北京航空航天大學
BEIHANG UNIVERSITY

3D点云数据处理和可视化实践(入门)

Lecture1-三维点云计算基础

国新院实验实践课
工程师通用技术科教平台

教 师 ： 欧 阳 真 超

邮 箱 ： ouyangkid@buaa.edu.cn

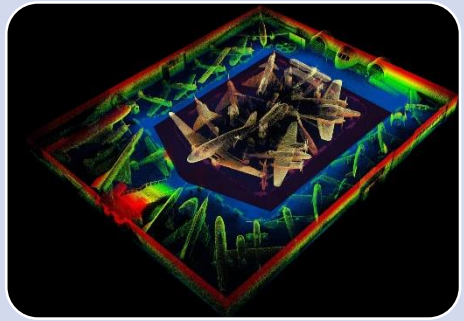
学 期 ： 2024年秋季

目录

Contents

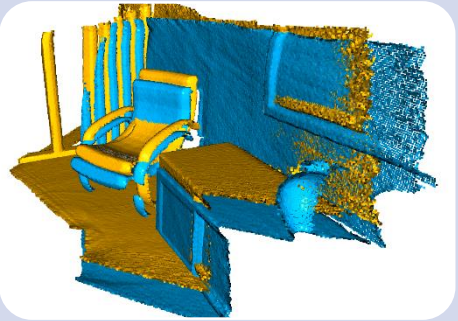
Part 1 | 课程内容安排

- 课程围绕**三维点云数据**的处理和可视化实践的**4个核心内容**开展，具体包括



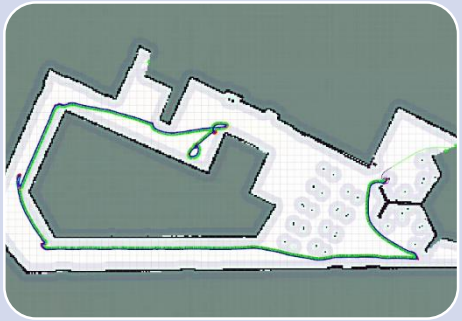
三维点云计算基础

（1+3学时）



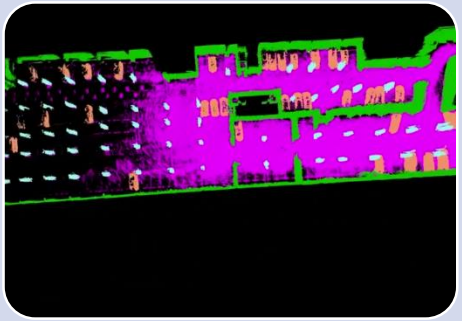
三维点云配准机制

（1+3学时）



同步定位与建图

（1+3学时）



点云语义分割模型

（1+3学时）

Part 1 | 课程内容安排

三维点云计算基础

理论教学-1课时

- 传感器原理
- 数据获取
- 数据结构
- 可视化
- Open3D API
- Realsense传感器

实践教学-3课时

- 开发环境安装
- 传感器数据获取
- 数据存储与播放
- 基础运算
- 可视化

三维点云配准机制

理论教学-1课时

- 配准机制
- 李群代数
- ICP配准流程
- CICP配准流程

实践教学-3课时

- 传感器数据采集
- ICP配准实践
- CICP配准实践
- 连续配准和可视化

同步定位与建图

理论教学-1课时

- 机械激光雷达
- 经典SLAM流程
- ROS
- SLAM评估机制

实践教学-3课时

- ROS基本命令操作
- Bag回放和数据读取
- 典型算法运行
- 可视化和测试

点云分割

理论教学-1课时

- 语义分割任务定义
- 深度学习典型框架
- 典型数据表征介绍
- 分割量化评估

实践教学-3课时

- 基于SLAM的点云语义标注
- 模型训练和测试
- 量化评估和可视化

目录

Contents

Part 2 | 课程目标

3D点云数据处理和可视化实践

目标学生

- 电子信息和交通运输

学习内容

- 多学科交叉的实践学习

先进传感器

激光雷达作为近年来普及的传感器被广泛应用于三维测量和机器人领域。

- 掌握传感器前沿技术
- 学习激光雷达特性和选型
- 理论与应用相结合

智能机器人

面向先进机器人移动感知建模，在机载和车载领域的理论与实践技术开展介绍。

- 机器人类人空间认知
- 智能感知前沿技术剖析
- 通过先进技术吸引学生科研兴趣

人工智能

深度学习在三维点云数据的应用技术、从“数据-建模-量化评估”的闭环流程开发。

- 平台和目标导向
- 感知系统设计
- 基本覆盖神经网络任务流程开发
- 基础算法和模型框架学习

编程实践

课程内容基于近年实验室研究的工程经验积累，前沿工具和算法实用性强。

- 软硬件环境操作学习
- 算法理论+编程实践
- 开发流程：多种工具的串联使用
- 可视化（定性）和量化评估（定量）

目录

Contents

Part 3 | 学习目标

3D点云数据处理和可视化实践入门

1.两类传感器（硬件）

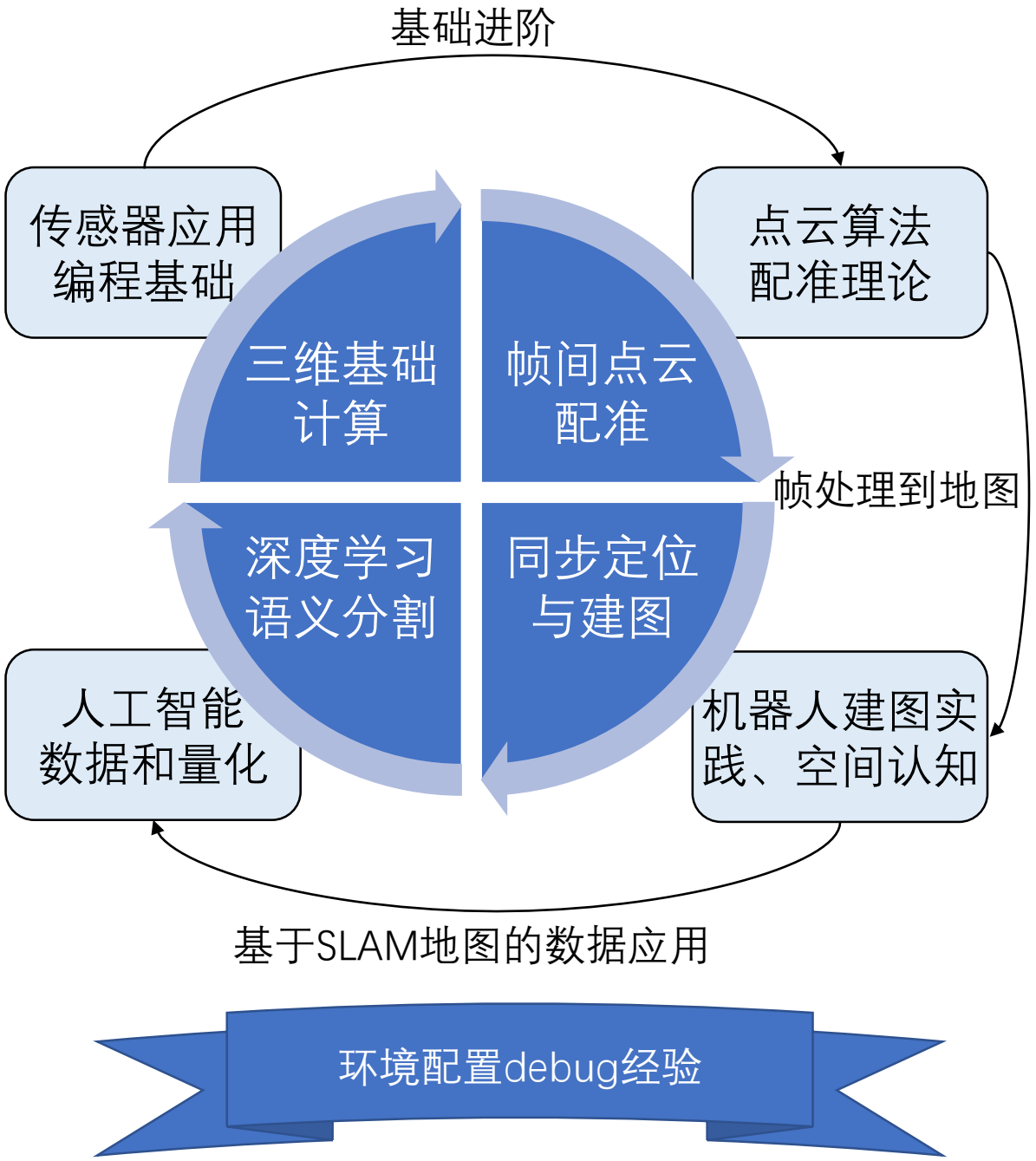
- Realsense：双目+结构光，稠密视场点云
- LiDAR：机械ToF测距，稀疏环视点云

2.三维空间表征与计算（理论编程）

- Open3D：先进算法库使用
- SLAM经典算法流程
- 深度学习：Pytorch+Spcov
- 李群代数，图优化

3.软件系统（操作实践）

- Ubuntu：shell
- ROS机器人操作系统
- PointLabeler



环境配置debug经验

目录

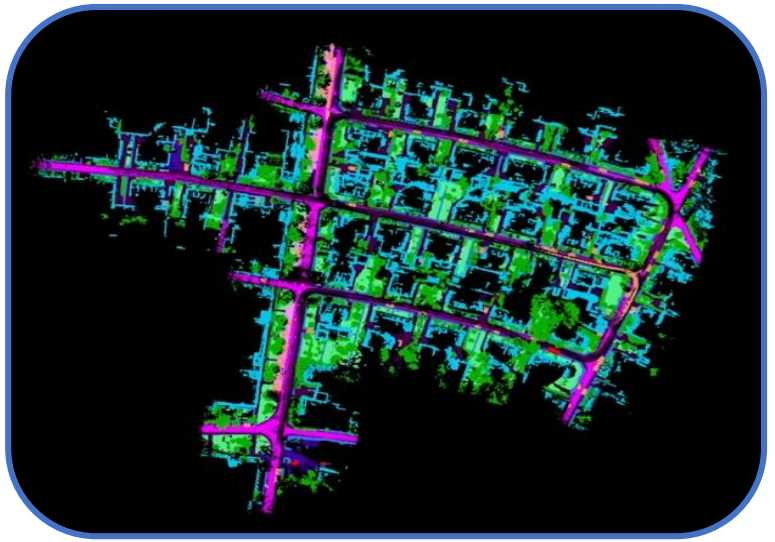
Contents

Part 4 | 硬件介绍

3D点云数据处理和可视化实践

传感器和移动机
器人平台

内网远程访问的
服务器和显卡



开源数据、本地
采集数据和软件

目录

Contents

Part 5 | Realsense传感器初探

传感器选型对比pdf

双目结构光相机

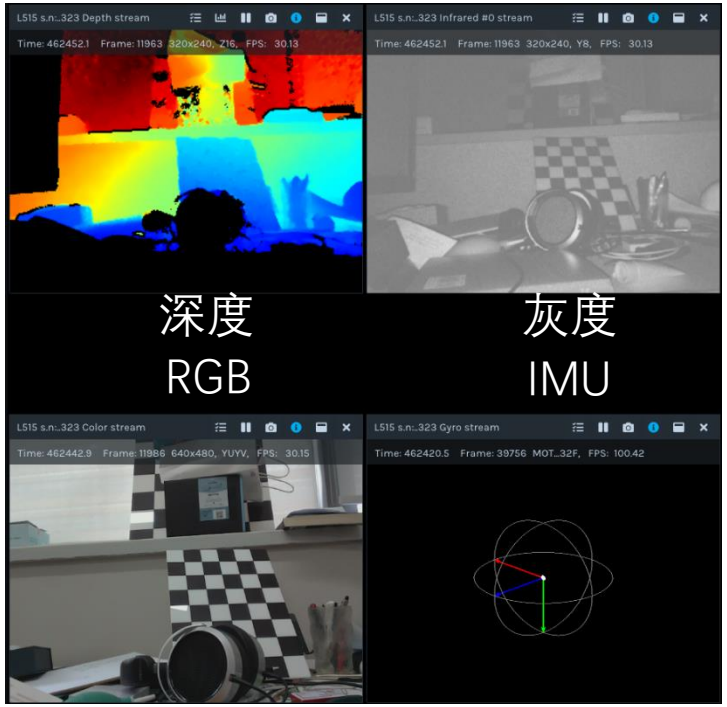
结构光测距相机

传感器构成:

- **数据接口：USB 3.0**
- **相机：单目/双目**
- **测距：结构光，激光ToF**
- **惯性单元：可选**

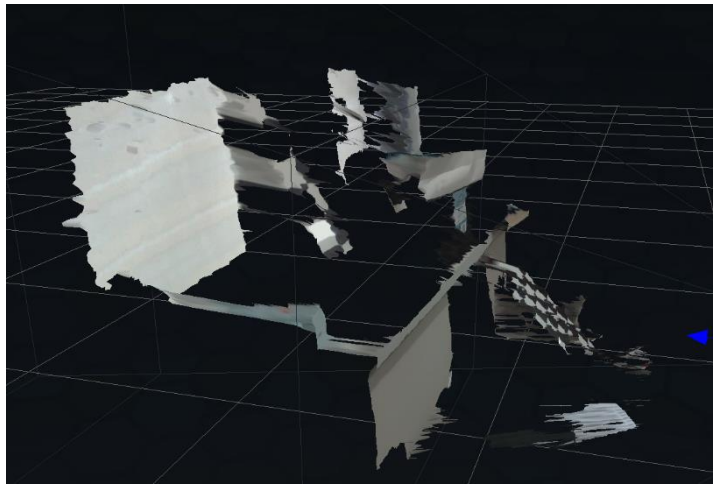
固态激光雷达

2D视角



细节放大

3D视角



Part 5 | Realsense传感器初探

传感器应用

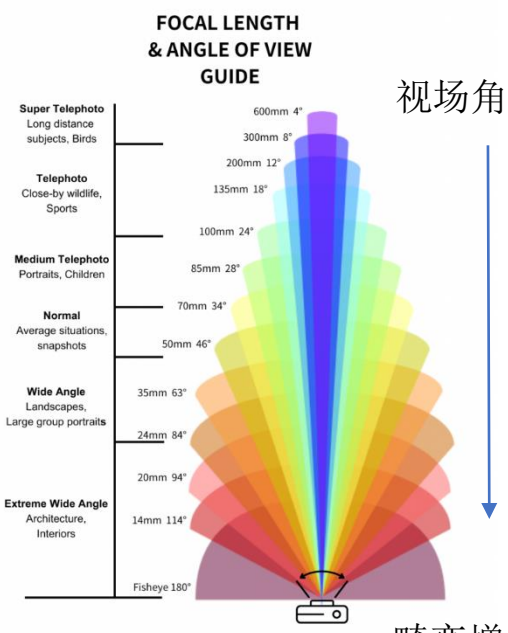
由于环境光噪声、传感器固有局限、使用过程引入的测量误差等，传感器环境采样难以达到理想结果，因此需要对采样数据进行计算、裁剪、降噪、修正等计算上的优化。

类比PS是对二维图像的优化处理，点云计算是未来三维处理的PS。

Part 5 | Realsense传感器初探

传感器关键参数对比

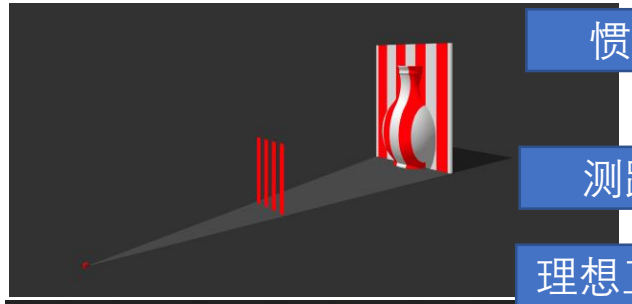
卷帘快门vs. 全局快门

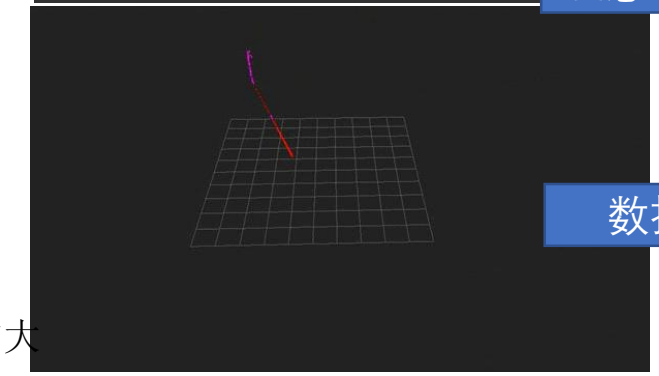


视场角

畸变增大

结构光 vs. ToF





快门模式

感知范围

测距精度

速度

惯性功能

测距极限

理想工作范围

数据接口

传感器介绍

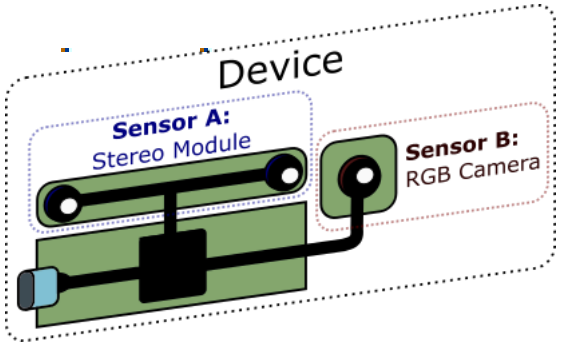
Feature	D457 / D456	D455/D455f	D435/D435i	D435f/D435if	D415	D405
						
Use Environment	Indoor & Outdoor	Indoor & Outdoor	Indoor & Outdoor	Indoor & Outdoor	Indoor & Outdoor	Indoor & Outdoor
Image Sensor Technology	Global Shutter	Global Shutter	Global Shutter	Global Shutter	Rolling Shutter	Global Shutter
Depth FOV (H × V) (1)	87° × 58°	87° × 58°	87° × 58°	87° × 58°	65° × 40°	84° × 58°
Depth Resolution	Up to 1280 × 720	Up to 1280 × 720	Up to 1280 × 720	Up to 1280 × 720	Up to 1280 × 720	Up to 1280 × 720
Depth Accuracy(2)	<2% at 4 m	<2% at 4 m	<2% at 2 m	<2% at 2 m	<2% at 2 m	<2% at 50 cm
Depth Frame Rate	Up to 90 fps	Up to 90 fps	Up to 90 fps	Up to 90 fps	Up to 90 fps	Up to 90 fps
Depth Filter	All-Pass	All-Pass/IR-Pass	All-Pass	IR-Pass	All-Pass	IR-Cut
RGB Sensor Technology	Global Shutter	Global Shutter	Rolling Shutter	Rolling Shutter	Rolling Shutter	Global Shutter(3)
RGB Resolution and Frame Rate	1280 × 800 at 30 fps	1280 × 800 at 30 fps	1920 × 1080 at 30 fps	1920 × 1080 at 30 fps	1920 × 1080 at 30 fps	1280 × 720 at 30 fps
RGB Sensor FOV (H × V)(1)	90° × 65°	90° × 65°	69° × 42°	69° × 42°	69° × 42°	87° × 58°
Inertial Measurement Unit	Yes	Yes	No/Yes	No/Yes	No	No
Minimum Depth Distance (Min-Z) at Max Resolution	~52 cm	~52 cm	~28 cm	~28 cm	~45 cm	~7 cm @ 480p
Ideal Range	.6 m to 6 m	.6 m to 6 m	.3 m to 3 m	.3 m to 3 m	.5 m to 3 m	7 cm to 50 cm
Main Components	Intel® RealSense™ Vision Processor D4, Intel® RealSense™ depth module D450	Intel® RealSense™ Vision Processor D4, Intel® RealSense™ depth module D450	Intel® RealSense™ Vision Processor D4, Intel® RealSense™ depth module D430	Intel® RealSense™ Vision Processor D4, Intel® RealSense™ depth module D430	Intel® RealSense™ Vision Processor D4, Intel® RealSense™ depth module D415	Intel® RealSense™ Vision Processor D4, Intel® RealSense™ depth module D401
Dimensions (L × H × D)	124 mm × 29 mm × 36 mm / 124mm x 29mm x 26mm	124 mm × 29 mm × 26 mm / 124 mm × 29 mm × 26.8 mm	90 mm × 25 mm × 25 mm	90 mm × 25 mm × 25.8 mm	99 mm × 23 mm × 20 mm	42 mm × 42 mm × 23 mm
Interface	GMSL FAKRA / USB 3	USB 3	USB 3	USB 3	USB 3	USB 3

Part 5 | Realsense传感器初探

传感器介绍

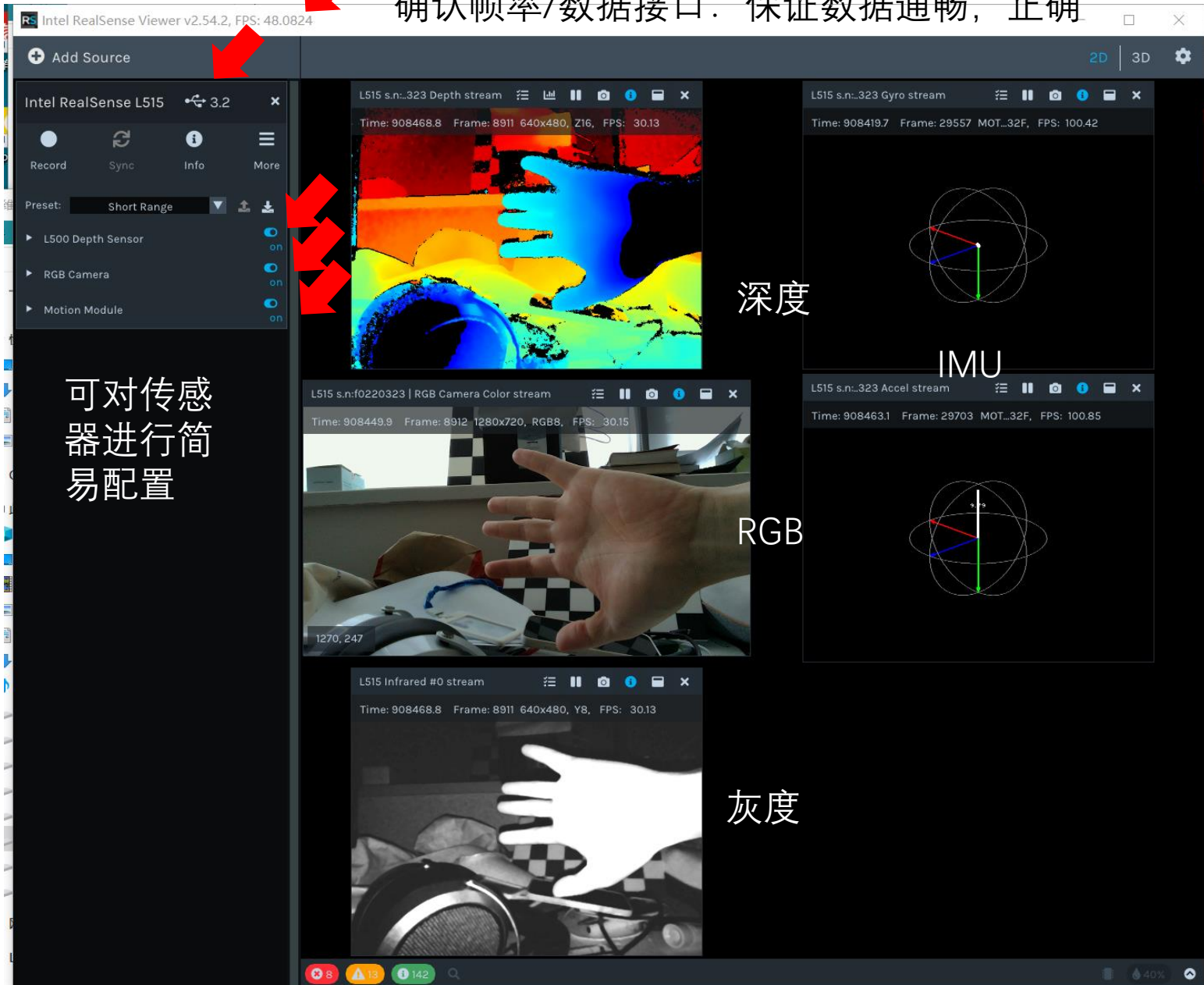
传感器数据获取

Intel.RealSense.Viewer.exe



- 连接传感器到电脑--usb
- 使用Intel.RealSense.Viewer 确认传感器功能。# 不要使用最新版测试L515（已停产）
- 深度信息纹理较弱，只有反射率差异体现
- RGB纹理丰富，但缺少距离置信度
- 黑白为单通道RGB
- IMU用于辅助姿态信息获取，可支持Visual-Inertial的数据融合。
- 三类sensor出厂完成了基础预标定（外参）。

确认帧率/数据接口：保证数据通畅，正确



可对传感器进行简易配置

深度

IMU

RGB

灰度

开发环境配置

- IDE： pycharm

https://www.jetbrains.com/pycharm/download/?section=window 选择 **PyCharm Community Edition**

- Python虚拟环境管理：anaconda

https://www.anaconda.com/download/success

- 源替换(国内加速):
  1. https://mirror.tuna.tsinghua.edu.cn/help/anaconda/ conda清华源
     1. conda config --set show_channel_urls yes # 生成 .condarc
  2. https://mirror.tuna.tsinghua.edu.cn/help/pypi/ pip源

Realsense github:

https://github.com/IntelRealSense/librealsense/tree/master

Part 5 | Realsense传感器初探

开发环境配置

绝大多数linux和开发环境自带操作说明：-h/-help查询

- conda虚拟环境创建 （conda常用命令，或者 conda -h）

conda create –n open3d python=3.8 # 创建一个名为 open3d的python3.8环境

- 每次通过 activate/deactivate 进入/退出相应的环境

conda activate open3d
pip install open3d, opencv-python, pyrealsense2 # 当前时间最新版本0.18.0
numpy==1.24.0 # 2.X版本有memory的bug
安装完成后 pip list 可以看到安装完成的相应包，并且可以在程序中进行import操作。

```
nptyping                1.4.4
numpy                   1.24.3
open3d                  0.18.0
opencv-contrib-python   4.9.0.80
opencv-python           4.9.0.80
```

```python
import json
import os

import numpy as np
import cv2
import time

import open3d as o3d
```

验证基础环境配置完成

Untill 2024/8/20 tested!

Supported Python versions:

- 3.8
- 3.9
- 3.10
- 3.11

Supported operating systems:

- Ubuntu 18.04+
- macOS 10.15+
- Windows 10+ (64-bit)

```
PS H:\中法\2024\研究生课程\三维点云处理和可视化实践\code\lesson1> python -c "import open3d as o3d; print(o3d.__version__)"
0.18.0
```

Part 5 | Realsense传感器初探

深度数据获取-pyrealsense

https://github.com/IntelRealSense/librealsense/blob/master/doc/d435i.md

深度信息流获取，及ASCII码打印简易可视化。

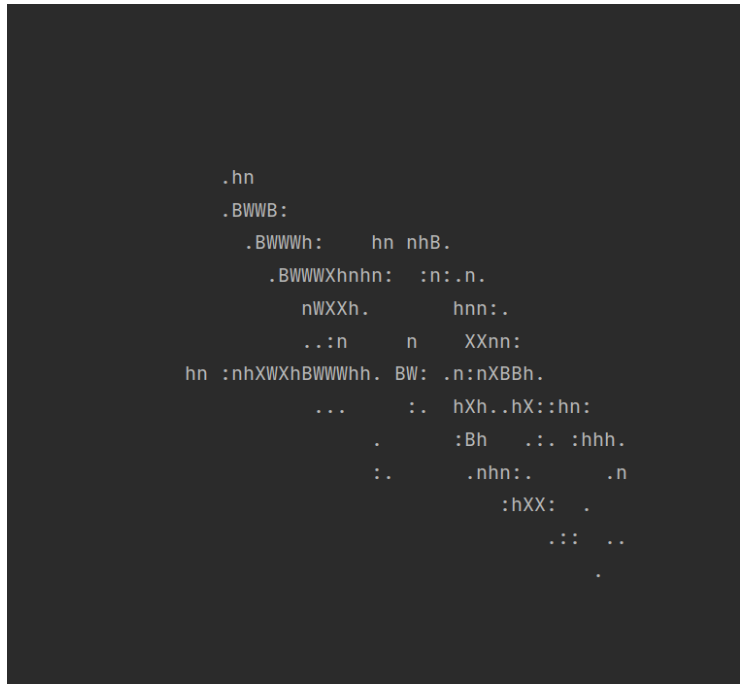
① python-depth.py

```python
import pyrealsense2 as rs

try:
    pipeline = rs.pipeline() # Create a context object.
    config = rs.config() # Configure streams
    config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
    pipeline.start(config) # Start streaming
    while True:
        frames = pipeline.wait_for_frames()
        depth = frames.get_depth_frame()
        if not depth:
            continue
        exit(0)

except Exception as e:
    print(e)
    pass
```

打印

```python
coverage = [0] * 64  # 列分成10块，每块64像素，累加统计
str_img = []
for y in range(480):  # size = 480*640
    for x in range(640):
        dist = depth.get_distance(x, y)  # get depth
        if 0 < dist < 1:  # filtering background >1 太远的
            coverage[x//10] += 1  # index 注意修改，累加

    if y % 20 == 19:  # 每行分成20块，每块24像素，基于coverage统计，进行可视化
        line = ""
        for c in coverage:
            line += " .:!@#$%…&"[c // 25]  # 按c(最大为10)的密度映射
        str_img.append(line)
        coverage = [0] * 64  # init
        print(line[::-1])  # 逆序

for l in str_img:
    print(''.join(l))
```

将手放在相机垂直正前方，摆出一定手势，比如victory，相应区域会出现随机字符串，如下图。



传感器数据获取

rs.pipeline()

传感器对象，用于处理所有连接到本机的realsense传感器。

rs.config()

数据流配置对象

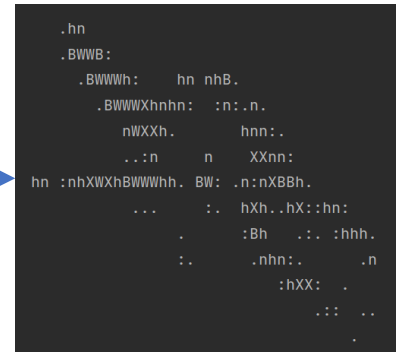
enable_stream()

数据流使能（重载），并配置对应参数，包括深度信息、RGB、IMU。

pipeline.start(config)

启动流

深度信息 → 距离阈值过滤 → 栅格化统计 → ASC码映射



frames = pipeline.wait_for_frames()

获取帧

depth = frames.get_depth_frame()

获取深度信息

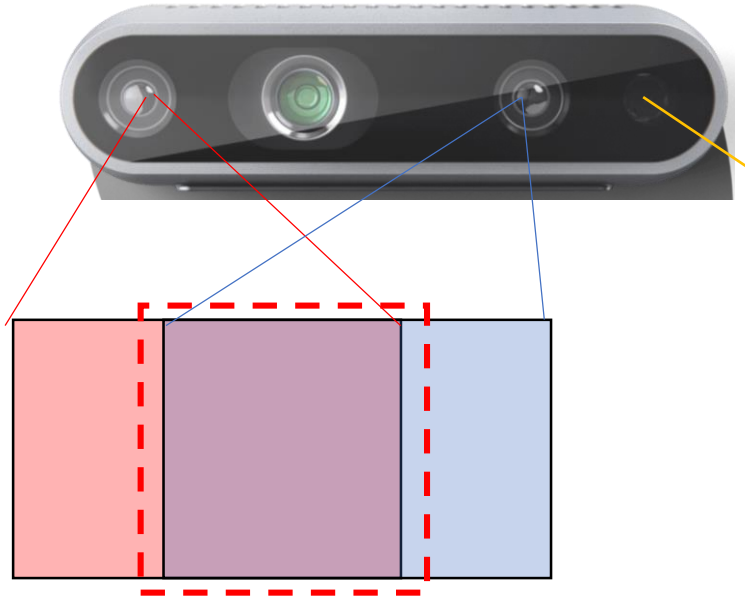
计算操作

exit(0)

停止

传感器数据获取-pyrealsense-深度获取可视化

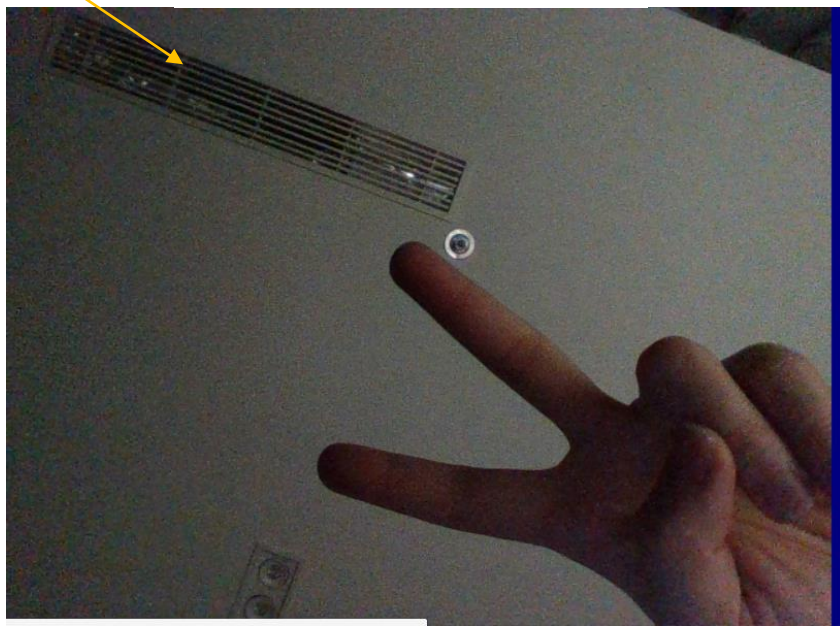
② opencv_viewer_example.py



```
frames = pipeline.wait_for_frames()
depth_frame = frames.get_depth_frame()  # get depth
color_frame = frames.get_color_frame()  # get rgb
```

RGB sensor FOV (H × V):
69° × 42°

Depth Field of View (FOV):
87° × 58°



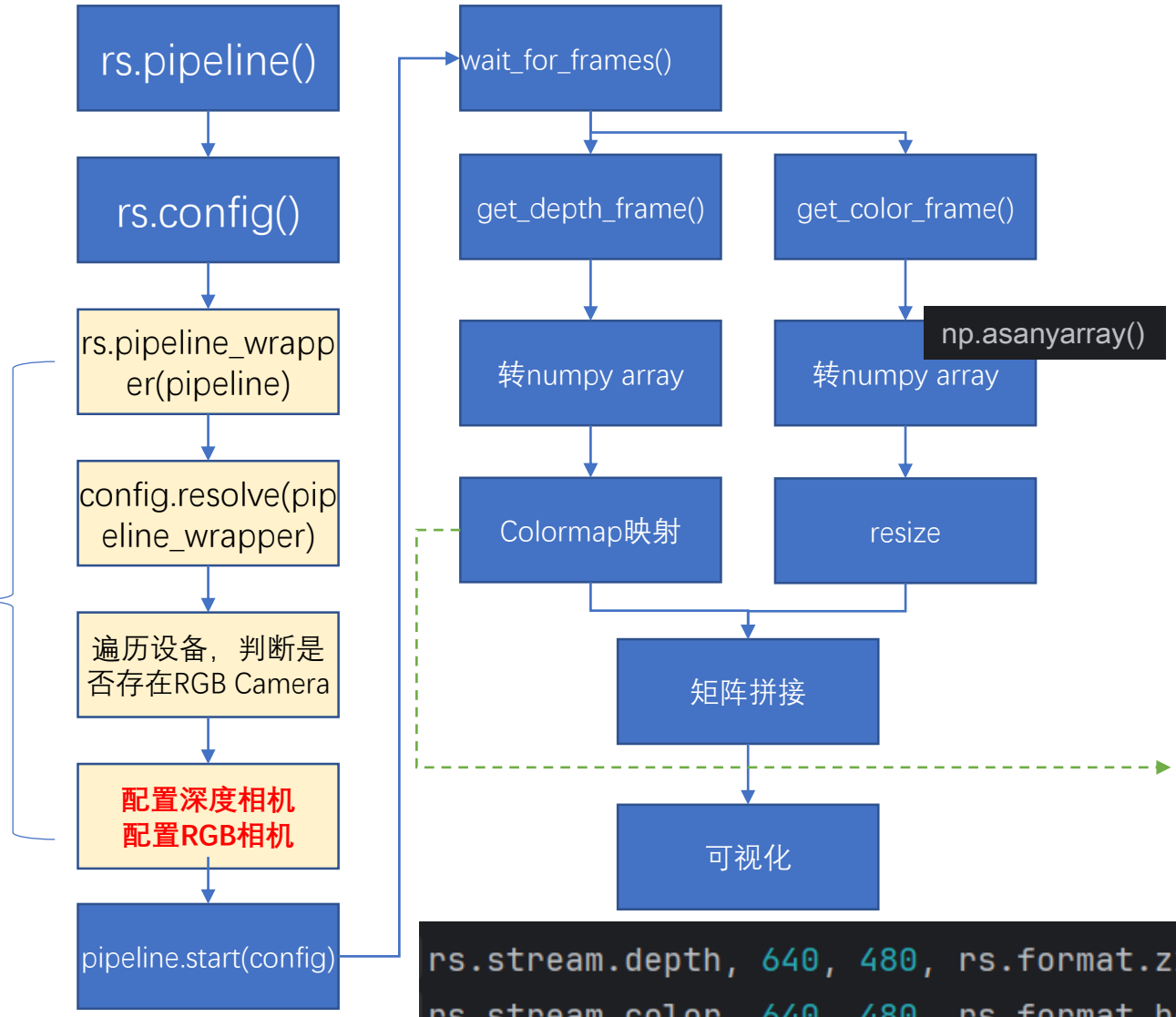
- 由于装配位置（外参），双目标定依赖共享视场，所以理论深度视场更小。
- 加上需要与RGB相机对齐，会进一步crop掉外圈视场，以匹配相机。

结果：深度图分辨率和视场更小。

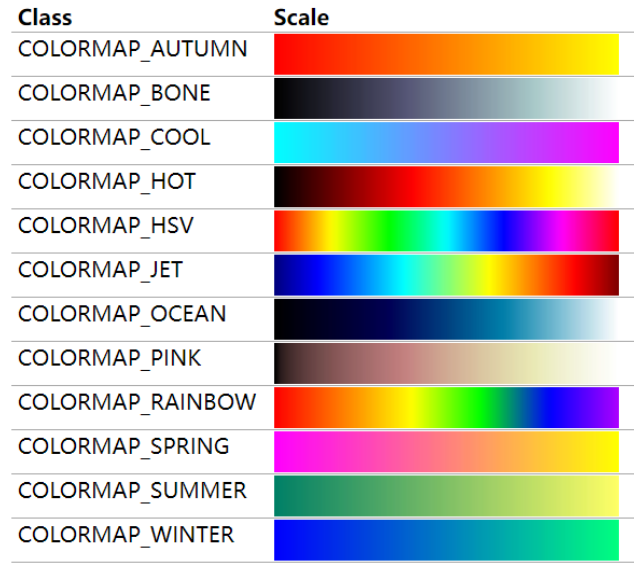
Part 5 | Realsense传感器初探

传感器数据获取

1. 判断传感器是否存在
2. 配置传感器数据参数
3. 要获取多个传感器数据




```
rs.stream.depth, 640, 480, rs.format.z16, 30)
rs.stream.color, 640, 480, rs.format.bgr8, 30)
```

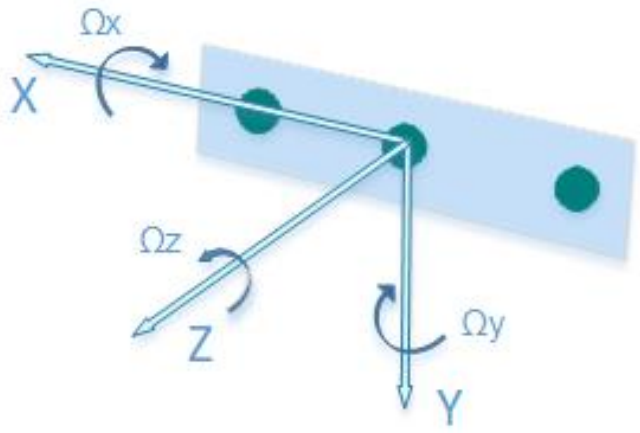


灰度图实际为单通道灰度，z16对应无符号整型，精度16，2^16-1=65535，映射压缩到0-255，并转换色彩空间：cv2.applyColorMap()

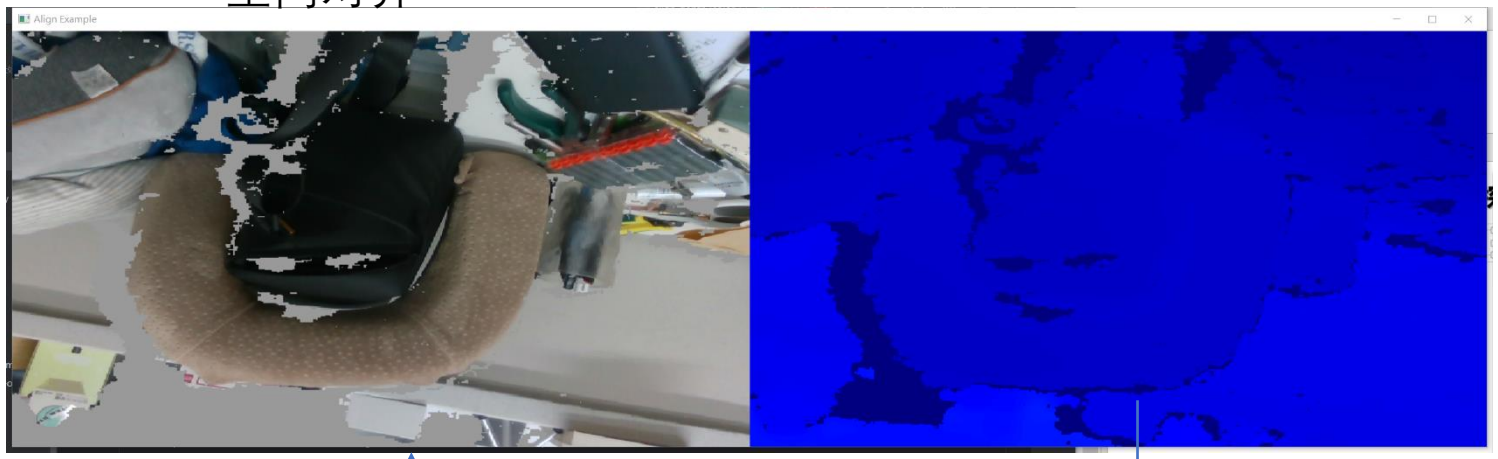
传感器数据获取-pyrealsense-数据对齐

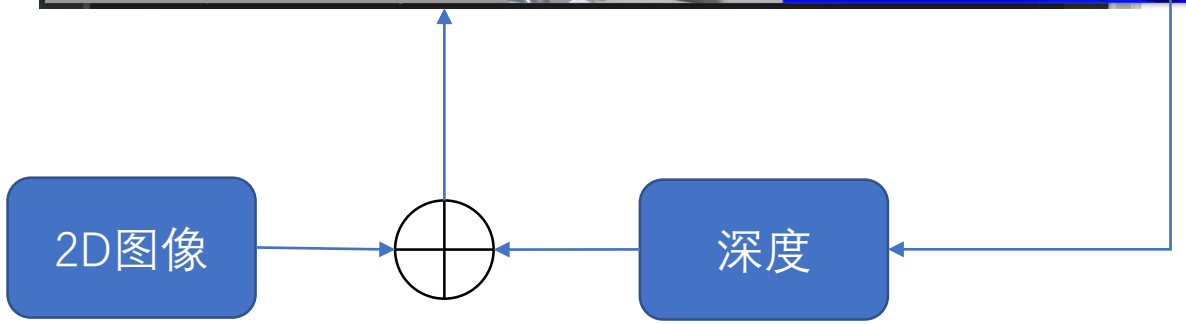
③ align-depth2color.py

A separate 3D coordinate space, specified in meters, with the coordinate [0,0,0] referring to the center of the physical imager.
Within this space, the positive x-axis points to the right, the positive y-axis points down, and the positive z-axis points forward.



空间对齐





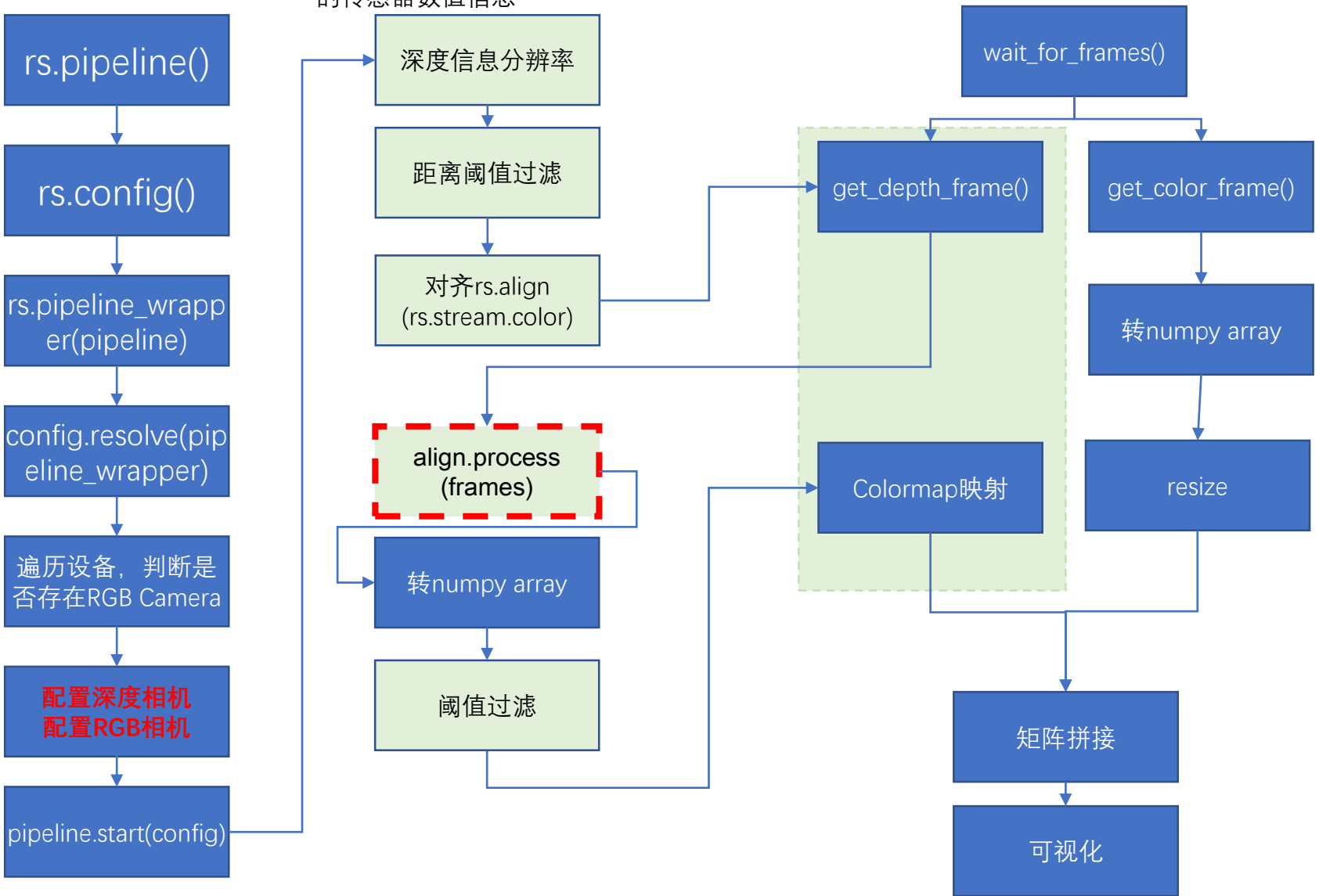
生成一个新帧，其大小与彩色流相同，但内容是彩色传感器坐标系中计算的深度数据。

Part 5 | Realsense传感器初探

传感器数据获取

Depth Scale is: 0.001 # 毫米，使用阈值距离(米)/scale= 整型的传感器数值信息

★实验：通过bar调整距离阈值的过滤。



配置深度相机
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)

rs.format.z16

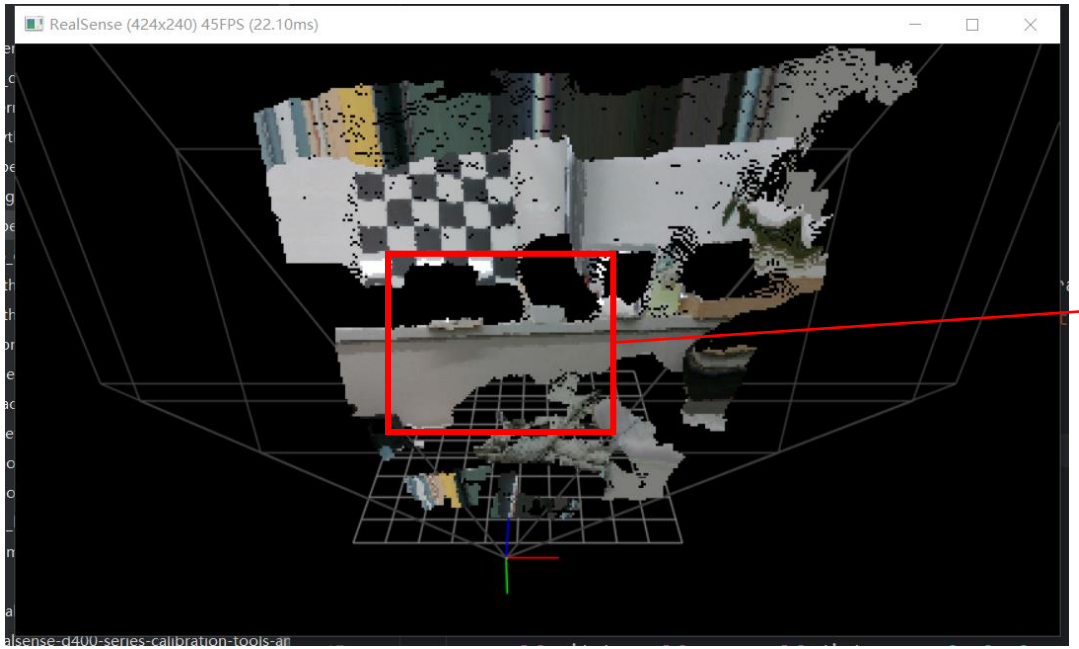
RS2_FORMAT_Z16 , /**< 16-bit linear depth values. The depth is meters is equal to depth scale * pixel value. */

Part 5 | Realsense传感器初探

深度数据获取-pyrealsense-pointcloud

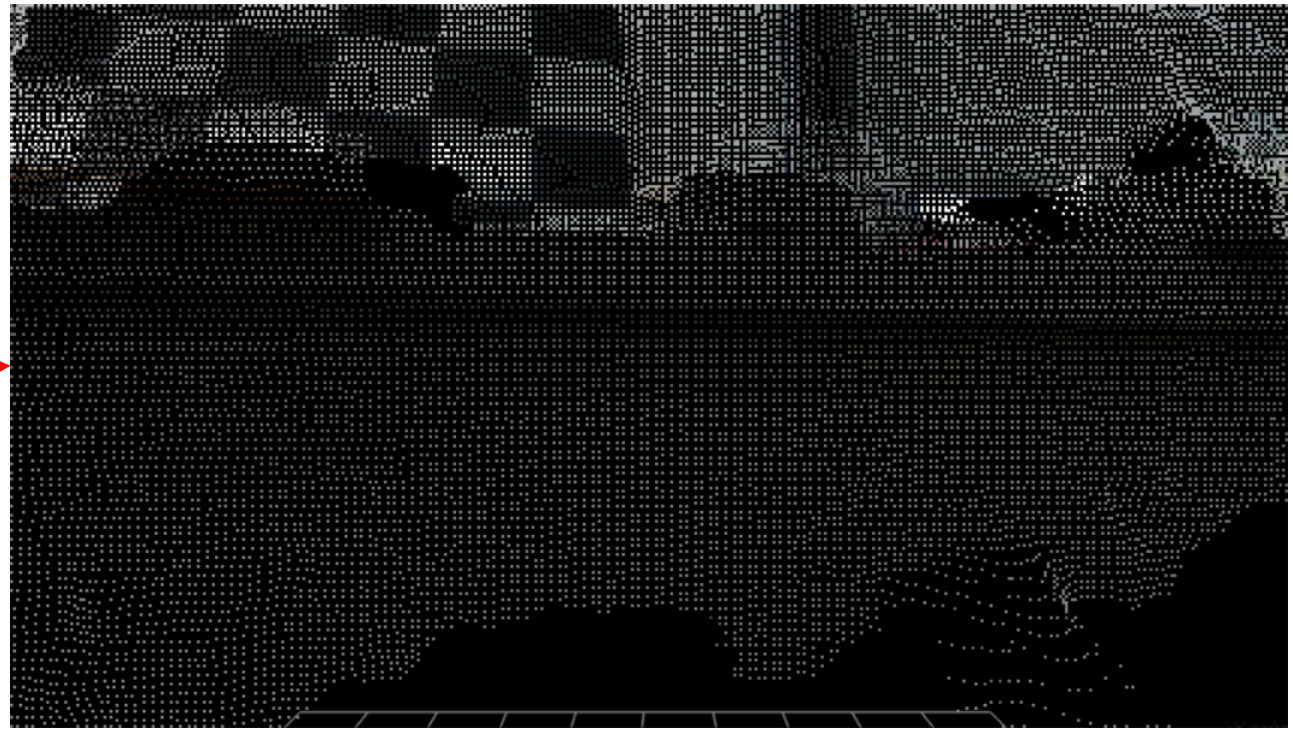
④ opencv_pointcloud_viewer.py

三维视角：
1. 点击鼠标左键， 上下左右旋转相机投影视角
2. 点击鼠标右键，上下左右平移相机投影视角
3. 鼠标滚轮/中间， 放缩/拉近视场



视窗由2D转为3D

快捷键：s-当前视角存储为图片
e-当前帧存储为ply
c- 切换着色，colormap映射/rgb



深度数据获取-pyrealsense-pointcloud

初始化UI窗口参数

rs.pipeline()

rs.config()

rs.pipeline_wrapper(pipeline)

config.resolve(pipeline_wrapper)

默认分辨率

配置深度相机
配置RGB相机

Pipeline.Start()

获取深度相机内参

配置过滤器和着色

配置opencv窗口autosize

交互可视化
While 循环

获取数据帧

按键监听

鼠标事件

投影相关函数定义

wait_for_frames()

get_depth_frame()

get_color_frame()

过滤

获取不同的着色方案

更新深度相机内参

生成Point

计算投影坐标和相机角度

生成Pointcloud

用于创建点云和相应的深度-颜色帧的纹理映射。从深度图像中创建的点云是深度流的3D坐标系统中的一组点。

```
points = pc.calculate(depth_frame)
pc.map_to(mapped_frame)
```

★**实验**：将数据转化为open3d的pcd格式，并存储（结合open3d数据结构）进行。

深度数据获取-pyrealsense-pointcloud

两种图像投影方式对比：

1. Pointcloud：基于深度信息和图像构建纹理映射，获得三维坐标系下的深度流坐标。

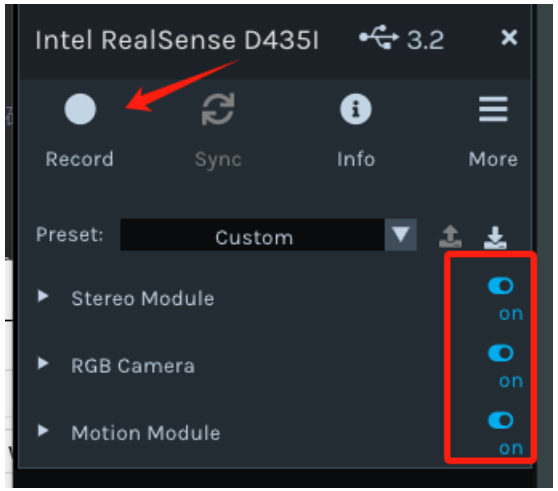
```
pc = rs.pointcloud()
points = pc.calculate(depth_frame)
pc.map_to(color_frame)
```

2. Frame alignment：图像和深度流对齐。

```
align = rs.align(rs.stream.color)
aligned_frames = align.proccess(depth_and_color_frameset)
color_frame = aligned_frames.first(rs.stream.color)
aligned_depth_frame = aligned_frames.get_depth_frame()
```

深度数据获取-pyrealsense-存bag

1. Bag录制：



1. 选择存放**路径**不包含中文，可能是码制的问题。
2. 录制时间不要太长（5s左右即可），原始数据的量很大。

2. 程序存Bag ⑤ save_bag_example.py

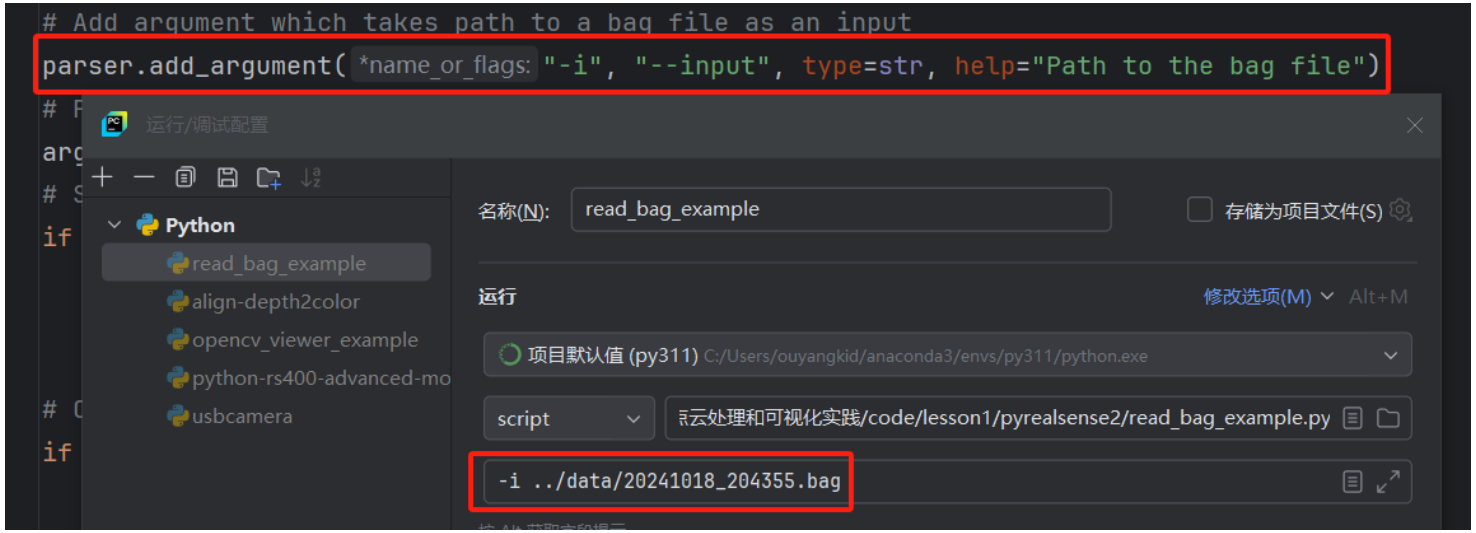
```
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
rs.config.enable_record_to_file(config, args.input)   # enable record
pipeline.start(config)
try:
    print('recoding ... press ctrl+c to stop.')  # 释放
    while True:
        # Get frameset of depth
        frames = pipeline.wait_for_frames()

except KeyboardInterrupt:
    print("Stop recoding.")

finally:
    pipeline.stop()
```

读取时替换为

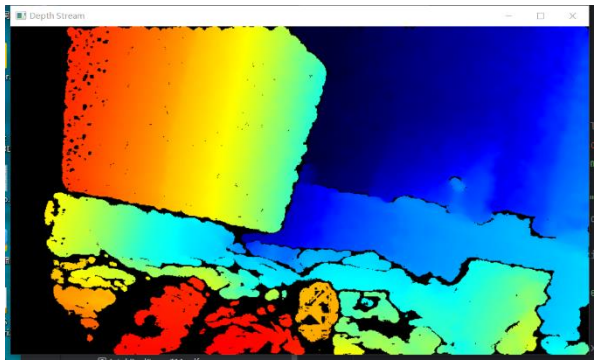
```
rs.config.enable_device_from_file(config, args.input)
```

深度数据获取-pyrealsense-读取bag

1. Bag播放　⑤ read_bag_example.py



2. 配置bag路径，播放录制结果。



深度数据获取-pyrealsense-读取bag

读取并判断参数和文件正确

↓

rs.pipeline()

↓

rs.config()

↓

enable_device_from_file()

↓

enable_stream()
需要配置相同参数

↓

pipeline.start(config)

↓

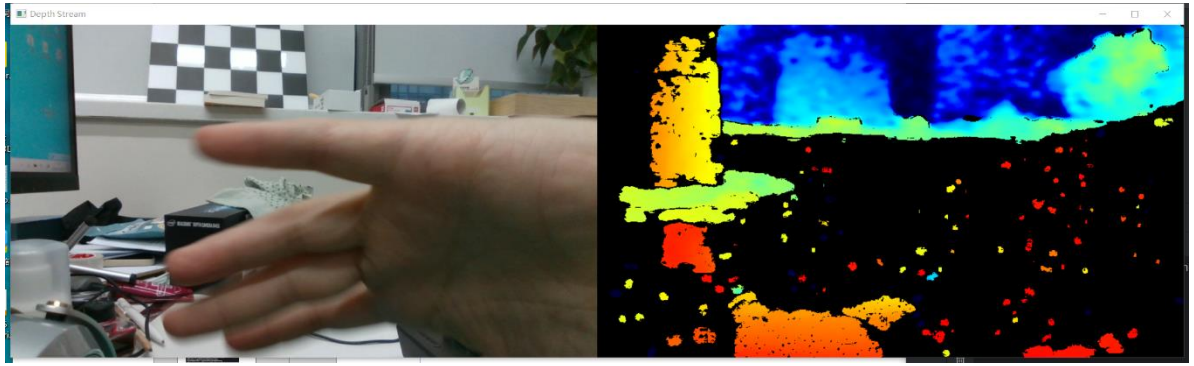
基于opencv的深度图播放

Argparse，参数获取和文件判断

配置数据源来自文件，而非传感器。

注意这里只播放了深度数据，

★**实验**：自己采集数据5s，完成对深度和图像的同时播放



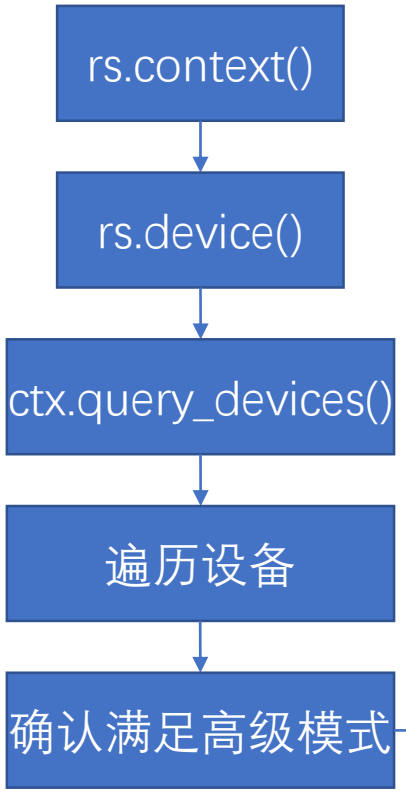
⑥：不同过滤器效果可视化测试
post_processing_filter.py

Part 5 | Realsense传感器初探

传感器数据获取-pyrealsense-高级模式

⑦ python-rs400-advanced-mode-example.py

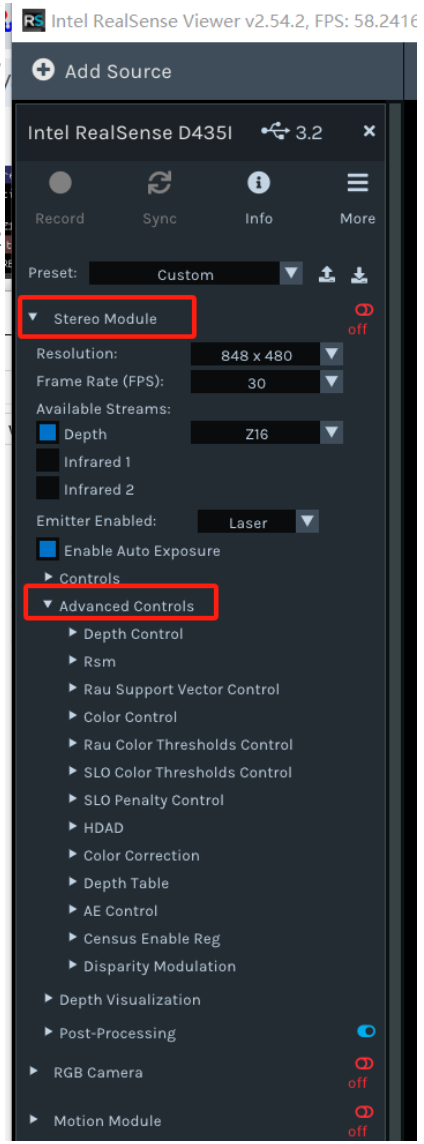
提供对相机控制参数的介绍示例。



依次打印高级模式及参数

```
# Get each control's current value
print("Depth Control: \n", advnc_mode.get_depth_control())
print("RSM: \n", advnc_mode.get_rsm())
print("RAU Support Vector Control: \n",
advnc_mode.get_rau_support_vector_control())
print("Color Control: \n", advnc_mode.get_color_control())
print("RAU Thresholds Control: \n",
advnc_mode.get_rau_thresholds_control())
print("SLO Color Thresholds Control: \n",
advnc_mode.get_slo_color_thresholds_control())
print("SLO Penalty Control: \n",
advnc_mode.get_slo_penalty_control())
print("HDAD: \n", advnc_mode.get_hdad())
print("Color Correction: \n", advnc_mode.get_color_correction())
print("Depth Table: \n", advnc_mode.get_depth_table())
print("Auto Exposure Control: \n", advnc_mode.get_ae_control())
print("Census: \n", advnc_mode.get_census())
```

主要是开发流程上缺少维护良好的API手册，开发过程中，参数配置和调参缺少相应的参考。建议配合可视化确认参数功能和效果。



手册： https://intelrealsense.github.io/librealsense/python_docs/index.html

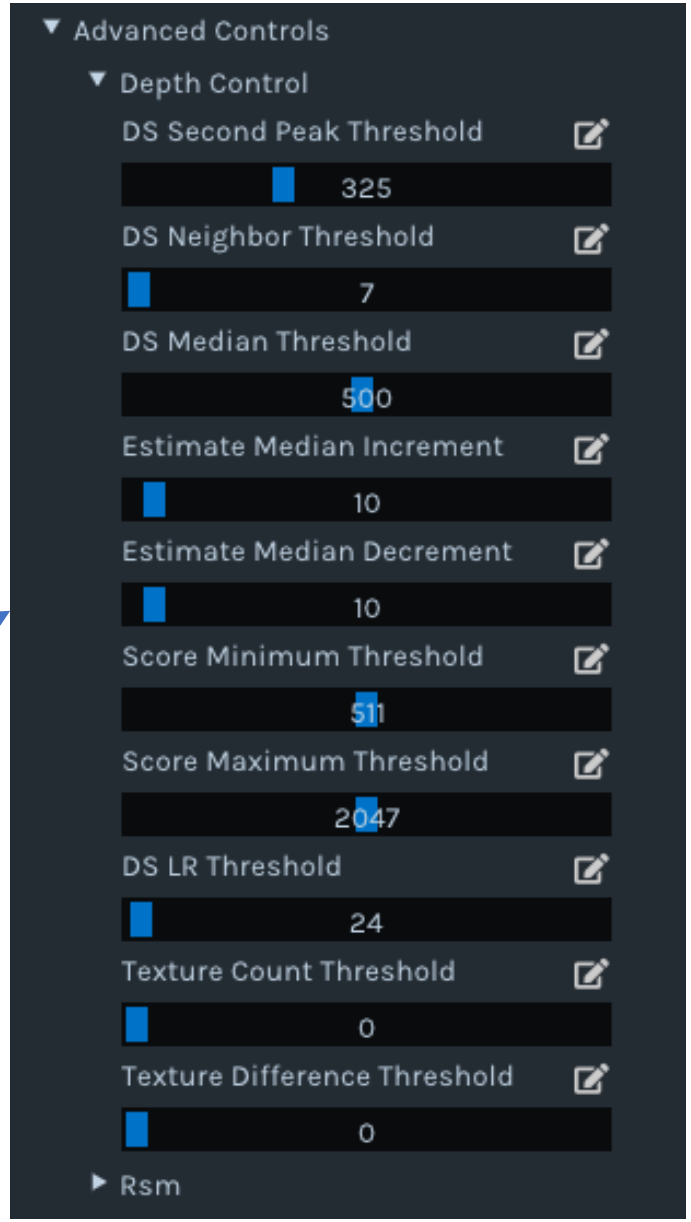
Part 5 | Realsense传感器初探

传感器数据获取-pyrealsense-高级模式

```
C:\Users\ouyangkid\anaconda3\envs\py311\python.exe H:\中法\2024\研究生
Found device that supports advanced mode: Intel RealSense D435I
Advanced mode is enabled
Depth Control:
 minusDecrement: 10, deepSeaMedianThreshold: 500, scoreThreshA: 1, s
RSM:
```

Depth Control:
minusDecrement: 10, **deepSeaMedianThreshold**: 500,
scoreThreshA: 1, **scoreThreshB**: 2047,
textureDifferenceThreshold: 0, **textureCountThreshold**: 0,
deepSeaSecondPeakThreshold: 325,
deepSeaNeighborThreshold: 7, **lrAgreeThreshold**: 24

一一对应
名称有所不同



在对程序进行调参，比如传感器参数和相关算法使能，通过此方法（Adv模式和UI程序）可以找到匹配的功能参数。

深度数据获取-pyrealsense-总结

- Realsense官方给出了一些基本的相机数据流的函数封装；
- 提供了相机参数调节和配置；
- 但是由于开发API缺少相应的介绍， 在Python开发时，缺少对点云和图像的高阶计算函数封装。
- 开发建议：pyrealsense2仅用于初步的传感器数据获取，结合opencv进行图像处理，结合open3d进行点云处理，以及可视化。

Part 5 | Realsense传感器初探

传感器数据获取-open3D

① device_read_bag.py

- 传感器参数获取

```
def camera_config():
    o3d.t.io.RealSenseSensor.list_devices()
```

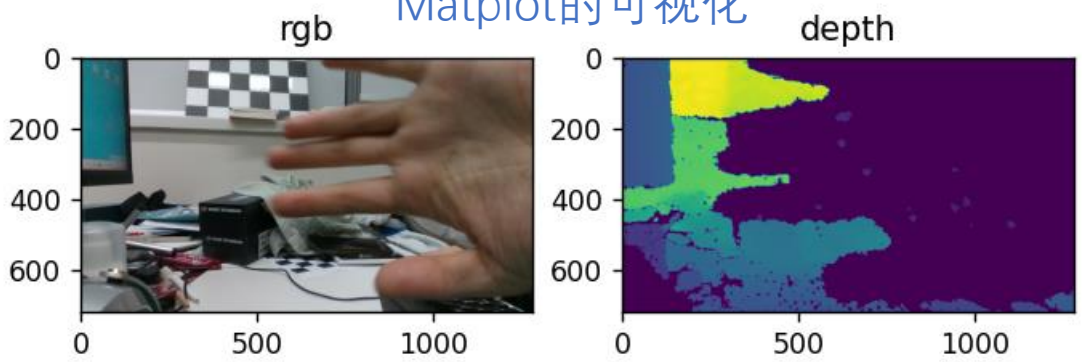
```
C:\Users\ouyangkid\anaconda3\envs\py311\python.exe H:\中法\2024\研究生课程\三维点云处理和可视化实践\code\lesson1\open3d\open3d_example.py
[Open3D INFO] [0] Intel RealSense D435I: 239122073486
[Open3D INFO]   color_format: [RS2_FORMAT_BGR8 | RS2_FORMAT_BGRA8 | RS2_FORMAT_RAW16 | RS2_FORMAT_RGB8 | RS2_FORMAT_RGBA8 | RS2_FOR
[Open3D INFO]   color_resolution: [1280,720 | 1920,1080 | 320,180 | 320,240 | 424,240 | 640,360 | 640,480 | 848,480 | 960,540]
[Open3D INFO]   depth_format: [RS2_FORMAT_Z16]
[Open3D INFO]   color_fps: [15 | 30 | 6 | 60]
[Open3D INFO]   depth_resolution: [1280,720 | 256,144 | 424,240 | 480,270 | 640,360 | 640,480 | 848,100 | 848,480]
[Open3D INFO]   depth_fps: [100 | 15 | 30 | 300 | 6 | 60 | 90]
[Open3D INFO]   visual_preset: []
[Open3D INFO] [1] Intel RealSense L515: f0220323
[Open3D INFO]   color_format: [RS2_FORMAT_BGR8 | RS2_FORMAT_BGRA8 | RS2_FORMAT_RGB8 | RS2_FORMAT_RGBA8 | RS2_FORMAT_Y16 | RS2_FORMA
[Open3D INFO]   color_resolution: [640,360 | 640,480]
[Open3D INFO]   color_fps: [15 | 30 | 6]
[Open3D INFO]   depth_format: [RS2_FORMAT_Z16]
[Open3D INFO]   depth_resolution: [320,240]
[Open3D INFO]   depth_fps: [30]
[Open3D INFO]   visual_preset: [RS2_L500_VISUAL_PRESET_CUSTOM | RS2_L500_VISUAL_PRESET_DEFAULT | RS2_L500_VISUAL_PRESET_LOW_AMBIENT
[Open3D INFO] Open3D only supports synchronized color and depth capture (color_fps = depth_fps).

进程已结束，退出代码为 0
```

Bag解析：

```
bag_reader = o3d.t.io.RSBagReader()
bag_reader.open(path)
count = 0
while not bag_reader.is_eof():
    print(count)
    im_rgbd = bag_reader.next_frame()
    if not im_rgbd.color or not im_rgbd.depth:
        continue

    rgb_im = np.asarray(im_rgbd.color)
    depth_im = np.asarray(im_rgbd.depth)

    count += 1
bag_reader.close()
```

计算操作

Matplot的可视化



Part 5 | Realsense传感器初探

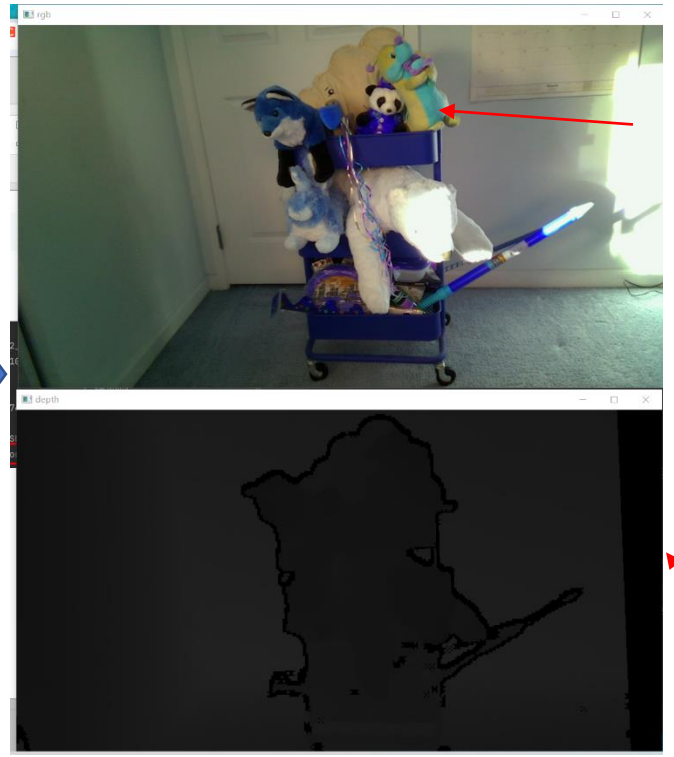
传感器数据获取-open3D

- 传感器参数获取

- Realsense bag播放 in opencv

```
bag_reader = o3d.t.io.RSBagReader()
bag_reader.open('../data/[redacted].bag')
im_rgbd = bag_reader.next_frame()
count = 0
while not bag_reader.is_eof():
    print(count)
    im_rgbd = bag_reader.next_frame()
    rgb_im = np.asarray(im_rgbd.color)
    depth_im = np.asarray(im_rgbd.depth)
    # merge_im = np.vstack((rgb_im, depth_im))
    cv2.imshow("rgb", rgb_im)
    cv2.imshow("depth", depth_im)
    k = cv2.waitKey()
    if k == 27:
        #break
        count += 1
        continue
cv2.destroyAllWindows()
bag_reader.close()
```

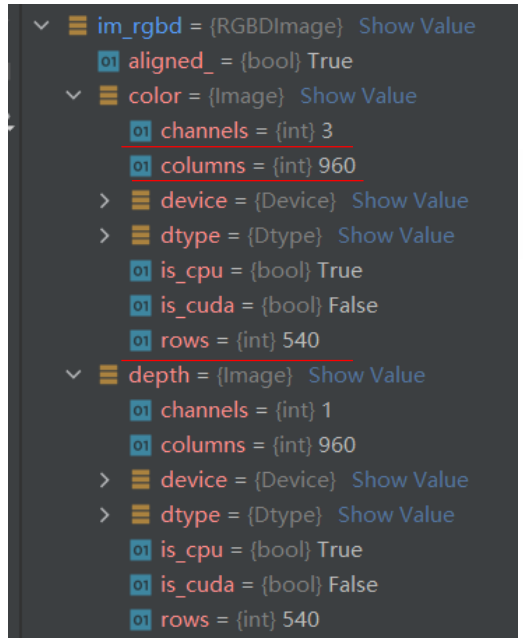
自定义数据路径



Opencv默认 BGR

在Debug模式下查看数据结构，注意RGBD数据的构成：

FOV差异



Part 5 | Realsense传感器初探

传感器数据获取

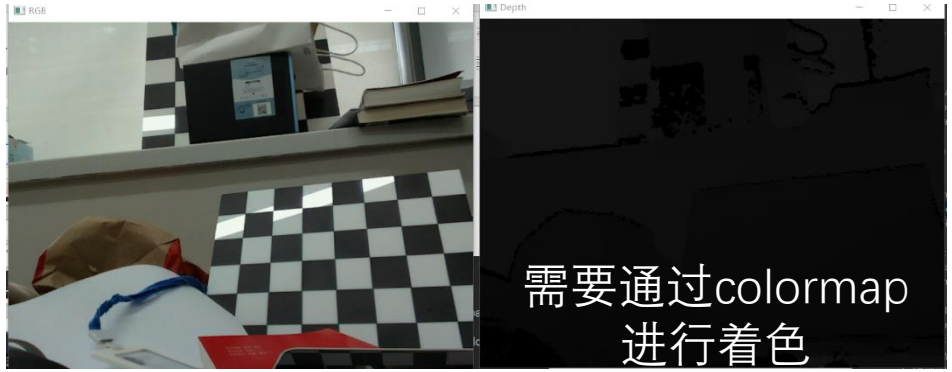
- 传感器实时2D数据处理和存储

```
def save_bag(savepath):
    config_filename = './realsense.json'
    with open(config_filename) as cf:
        rs_cfg = o3d.t.io.RealSenseSensorConfig(json.load(cf))
    #  only if with a config
    data = time.strftime("%Y%m%d_%H%M%S", time.localtime())
    bag_filename = savepath+"/"+data+'.bag'
    o3d.t.io.RealSenseSensor.list_devices()
    rs = o3d.t.io.RealSenseSensor()
    rs.init_sensor(rs_cfg, 0, bag_filename)
    rs.start_capture(True)
```

将传感器数据存储到本地，方便后处理计算。

名称	修改日期	类型	大小
20240820_161116.bag	2024/8/20 16:11	BAG 文件	284,904 KB
L515_test_s.bag	2024/8/20 14:24	BAG 文件	10,702 KB

利用opencv对数据进行可视化，需要注意不同数据的通道数有所差异




```
count = 0
while True:
    im_rgbd = rs.capture_frame(True, True)
    rgb = np.asarray(im_rgbd.color)
    rgb = cv2.cvtColor(rgb, cv2.COLOR_BGR2RGB)
    name = os.path.join(bag_filename, str(time.time())+'.jpg')
    #name = os.path.join(bag_filename, str(count).zfill(8)+'.jpg')
    count += 1
    cv2.imwrite(name, rgb)
    #depth = np.asarray(im_rgbd.depth)
    #depth = cv2.cvtColor(depth, cv2.COLOR_GRAY2BGR)
    #rgb = cv2.resize(rgb, 640, 360)
    cv2.imshow("RGB", rgb)
    #cv2.imshow("Depth", depth)
    k = cv2.waitKey(90)
    if k == 27:
        break
cv2.destroyAllWindows()
rs.stop_capture()
```

Part 5 | Realsense传感器初探

传感器数据获取-open3D

② vis_in_open3d.py

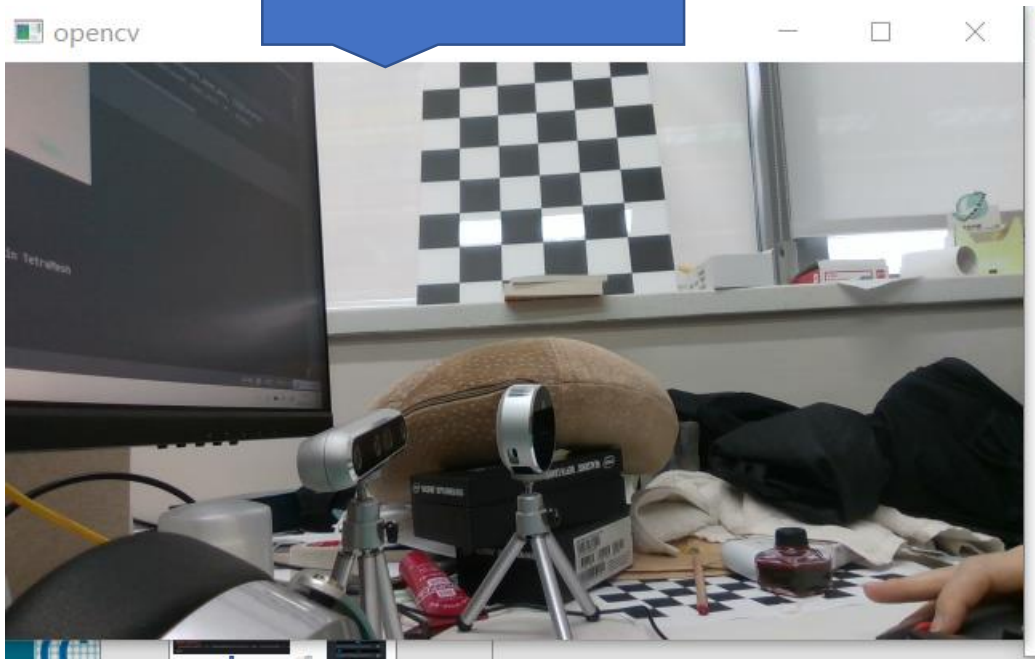
- 传感器参数获取

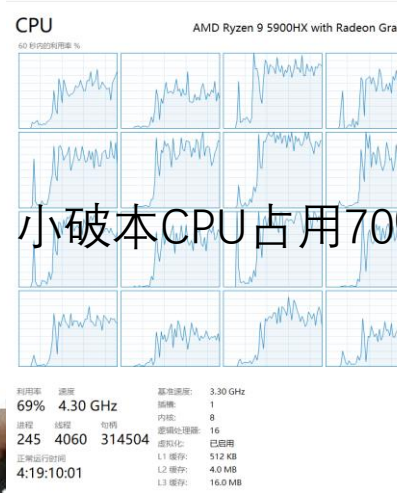
```
def camera_config():
    o3d.t.io.RealSenseSensor.list_devices()
```

首先进行参数获取，方便调整config()参数，适配传感器，再注释掉。

RGB在opencv窗口

点云在open3d窗口





小破本CPU占用70

```
config.enable_stream( *args: rs.stream.depth, 1280, 720, rs.format.z16, 15)
config.enable_stream( *args: rs.stream.color, 1280, 720, rs.format.bgr8, 15)
```

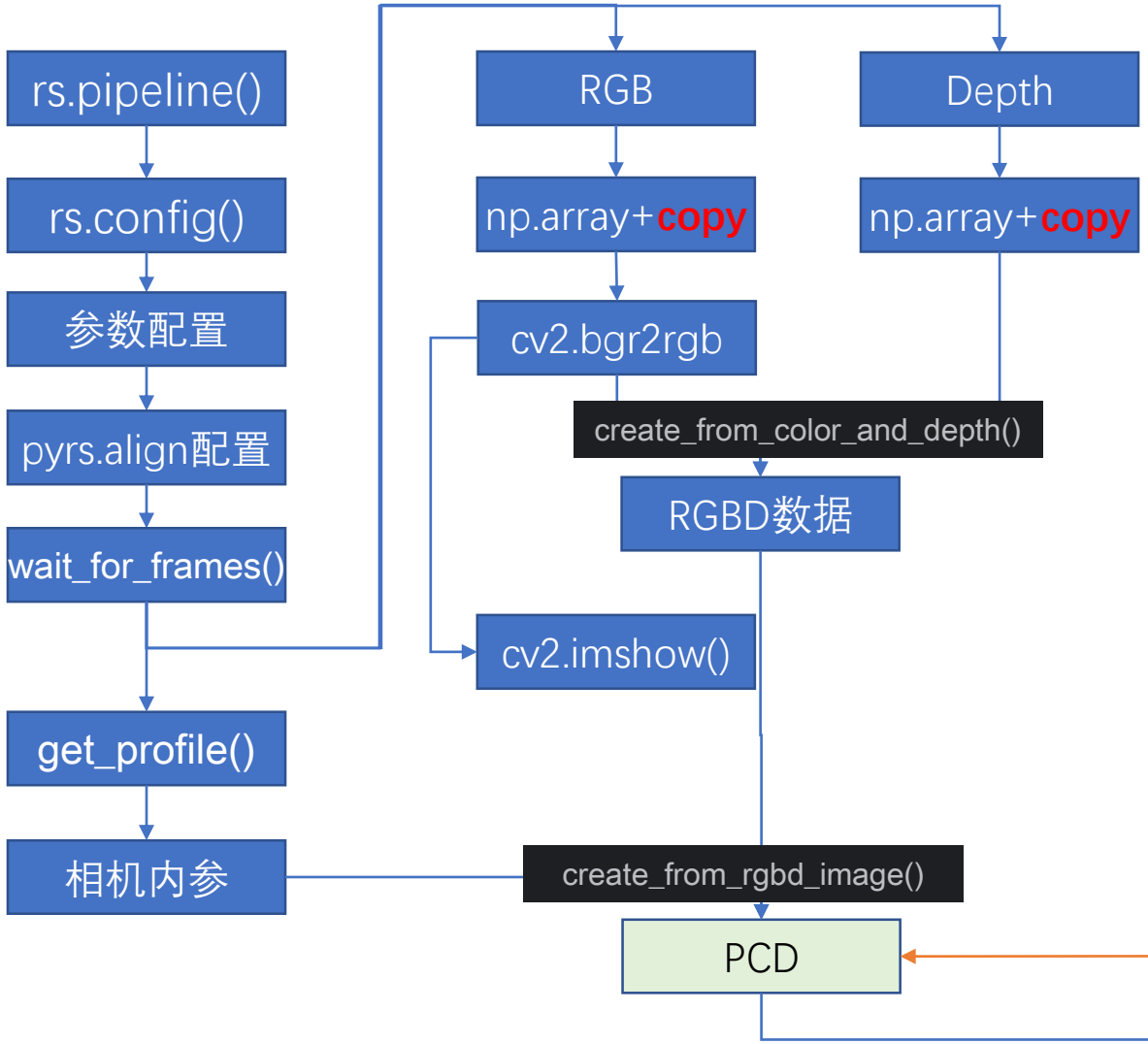
15

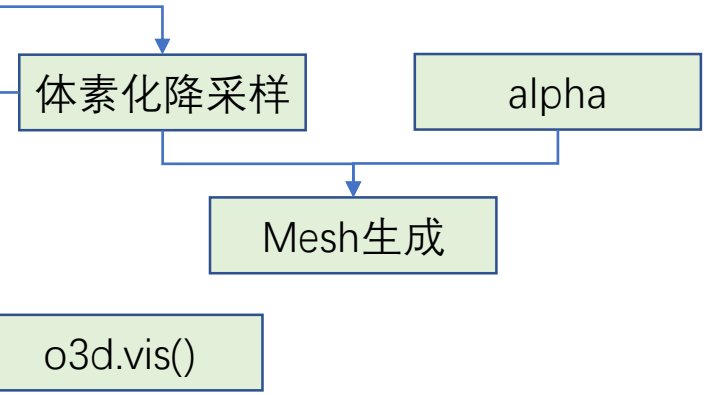
```
FPS: 9.169600938967136
FPS: 9.208018342372538
FPS: 7.932982166656089
FPS: 8.998227349212206
FPS: 10.575967172197897
FPS: 9.196332502597963
FPS: 8.924507590293706
FPS: 9.171366992250194
```

与可视化渲染效果有关
Voxel=**0.05**→0.02 掉到4fps

Part 5 | Realsense传感器初探

传感器数据获取-open3D






```
pcd = o3d.geometry.PointCloud()
pointcloud.points = pcd.points
pointcloud.colors = pcd.colors
```

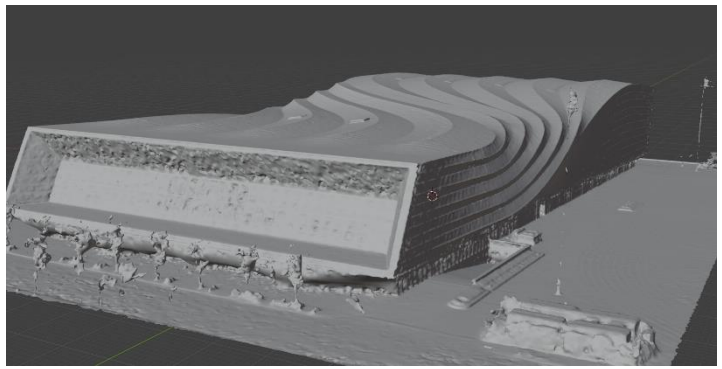
Part 5 | Realsense传感器初探

传感器数据获取

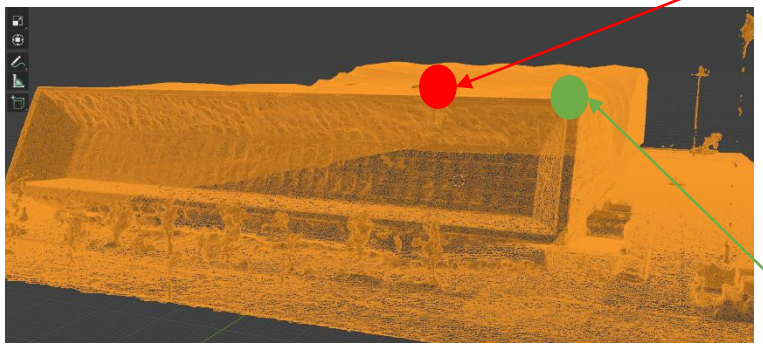
③ merge_img_depth_2_mesh.py

In 3D computer graphics and solid modeling, a **polygon mesh** is a collection of **vertices**, **edges** and **faces** that defines the shape of a polyhedral object's surface.

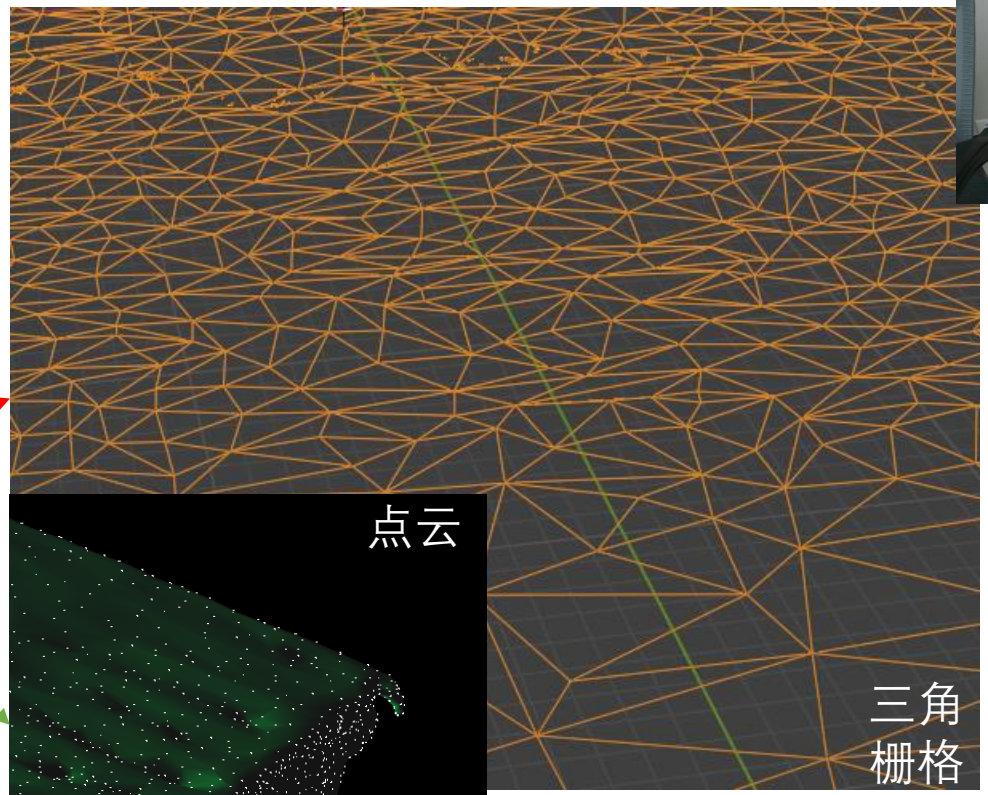
Mesh with Normal



Mesh



细节



8-bit RGB　　16-bit depth

相机内参

Open3D has a data structure for 3D triangle meshes called TriangleMesh.

Part 5 | Realsense传感器初探

传感器数据获取

- 传感器实时3D数据处理和三维可视化

获取配置，参数设置

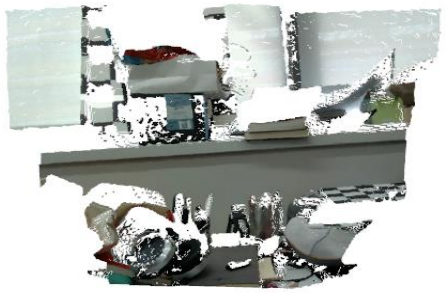
```
align = rs.align(rs.stream.color)
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.rgb8, 30)
```

处理流和相机外参

```
pipeline = rs.pipeline()
profile = pipeline.start(config)
intr = profile.get_stream(rs.stream.color).as_video_stream_profile().get_intrinsics()
pinhole_camera_intrinsic = o3d.camera.PinholeCameraIntrinsic(
    intr.width, intr.height, intr.fx, intr.fy, intr.ppx, intr.ppy)
extrinsic = [[1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0], [0, 0, 0, 1]]
```

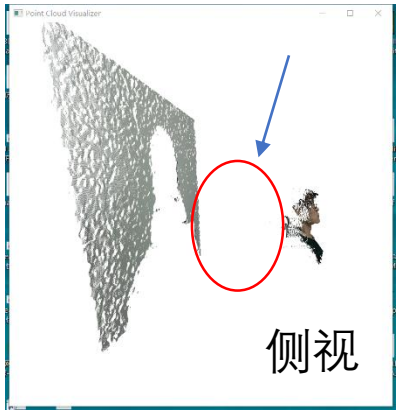
数据格式转换

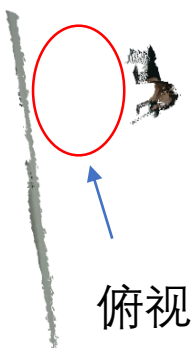
```
def convert_rs_frames_to_pointcloud(rs_frames):
    aligned_frames = align.process(rs_frames)
    rs_depth_frame = aligned_frames.get_depth_frame()
    np_depth = np.asanyarray(rs_depth_frame.get_data())
    o3d_depth = o3d.geometry.Image(np_depth)

    rs_color_frame = aligned_frames.get_color_frame()
    np_color = np.asanyarray(rs_color_frame.get_data())
    o3d_color = o3d.geometry.Image(np_color)

    rgbd = o3d.geometry.RGBDImage.create_from_color_and_depth(
        o3d_color, o3d_depth, depth_scale=4000.0, convert_rgb_to_intensity=False)

    pcd = o3d.geometry.PointCloud.create_from_rgbd_image(
        rgbd, pinhole_camera_intrinsic, extrinsic)

    return pcd
```



注意ToF测距原理：视距内
非视距内无法被感知，如何解决？

类人空间认知--SLAM

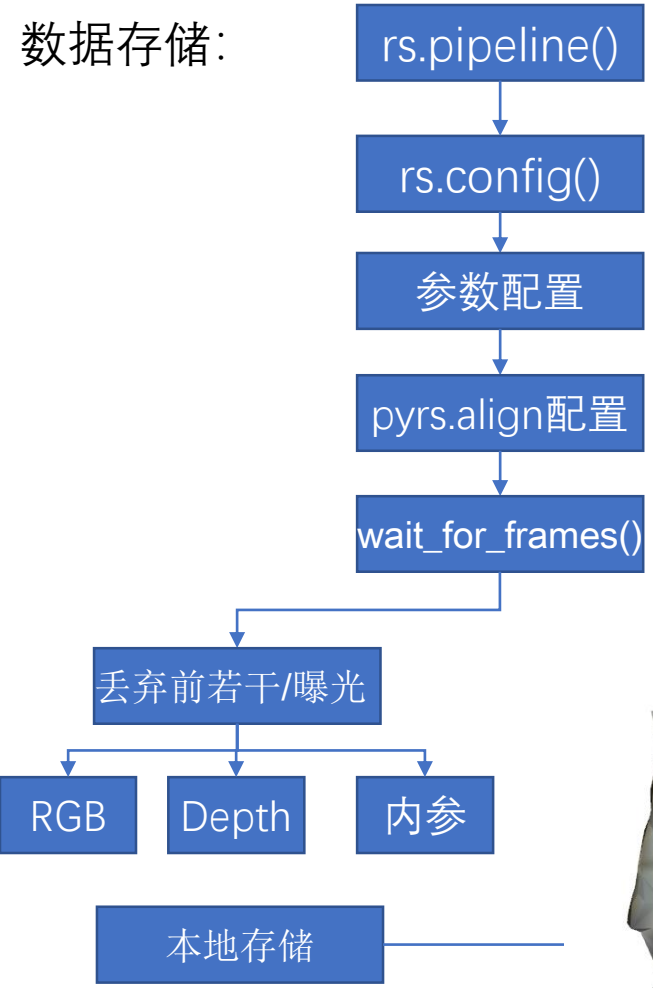




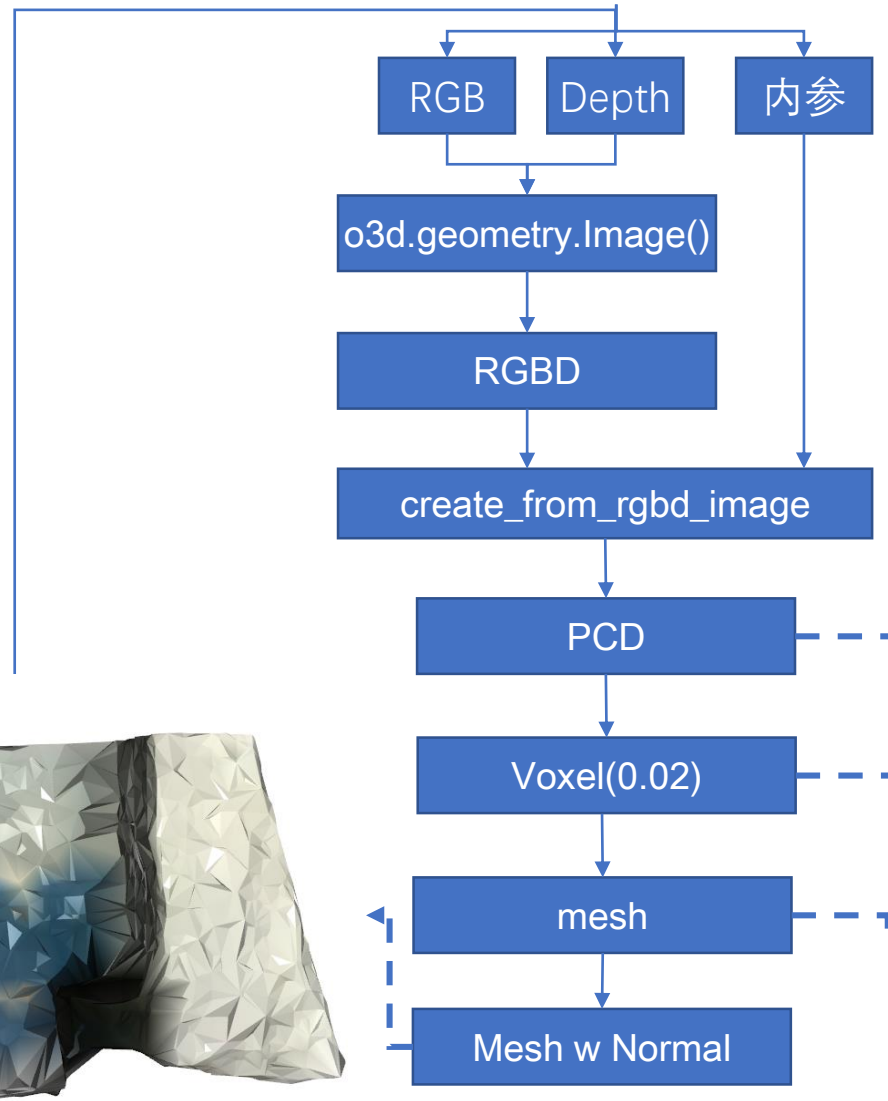
Part 5 | Realsense传感器初探

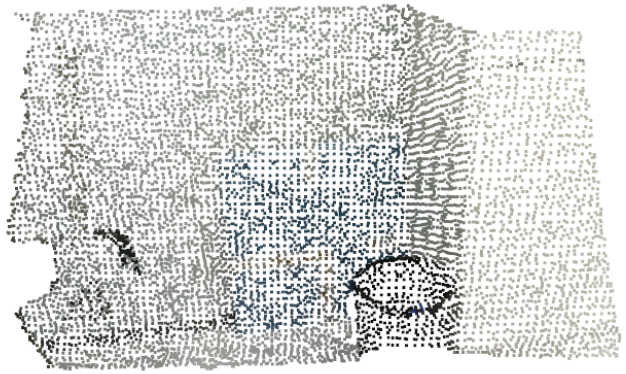
传感器数据获取

数据存储:



数据读取:





Part 5 | Realsense传感器初探

传感器数据获取-open3d基础函数

④ basic_op.py .pcd文件存储

RGB=829K, depth=423K

PCD	write_ascii	compressed	文件大小
1			13.8M
2	√		48.5M
3	√	√	48.5M
4		√	6.09M

Format	Description
`xyz`	Each line contains `[x, y, z]`, where `x`, `y`, `z` are the 3D coordinates
`xyzn`	Each line contains `[x, y, z, nx, ny, nz]`, where `nx`, `ny`, `nz` are the normals
`xyzrgb`	Each line contains `[x, y, z, r, g, b]`, where `r`, `g`, `b` are in floats of range `[0, 1]`
`pts`	The first line is an integer representing the number of points. The subsequent lines follow one of these formats: `[x, y, z, i, r, g, b]`, `[x, y, z, r, g, b]`, `[x, y, z, i]` or `[x, y, z]`, where `x`, `y`, `z`, `i` are of type `double` and `r`, `g`, `b` are of type `uint8`
`ply`	See Polygon File Format, the ply file can contain both point cloud and mesh data
`pcd`	See Point Cloud Data

```
o3d.io.write_point_cloud(file_name, pcd, write_ascii=False, compressed=False, print_progress=True)
```

```
o3d.io. read_point_cloud(filename, format='auto', remove_nan_points=False, remove_infinite_points=False, print_progress=False)
```

传感器数据获取

PCD文件格式一般包括:

1. Version 版本信息
2. Fields 文件头
3. Size 大小
4. Type 类型
5. Count 元素数量
6. Width 宽度
7. Height 高度
8. Viewpoint 视点信息
9. Points点规模
10. Data 真数据部分

ASCII编码

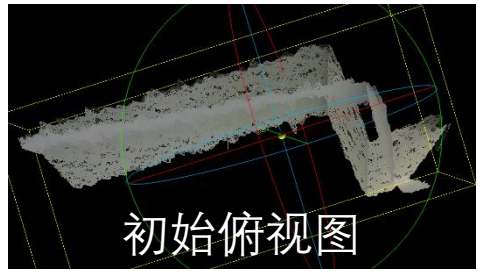
```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 907162
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 907162
DATA binary
```

非ASCII编码

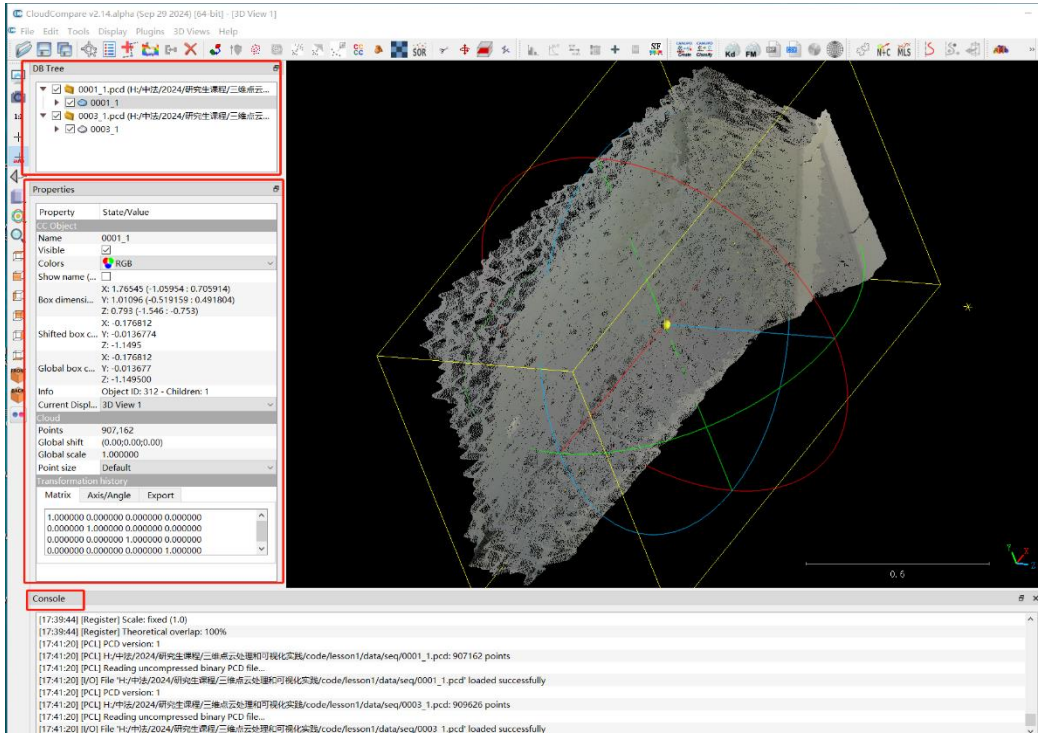
```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 907162
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 907162
DATA ascii
-0.8070340818 0.4785680425 -1.156999946 1.263151315e-38
-0.8057702106 0.4785680425 -1.156999946 1.263151315e-38
-0.8058970811 0.479395338 -1.159000039 1.253931752e-38
-0.8046310251 0.479395338 -1.159000039 1.244712189e-38
-0.8033649692 0.479395338 -1.159000039 1.244712189e-38
```

Part 5 | Realsense传感器初探

传感器数据获取

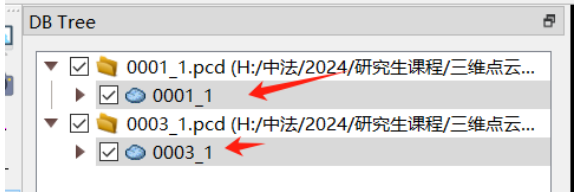


Cloudcompare： https://www.cloudcompare.org/ 2.14

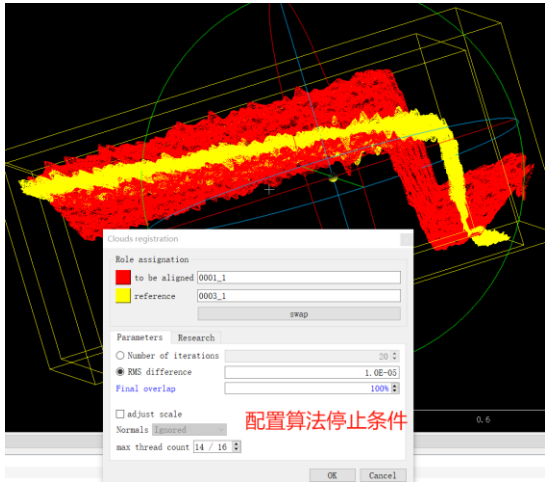


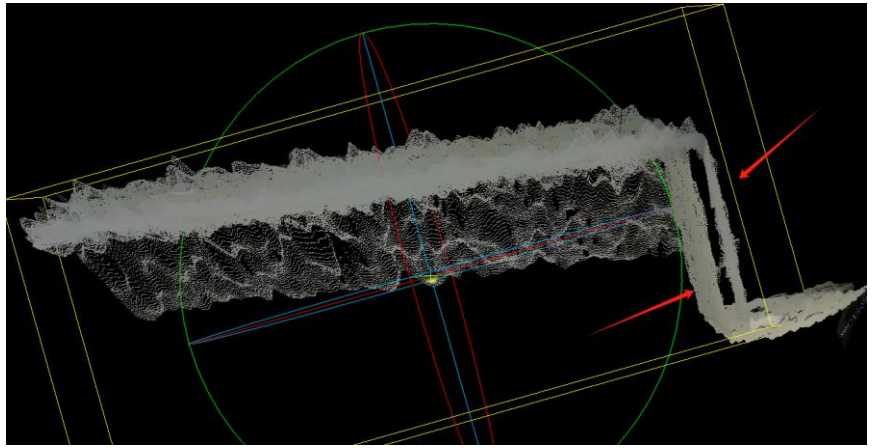
配准功能：

1. 导入两帧点云，需要有一定的共视场（在图像配准里面为 同名点）
    1. 先测试 frame1 和 frame2： 失败
    2. 再测试frame1 和 frame3： 成功
2. 在DB Tree选择目标点云文件
3. Tools-Registration-ICP
4. 执行并等待



匹配查看PCD文件信息，并且调整相关属性。





传感器数据获取-open3d基础函数

④ basic_op.py

- 法向估计
- 体素化
- Bounding box生成
- 凸包生产
- 聚类
- 平面检测

开始实验

Part 5 | Realsense传感器初探

传感器数据获取

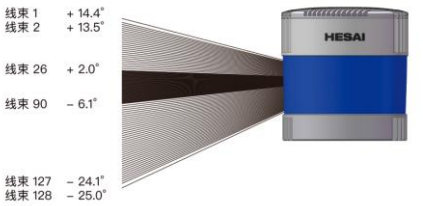


图 1.5 线束分布示意图

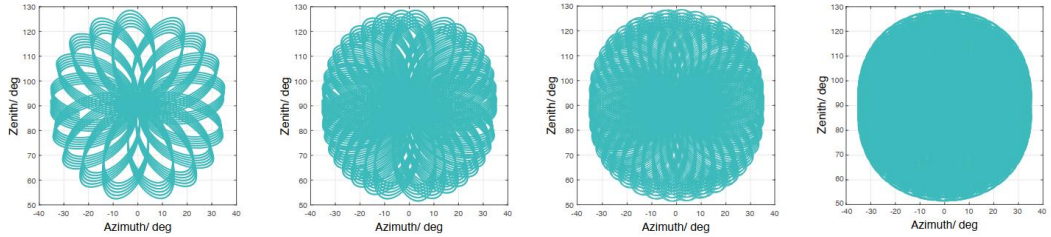
Velodyne激光雷达参数对比（公开数据集可能用到，已退出大陆市场）

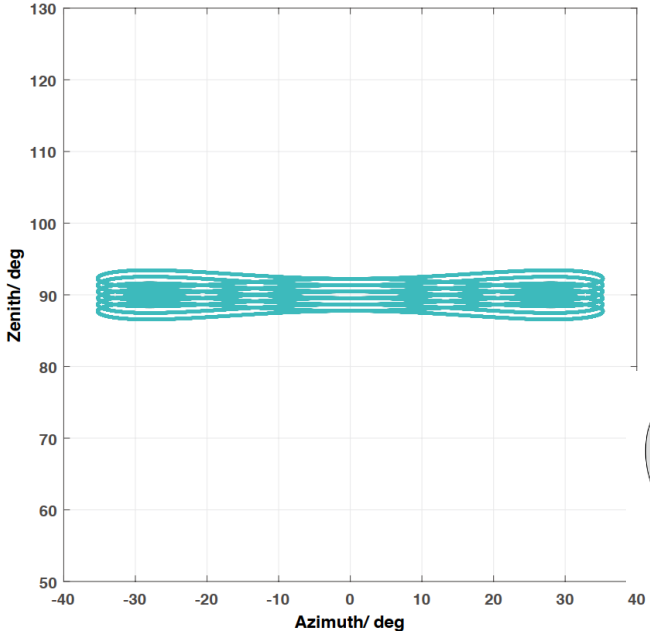
Sensor	HDL-64E	HDL-32	Puck	Puck LITE	Puck Hi-Res	Ultra Puck	Alpha Prime
							
Range	Up to 120m	Up to 100m	100m	100m	100m	200m	Up to 245m[5]
Range Accuracy	Up to ±2 cm (Typical)[4]	Up to ±2 cm (Typical)[1]	Up to ±3 cm (Typical)[1]	Up to ±3 cm (Typical)[1]	Up to ±3 cm (Typical)[1]	Up to ±3 cm (Typical)[1]	Up to ±3 cm (Typical)[1]
# of Lines	64	32	16	16	16	32	128
Horizontal FoV	360°	360°	360°	360°	360°	360°	360°
Vertical FoV	26.9°	41.33°	30°	30°	20°	40°	40°
Horizontal Resolution	0.08° – 0.35°	0.08° - 0.33°	0.1° – 0.4°	0.1° – 0.4°	0.1° – 0.4°	0.1° – 0.4°	0.1° – 0.4°
Vertical Resolution	0.4°	1.33°	2.0°	2.0°	1.33°	0.33° (min)	0.11° (min)
Points Per Second (Single Return Mode)	~ 1,300,000	~ 695,000	~ 300,000	~ 300,000	~ 300,000	~ 600,000	~ 2,400,000
Points Per Second (Dual Return mode)	~ 2,200,000[5]	~ 1,390,000	~ 600,000	~ 600,000	~ 600,000	~ 1,200,000	~ 4,800,000
Refresh Rate	5-20 Hz	5-20 Hz	5-20 Hz	5-20 Hz	5-20 Hz	5-20 Hz	5-20 Hz
Operating Voltage	12V - 32V	9 V – 18 V	9 V – 18 V	9 V – 18 V	9 V – 18 V	10.5 V – 18 V	9 V – 28 V
Power Consumption	60 W (Typical)[2]	12 W (Typical)[2]	8 W (Typical)[2]	8 W (Typical)[2]	8 W (Typical)[2]	10 W (Typical)[2]	22 W (Typical)[2]
Weight (without cabling)	~ 28 lbs. (12.7 Kg)	~1.0 kg	~830 g	~590 g	~830 g	~925 g	~3.5 kg
Operating Temp	-10°C to +60°C[3]	-10°C to +60°C[3]	-10°C to +60°C[3]	-10°C to +60°C[3]	-10°C to +60°C[3]	-20°C to +60°C[3]	-20°C to +60°C[3]
Storage Temp	-40°C to +85°C	-40°C to +105°C	-40°C to +105°C	-40°C to +105°C	-40°C to +105°C	-40°C to +85°C	-40°C to +105°C
Output	UDP packets over Ethernet	UDP packets over Ethernet	UDP packets over Ethernet	UDP packets over Ethernet	UDP packets over Ethernet	UDP packets over Ethernet	UDP packets over Ethernet
Ethernet Connection	100 Mbps	100 Mbps	100 Mbps	100 Mbps	100 Mbps	100 Mbps	1000 Mbps
GPS Timesync	$GPRMC	$GPRMC + $GPGGA	$GPRMC + $GPGGA	$GPRMC + $GPGGA	$GPRMC + $GPGGA	$GPRMC + $GPGGA	$GPRMC + $GPGGA
Laser	903nm Class 1 eye safe	903nm Class 1 eye safe	903nm Class 1 eye safe	903nm Class 1 eye safe	903nm Class 1 eye safe	903nm Class 1 eye safe	903nm Class 1 eye safe
Water Resistance	IP67	IP67	IP67	IP67	IP67	IP67	IP67

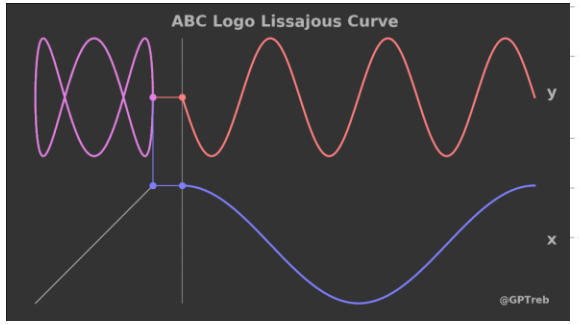
Part 5 | Realsense传感器初探

传感器数据获取

非重复扫描激光雷达Livox参数对比。
包含两种扫描模式：玫瑰线rosedosventor 、李萨如（Lissajous）







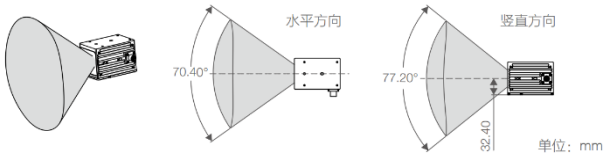


图 3.1.1 Livox Avia 有效 FOV 范围

产品型号	MID-40	HAP	TELE-15	MID-70	AVIA
					
应用场景	移动机器人、智慧物流等	自动驾驶、 智能辅助驾驶系统等	轨道交通、智慧交通	无人配送车、 园区物流车等	测绘、安防、智慧城市等
量程[1] (@ 100 klx)	90 m @ 10% 反射率 130 m @ 20% 反射率 260 m @ 80% 反射率	150 m @ 10% 反射率 200 m @ 80% 反射率	320 m @ 10% 反射率 500 m @ 50% 反射率	90 m @ 10% 反射率 130 m @ 20% 反射率 260 m @ 80% 反射率	190 m @ 10% 反射率 230 m @ 20% 反射率 320 m @ 80% 反射率 450 m @ 80% 反射率, 0 klx
虚警率[2] (@ 100 klx)	< 0.01%	< 0.01%	< 0.01%	< 0.01%	< 0.0003%
距离精度[3] (1σ @ 20m)	2 cm	2 cm	2 cm	2 cm	2 cm
角度精度	< 0.05°	< 0.1°	< 0.03°	< 0.1°	< 0.05°
水平视场 (FOV)	38.4°	120°	14.5°	70.4°	70.4°
垂直视场 (FOV)	38.4°	25°	16.1°	70.4°	77.2°
光束发散度	0.28° (垂直) × 0.03° (水平)	0.28° × 0.03°	0.12° × 0.02°	0.28° × 0.03°	0.28° × 0.03°
数据率 (点/秒)	~100,000	452,000	~240,000 ~480,000(双回波)	~100,000 ~200,000(双回波)	~240,000 ~480,000(双回波) ~720,000(三回波)
供电电压范围	10 ~ 16 V	9 ~ 18V	10 ~ 15 V	10 ~ 15 V	10 ~ 15 V
功率	10 W(启动功率 25W)	14 W(启动功率 26W)	12 W(启动功率 30W)	8 W(启动功率 30W)	9 W(启动功率 16W)
尺寸 (mm)	88×69×76	105×131.6×65	122×105×95 无风扇时 112×122×85	97×64×62.7	91×61.2×64.8
重量	760 g	1120g	1600g 无风扇时 1500g	580g	498g
工作温度	-20˚C to 65˚C	-40˚C to 85˚C	-40˚C to 85˚C	-20˚C to 65˚C	-20˚C to 65˚C
数据同步	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)	IEEE 802.1AS (gPTP)	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)
接口	Ethernet	HAP (T1) : 100 Base-T1 标准 HAP (TX) : 100 Base-TX 标准	Ethernet	Ethernet	Ethernet
激光波长及安全级别	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全