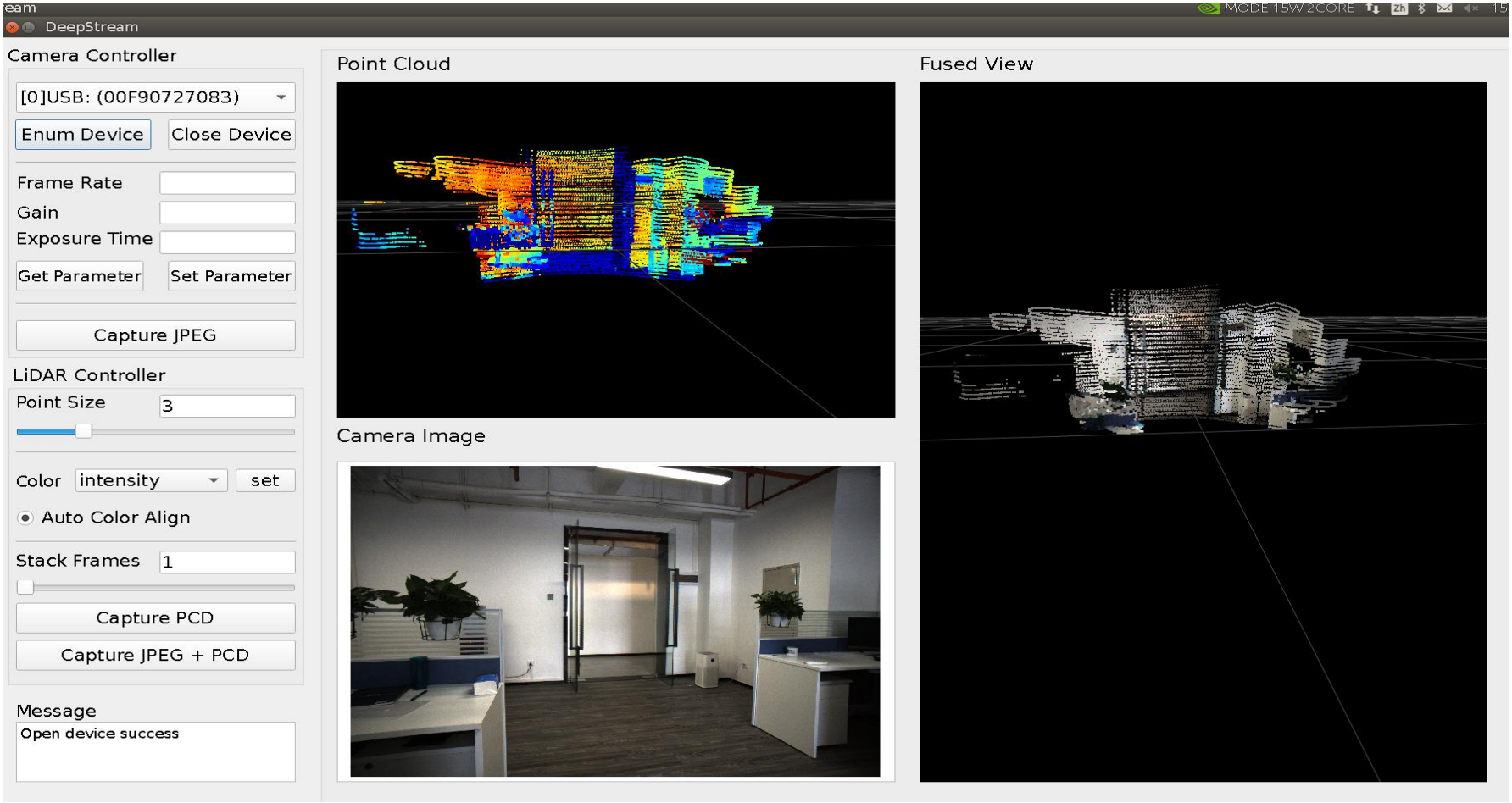
Project Report

Li Rong

SCUT Machine Intelligence Laboratory

最终成果展示

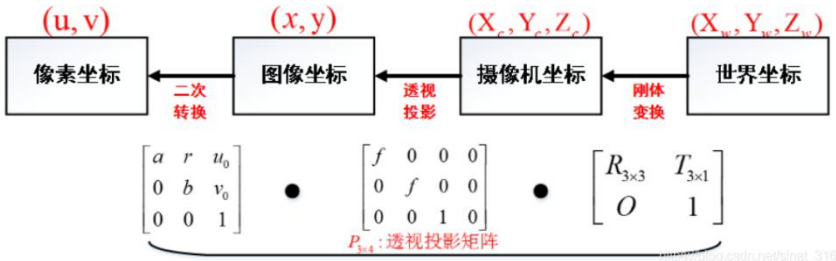
Video Link : https://www.bilibili.com/video/BV1uf4y1L7Jf/



相机内参标定

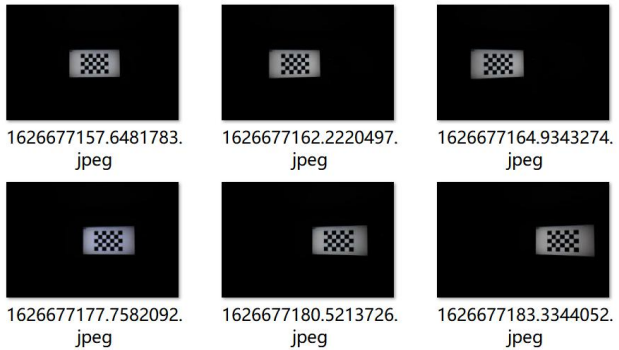
➢ 任务描述：

计算相机内参和畸变矩阵。以矫正图像和进行后续投影3D点云到2D图像的任务。



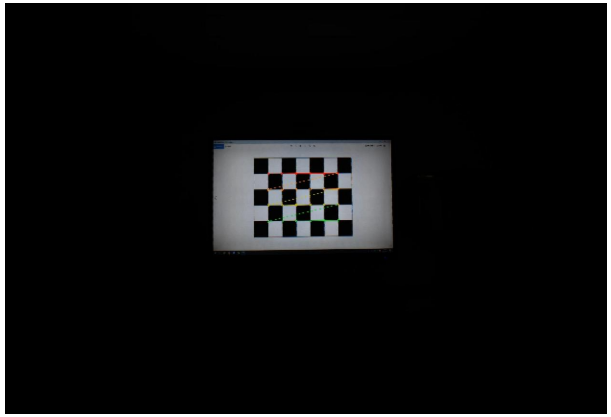
➢ 数据采集：

本次标定选择黑暗条件下85寸电视来显示标定图片，一共采集了35帧各个角度的图片。



➢ 张正友标定法过程：

1. 初始角点检测
2. 进一步提取亚像素角点
3. 绘制角点观察结果
4. 相机标定
5. 重投影空间三维点，评价标定效果



相机内参标定

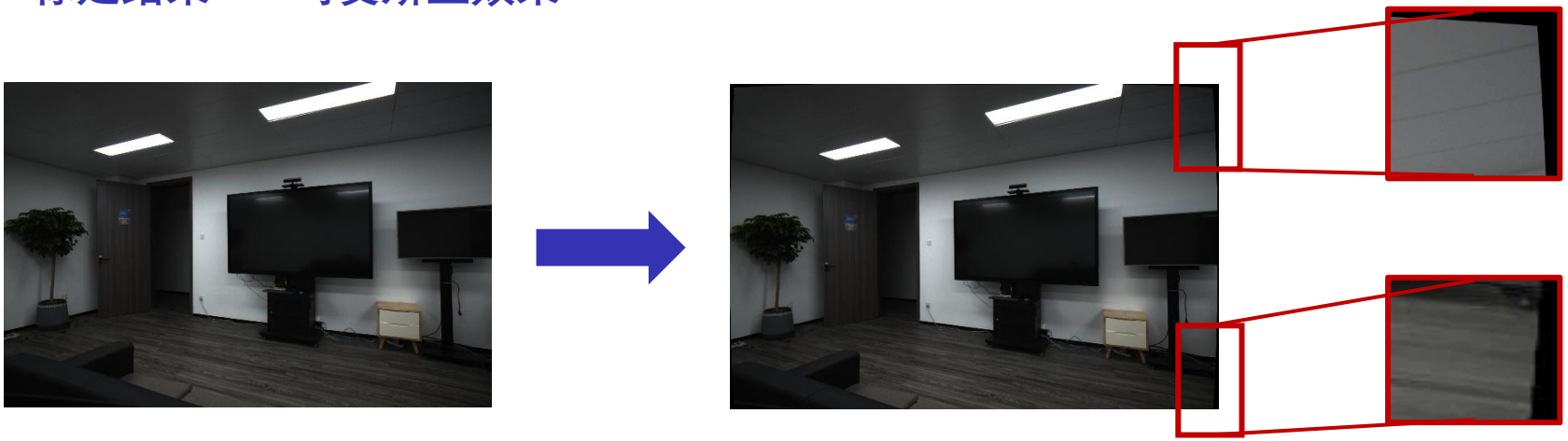
➢ 遇到过的问题：

1. 张正友标定法对**环境干扰**非常敏感。因此最后选择了**黑暗条件下电视屏幕**显示的标定板。
2. 拍摄**角度**不够diversity、拍摄**图片数目**不足导致精度不足。
3. 拍摄时**曝光强度**需要仔细调节，否则无法识别角点。

➢ 标定结果——重投影误差：

0.014394424473247082 (误差在允许范围之内)

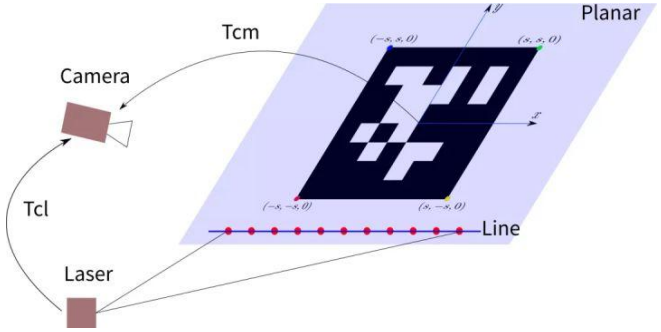
➢ 标定结果——畸变矫正效果：



激光雷达和相机外参标定

➢ 任务描述：

计算激光雷达和相机的外参，以实现不同传感器之间的坐标转换。

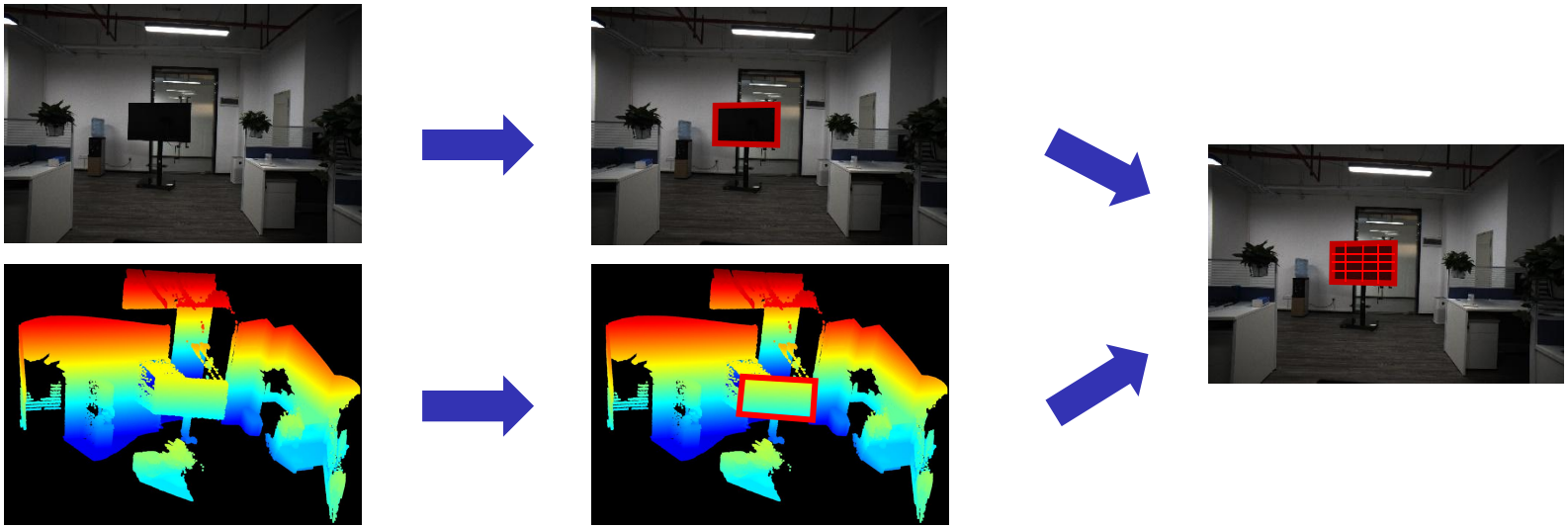


➢ 数据采集：

本次标定选择1m x 1.5m的黑色电视作为标定板。拍摄距离在3米左右。共采集数据15组。

➢ 标定方法：

基于平面约束的相机激光标定算法

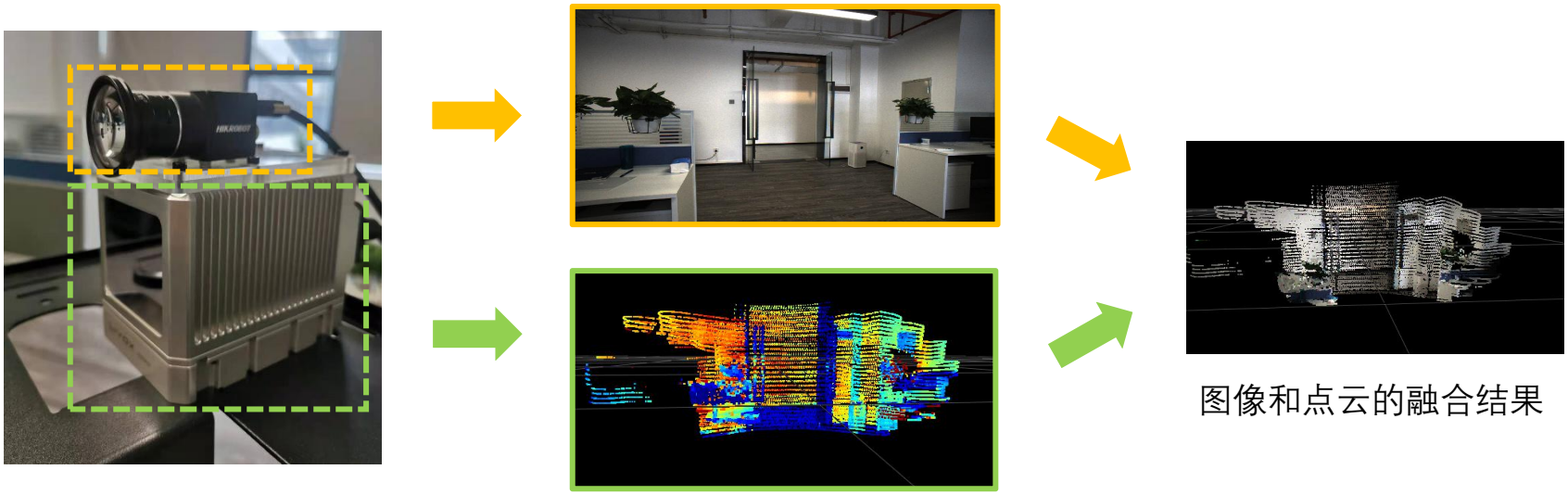


激光雷达和相机外参标定

➢ 遇到过的问题：

1. 单帧点云密度不足，导致标注的点云角点精度不足，因此影响外参标注的精度。最终选择**叠加多帧点云**的方式解决问题。
2. 标记图像角点的精度不足，导致标注的图像角度精度不足，因此影响外参标注的精度。最终采用**亚像素优化**对手工选择的像素进一步优化解决问题。

➢ 标定效果展示：

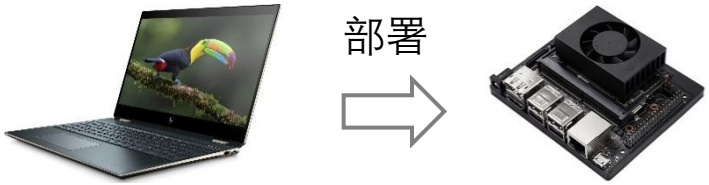


传感器　　传感器采集的数据　　图像和点云的融合结果

嵌入式设备的应用部署

➢ 任务描述：

由于嵌入式设备算力有限，因此在高算力电脑上写好的程序在嵌入式设备上无法达到流畅运行。



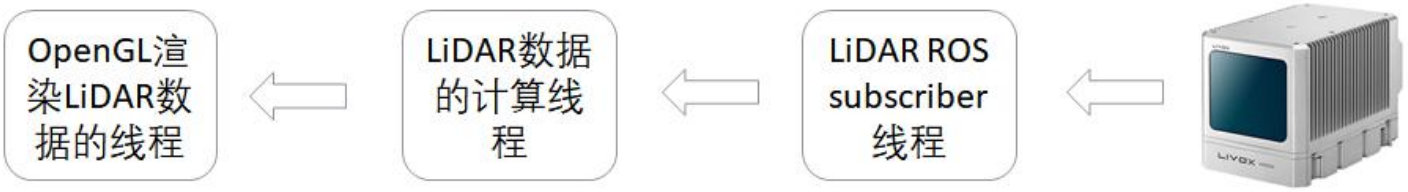
➢ 一些经验：

1. 主要是定位性能瓶颈，分析瓶颈产生的原因，针对性解决。
2. 修改/删除冗余代码，换用更高效率的表达。
3. 更换数据处理方式 （比如matplot把灰度图映射到彩色图片速度很慢，可以通过把颜色映射矩阵保存成numpy数组来提速）。
4. 有循环的地方要尤其注意。
5. 一些非必要的操作可以移出callback，并通过添加新的触发组件来保留这个操作。
6. 如果是IO密集型的操作可以采用多线程编程。
7. 线程的while中可以加入短暂的sleep，为其他操作留出空隙可以提升运行的流畅度。
8. 把Python代码改成C++并不一定能提升效率，比如numpy运算的底层是C，所以直接使用numpy运算效率并不低。
9. 可以通过观察CPU的占用率来辅助定位问题。

嵌入式设备的应用部署

➢ 关于Python多线程的经验：

1. Python的多线程并不总是有效。Python的多线程受到GIL的限制，因此对IO密集型的应用比较有效，如果是计算密集型应用，使用多线程可能没有预料中的提升。
2. 线程不同步可能的原因：CPU计算资源不足。由于Python的GIL机制的存在，只有当其他线程释放出足够的计算资源时候，等待的线程才能运行。
3. 多线程程序CPU Core数目选择： 运行应用的CPU Core不是越多越好建议使用能满足app算力的最小CPU Core数目，解释如下：



本应用中至少存在以上三个处理点云的线程（处理图像的线程类似）。LiDAR ROS subscriber 线程是处理IO的线程，它在得到point cloud数据之后需要把数据传递给LiDAR数据的计算线程，然后OpenGL渲染LiDAR数据的线程再把point cloud数据画出来。由于这三个线程使用的是同一份数据，但是由于运行在不同的核心上面，所以切换的时候还要把这个数据复制至少两遍，是很耗费时间的。注意：点云的数据量是很大的，比如我们app里的点云数据大小是(21000, 4)，因此数据传递费时，如果数据量不大则没有影响。