, and use the python script for point cloud registration.

When running the C++ 3D fusion main program, you can input `s` and click `Enter` to save the current point cloud . The UI is shown below:



and the point cloud saved into the folder `thermal_pointcloud/savedpcd` :



The folder contains 10 `.pcd` files in limit, and the new generated point cloud would overwrite the old one.

While the `savedpcd` folder has more than two `.pcd` files, you can use the python script `registration.py` in the folder to registrate all the point clouds in the folder, and the result will be saved into `result.py`, the registration result in our test is shown as below.

3 THERMAL POINT CLOUD GUI SOFTWARE USAGE

3.1 Environment setup

The GUI software environment is similar to that C++ one, we only need to add

- Qt 5.14.0
- VTK 8.2.0

1. Qt 5.14.0

We can download the archive of qt 5.14.0 from https://download.qt.io/archive/qt/ and install in the path: `/opt` , and then run the following command:

```
# check system default qt version
qmake -v

# change system default qt version
cd /usr/lib/x86_64-linux-gnu/qt-default/qtchooser
sudo vim default.conf

# change the following contents
/usr/lib/x86_64-linux-gnu/qt4/bin
/usr/lib/x86_64-linux-gnu

# to
/opt/Qt5.14.0/5.14.0/gcc_64/bin
/opt/Qt5.14.0/5.14.0/gcc_64

# Check the qt version again
qmake -v
```

Then add the new Qt into the environment path:

```
sudo vim /etc/profile

# add
export PATH=/opt/Qt5.14.0/Tools/QtCreator/bin:$PATH
export PATH=/opt/Qt5.14.0/5.14.0/gcc_64/bin:$PATH

# refresh the profile
sudo source /etc/profile
```

2. VTK 8.2.0

Download the source code from https://vtk.org/download/. The release path is `~/software`

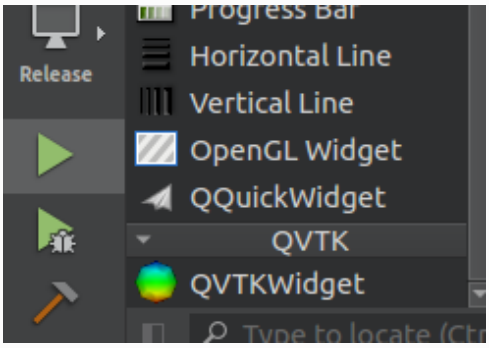
Compile and install:

```
cd ~/software/VTK-8.2.0
mkdir build && cd build

cmake -DVTK_QT_VERSION:STRING=5 \
-DVTK_Group_Qt:BOOL=ON \
-DBUILD_SHARED_LIBS:BOOL=ON \
-DQT_QMAKE_EXECUTABLE:PATH=/opt/Qt5.14.0/5.14.0/gcc_64/bin/qmake \
-DCMAKE_PREFIX_PATH:PATH=/opt/Qt5.14.0/5.14.0/gcc_64/lib/cmake ..

make -j8
sudo make install
```

and generate the QVTKWidget:

```
cd ~/software/VTK-8.2.0/build/lib
sudo cp ./libQVTKWidgetPlugin.so
/opt/Qt5.14.0/5.14.0/gcc_64/plugins/designer
sudo cp ./libQVTKWidgetPlugin.so
/opt/Qt5.14.0/Tools/QtCreator/lib/Qt/plugins/designer
```

And we can check the installation by reopen the Qt Creator and check the QVTKWidget:



Then, we can go into the Qt Creator to open our software project by open `thermal_pointcloud/qt_develop/CMakeLists`.

After this, we change the following line of `CMakeLists.txt` in this folder

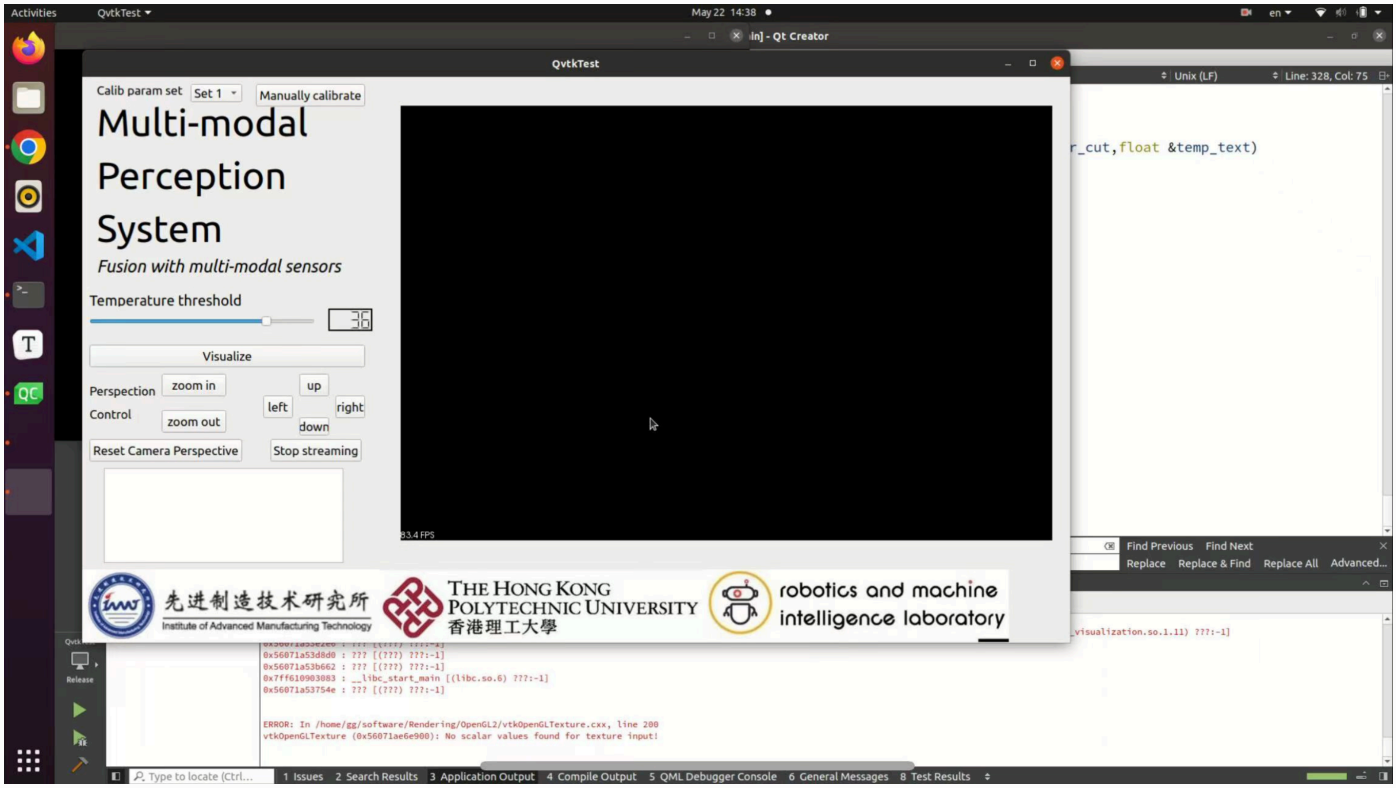
```
target_link_libraries(QvtkTest PRIVATE Qt5::Widgets
    ${QT_LIBRARIES}
    ${PCL_LIBRARIES}
    ${VTK_LIBRARIES}
    "/home/gg/librealsense/build/librealsense2.so"
    "/home/gg/software/build/lib/libvtkGUISupportQt-8.2.so"
     ${OpenCV_LIBS}
    )
```

to your path to this two link libraries.

Then, you can build and run the project in `Release` mode. Also, you can directly navigate into `qt_develop/build` folder to run build command:

```
cmake ..
make -j8
```

to build in a quicker way. If no error occurs, the UI will appears:



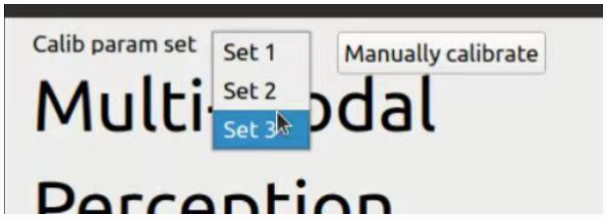
3.2 Software UI description

The software can be started by pressing the button `Visualize` :

and the real-time 3D thermal point cloud will be shown in the `QVTKWidget` in the right.

Then, we introduce the functions of the UI from the left above to bottom.

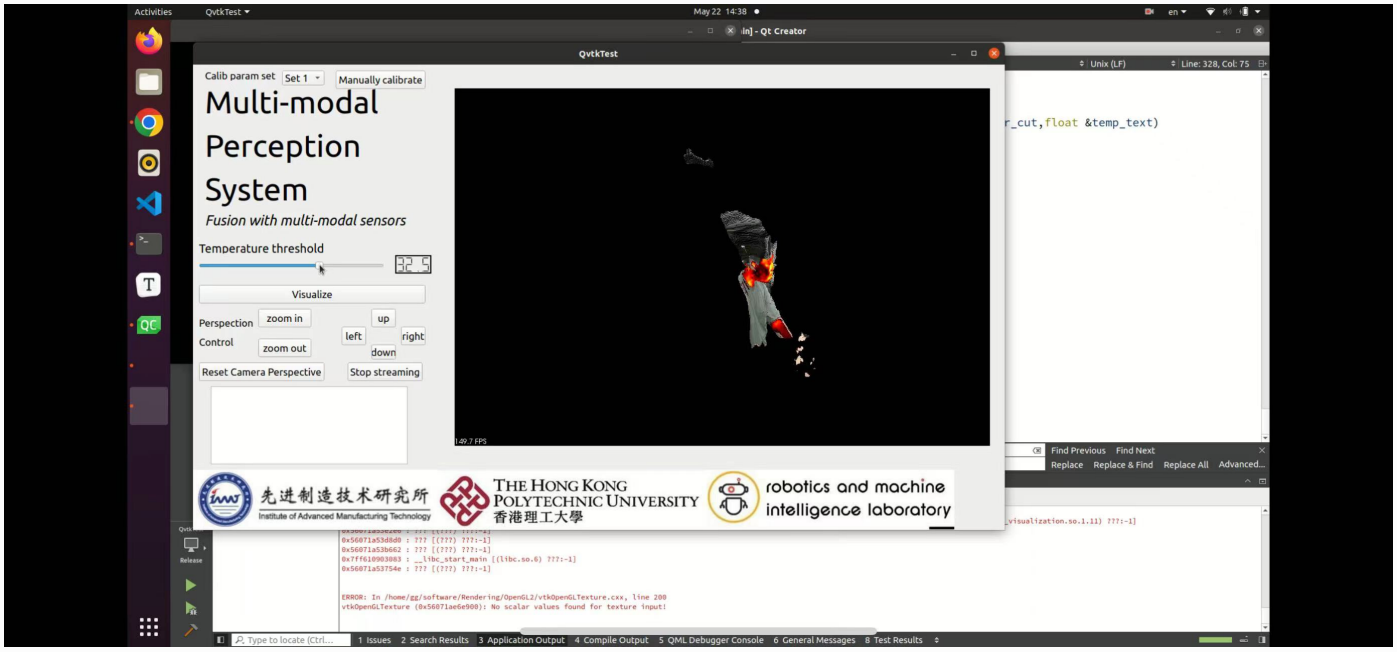
1. Calib param set `Set` toogle menu can select different set of calibration parameters for different cameras settings.



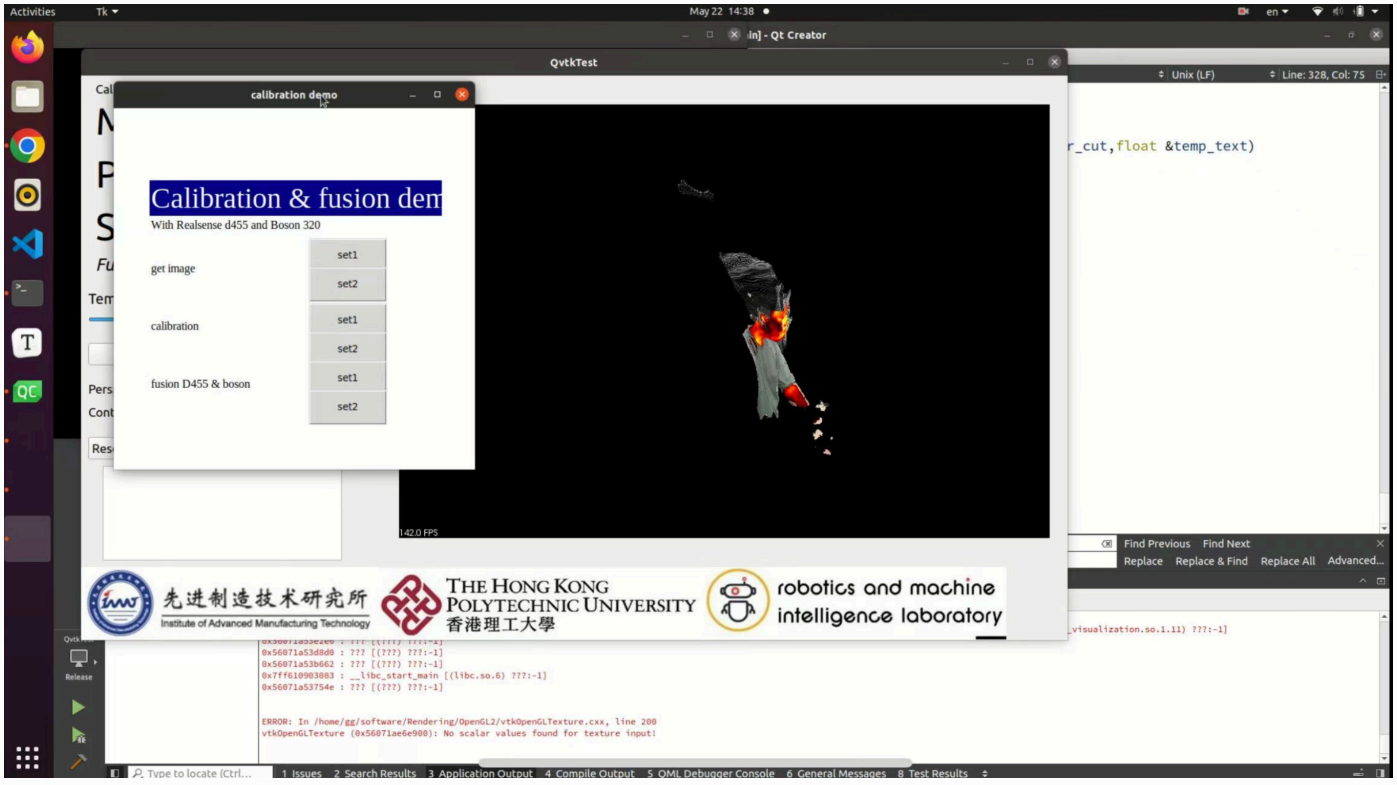
2. Button `Manually calibrate` will run the calibration GUI program used in 2.1 for getting a new set of calibration parameters.

The temperature threshold slider bar controls the threshold temperature of visualizing RGB or thermal heat colormap, the unit is Celcius degree. The effect is shown as below:

32.5 celcius degree:

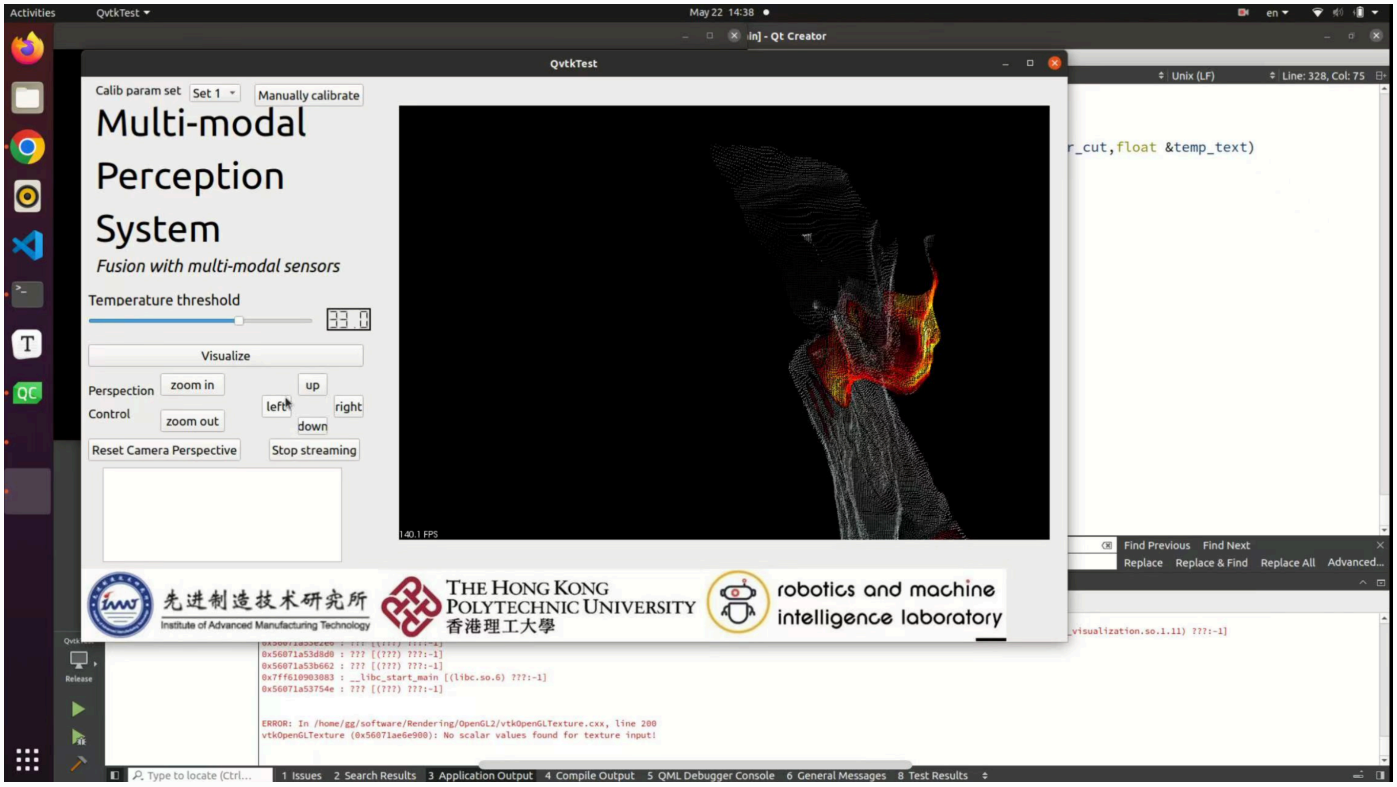


22.7 Celcius degree:

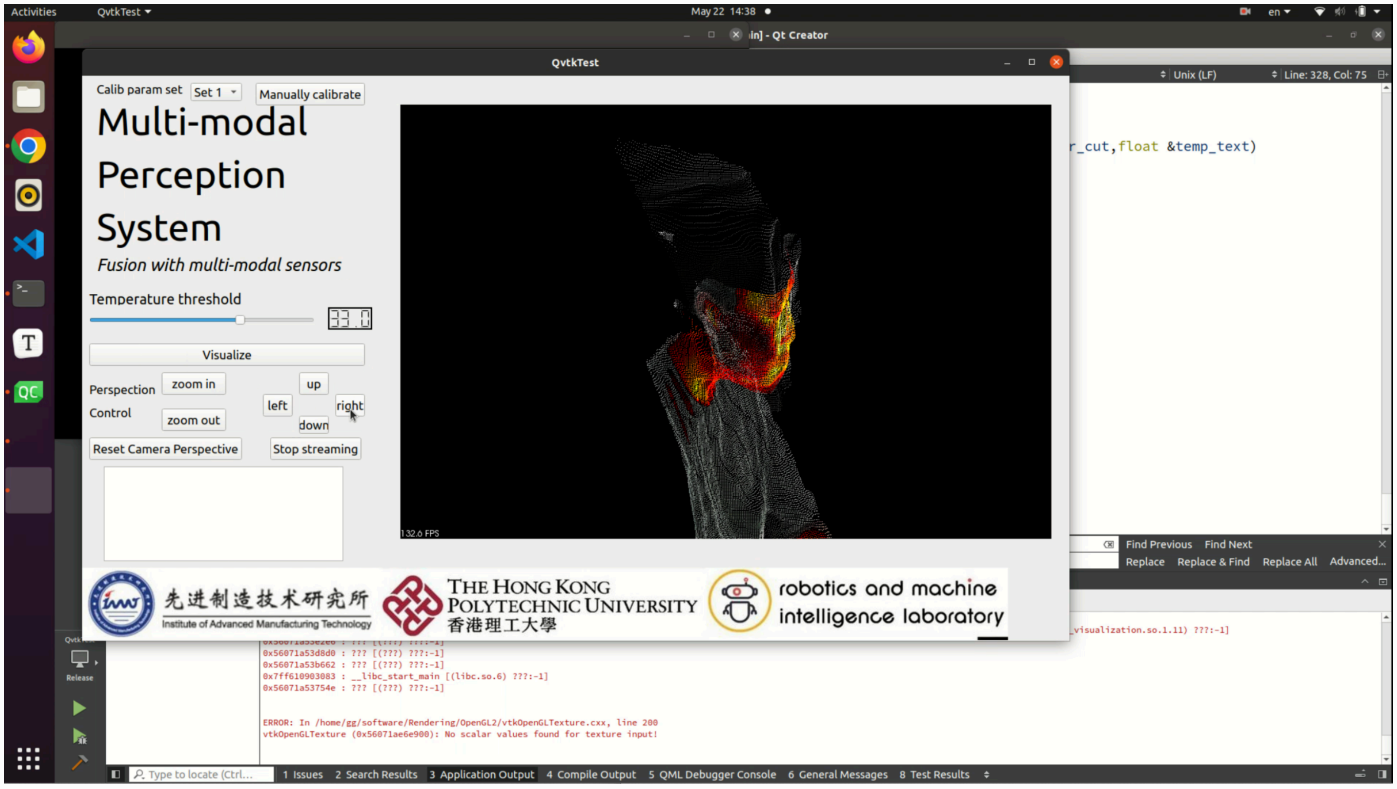


3. The perspection control panel controls the camera perspective to the visualizing point cloud. The initial perspection is set to in front of the `z` axis.

Press `zoom in` :

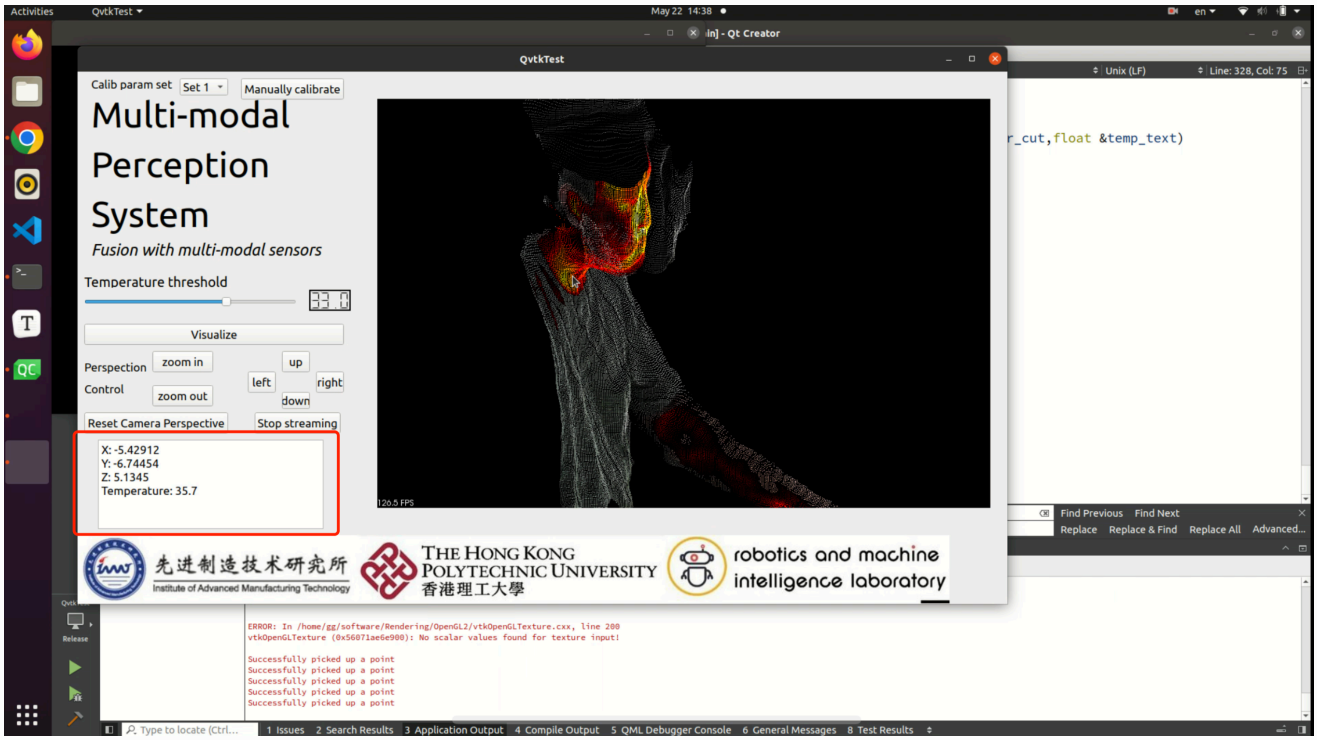


Press `right` :



Also provide `Reset Camera Perspective` for reset to the initial camera perspective

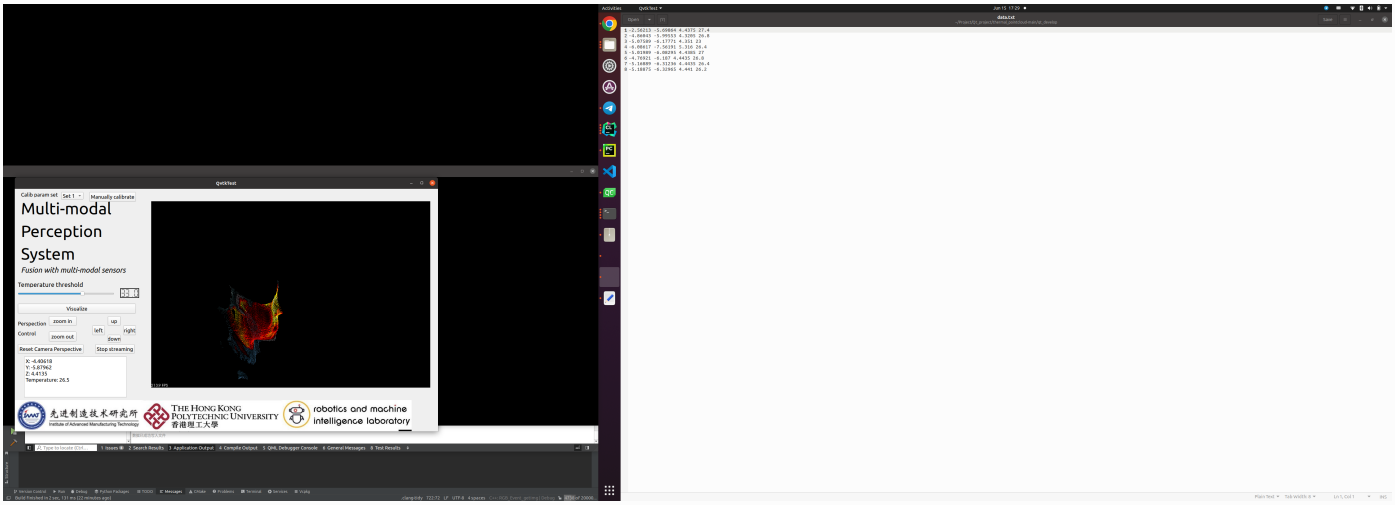
4. The message box interact with the `left mouse` and `shift` pressed simultaneously on the interested point in the right QvtkWidget, it will give the coordinate of the selecting point in the **RGB camera axis** and the celsius temperature of it. These parameters can be used as the input for the robot manipulation or other tasks.



Meanwhile, the thermal point cloud data `point` and `temperature` information will be saved into the file `data.txt`, because it need to use an absolute path, you should change it by yourself in the `QvtkTest.cpp`:

```
// write point data to a log file
std::ofstream outputFile("/home/emrys/Project/Qt_project/thermal_pointcloud-main/qt_develop/data.txt", std::ios::app);
```

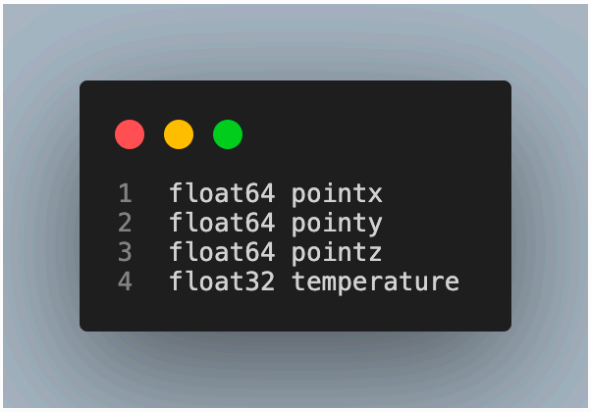
Then, the selected point data will be saved to the log file used for the next ROS node.



3.3 ROS Read TPCD Data

The data we just selected is saved into the `data.txt` file in the same folder. And use the ROS package we offer.

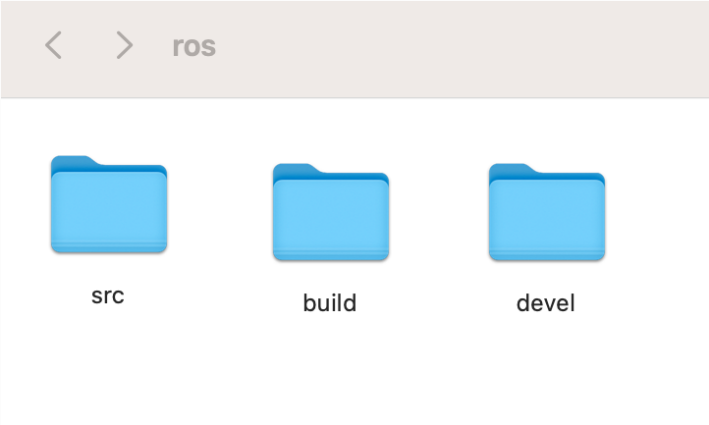
The ROS message `tpcd.msg` is shown as below:



Here is the instruction of how to use this package:

1. Move the package to your ROS workspace

The source code of the message publish node is in the `src` folder. Delete the `build` and `devel` folder if you need to recompile the package.



2. Modify the CMakeList file and source code

Modify your `CMakeLists.txt` file, in the my_ros_package package so that our ROS node can be compiled. Add the following content to the file:

```
find_package(catkin REQUIRED COMPONENTS
    roscpp
    my_ros_msgs
)

catkin_package()

include_directories(
    ${catkin_INCLUDE_DIRS}
)

add_executable(read_data src/read_data.cpp)
target_link_libraries(read_data ${catkin_LIBRARIES})
```

Here we use some common catkin commands to specify dependencies, include directories, and library files, and compile our ROS node read_data.

Also, change the path in the `ros/src/tpcd/src/read_data.cpp` to your own:

3. Compile the ROS package

```
cd ~/my_ros_ws
catkin_make
```

4. Run the ROS node

```
rosrun tpcd read_data
```

4 TROUBLE SHOOTING

1. When running the python scripts, the terminal outputs `Device permission denied`

This is because of the port permissions, type in the terminal:

```
# ttyACM0 is the error device in the info
sudo chmod 666 /dev/ttyACM0
```

2. Error: allocation of incomplete type 'Ui:xxxxxx' while compiling the Qt project

This is an error caused by the IDE, just select: build -> clean build will solved.

3. Error `QMetaObject::connectSlotsByName: No matching signal for pushButton2_clicked()` in Qt software compiling.

This is because the qt compiler doesn't update the `ui_QvtkTest.h` properly, navigate into this file and substitute `pushButton_2` to `pushButton_zoomout`