

学号：2053932

姓名：雷翔

指导老师：朱亚萍

日期：2024.6.18

## 引言

本报告记录了“SLAM理论与系统”课程期末实验汇报的工作。主要目标是使用三角化方法估计 Portland\_hotel 数据集中图像的深度，并对结果进行误差分析。此外，还尝试复现 DELTAS 论文中提出的方法，以比较传统基于特征的深度估计与学习的三角化方法。

SLAM (Simultaneous Localization and Mapping) 是机器人和计算机视觉领域的核心研究方向之一。SLAM 技术使机器人能够在未知环境中构建地图并同时确定自身的位置，对于自动驾驶、无人机导航、增强现实等应用具有重要意义。传统的 SLAM 方法依赖于特征检测和几何推理来进行地图构建和定位，而这些方法在面对复杂或动态环境时可能表现出一定的局限性。

随着深度学习技术的发展，基于学习的方法在计算机视觉任务中显示出了显著的优势，特别是在特征提取和匹配方面。通过复现和改进现有的学习方法，可以为 SLAM 系统的研究和应用提供新的思路和方法，进一步提高系统的鲁棒性和精度。

本次实验的主要目标包括两个方面：

### 1. 传统的三角化方法【必做实验】：

- 使用 SURF、SIFT 和 ORB 三种特征检测器，估计 Portland\_hotel 数据集中图像的深度。
- 计算和分析误差指标，包括绝对误差 (Abs)、均方根误差 (RMSE) 和对数均方根误差 (RMSE log)。

### 2. 学习的三角化方法【扩展实验】：

- 复现 DELTAS 论文中的方法，利用深度学习进行三角化和稀疏点的密集化。
- 比较传统方法与学习方法在深度估计中的性能差异，验证 DELTAS 方法的有效性。

## 方法


### 必做实验

#### 数据处理

使用 Portland\_hotel 数据集进行实验，包含大约 10641 张图像。图像通过以下步骤下载：

```
wget -r -np -nH --cut-dirs=1 -R "index.html*"
https://sun3d.cs.princeton.edu/data/Portland_hotel/
```

需要注意的时候，Portland\_hotel 并不连续，注意看下图直接从 4755 跳到 7438。

	<a href="#">0004746-000159033146.png</a>	2014-08-07 21:44	135K
	<a href="#">0004747-000159066579.png</a>	2014-08-07 21:44	136K
	<a href="#">0004748-000159100012.png</a>	2014-08-07 21:44	137K
	<a href="#">0004749-000159133445.png</a>	2014-08-07 21:44	137K
	<a href="#">0004750-000159166909.png</a>	2014-08-07 21:44	138K
	<a href="#">0004751-000159200374.png</a>	2014-08-07 21:44	138K
	<a href="#">0004752-000159233902.png</a>	2014-08-07 21:44	137K
	<a href="#">0004753-000159267430.png</a>	2014-08-07 21:44	137K
	<a href="#">0004754-000159300989.png</a>	2014-08-07 21:44	137K
	<a href="#">0004755-000159334549.png</a>	2014-08-07 21:44	137K
	<a href="#">0007438-000249260519.png</a>	2014-08-07 21:50	123K
	<a href="#">0007439-000249294047.png</a>	2014-08-07 21:50	123K
	<a href="#">0007440-000249327543.png</a>	2014-08-07 21:50	124K
	<a href="#">0007441-000249361039.png</a>	2014-08-07 21:50	124K
	<a href="#">0007442-000249394504.png</a>	2014-08-07 21:50	125K

Portland\_hotel 数据集格式

```

.
├── depth
│   ├── 0000001-000000000000.png
│   ├── ...
│   └── 0013323-000446500440.png
├── extrinsics
│   └── 20140808220511.txt
├── image
│   ├── 0000001-000000000000.jpg
│   ├── ...
│   └── 0013323-000446500152.jpg
├── intrinsics.txt
└── thumbnail
    └── 20140808220511.jpg

```

## 特征检测

使用三种不同的特征检测器：

- **SURF**：以速度和鲁棒性著称，但受专利保护，需要特殊配置。
- **SIFT**：一种高度有效的图像局部特征检测和描述方法。
- **ORB**：一种结合 FAST 关键点检测器和 BRIEF 描述子的高效替代方案。

为了使用 SURF 特征检测器，需要重新编译 OpenCV 以启用某些受专利限制的功能模块：

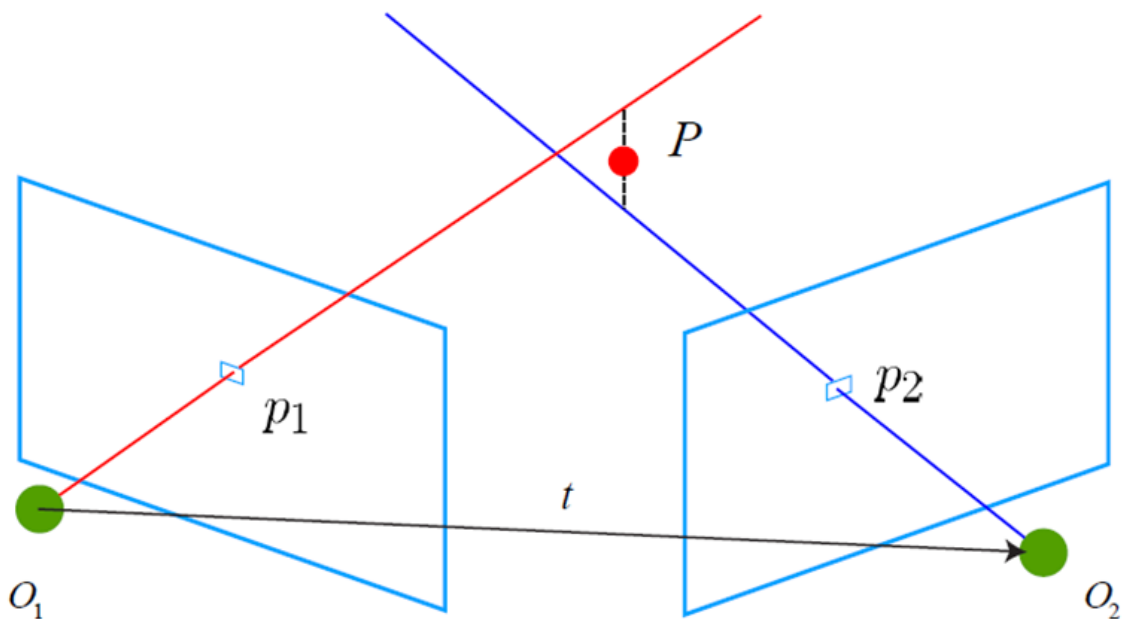
```
cmake -DOPENCV_ENABLE_NONFREE=ON ..
```

注：SIFT 专利到期、SURF 仍在专利有效期。

在特征检测阶段，通过以下步骤进行操作：

1. **特征点检测**：使用 SURF、SIFT 和 ORB 三种检测器分别检测图像中的特征点。
2. **特征描述**：计算每个特征点的描述子，以便后续匹配使用。
3. **特征匹配**：使用 BruteForce 匹配器对两个图像中的特征点进行匹配。

## 三角化



图：三角化获得地图点深度

三角化过程涉及在图像对之间匹配特征，以计算视差并估计深度。这是通过使用提供的相机内参来实现的。以下是实现三角化方法的主要代码：

```
void triangulation(  
    const vector<KeyPoint> &keypoint_1,  
    const vector<KeyPoint> &keypoint_2,  
    const std::vector<DMatch> &matches,  
    const Mat &R, const Mat &t,  
    vector<Point3d> &points) {  
    Mat T1 = (Mat_(3, 4) <<  
        1, 0, 0, 0,  
        0, 1, 0, 0,
```

```

    0, 0, 1, 0);
    Mat T2 = (Mat_<float>(3, 4) <<
        R.at<double>(0, 0), R.at<double>(0, 1), R.at<double>(0, 2), t.at<double>(0,
0),
        R.at<double>(1, 0), R.at<double>(1, 1), R.at<double>(1, 2), t.at<double>(1,
0),
        R.at<double>(2, 0), R.at<double>(2, 1), R.at<double>(2, 2), t.at<double>(2,
0)
    );

    vector<Point2f> pts_1, pts_2;
    for (DMatch m:matches) {
        // 将像素坐标转换至相机坐标
        pts_1.push_back(pixel2cam(keypoint_1[m.queryIdx].pt, K));
        pts_2.push_back(pixel2cam(keypoint_2[m.trainIdx].pt, K));
    }

    Mat pts_4d;
    cv::triangulatePoints(T1, T2, pts_1, pts_2, pts_4d);

    // 转换成非齐次坐标
    for (int i = 0; i < pts_4d.cols; i++) {
        Mat x = pts_4d.col(i);
        x /= x.at<float>(3, 0); // 归一化
        Point3d p(
            x.at<float>(0, 0),
            x.at<float>(1, 0),
            x.at<float>(2, 0)
        );
        points.push_back(p);
    }
}

```

通过使用特征检测器提取图像中的特征点，并进行特征匹配，计算图像对之间的相对运动（R 和 t），然后通过三角化方法估计深度。最后，使用误差指标对深度估计结果进行评估。

需要注意的是，由于尺度不确定性，三角化特征点的距离无法直接确定，需要依赖数据集提供的参考信息。例如，TUM数据集提供了详细的相机内参和姿态信息，有助于准确计算深度。

- The depth images are scaled by a factor of 5000, i.e., a pixel value of 5000 in the depth image corresponds to a distance of 1 meter from the camera, 10000 to 2 meter distance, etc. A pixel value of 0 means missing value/no data.

[https://cvg.cit.tum.de/data/datasets/rgbd-dataset/file\\_formats](https://cvg.cit.tum.de/data/datasets/rgbd-dataset/file_formats)

## 扩展实验

### 数据处理

每个 sequence 包含 7 张图片，作为 1 个 scene。数据处理步骤如下：

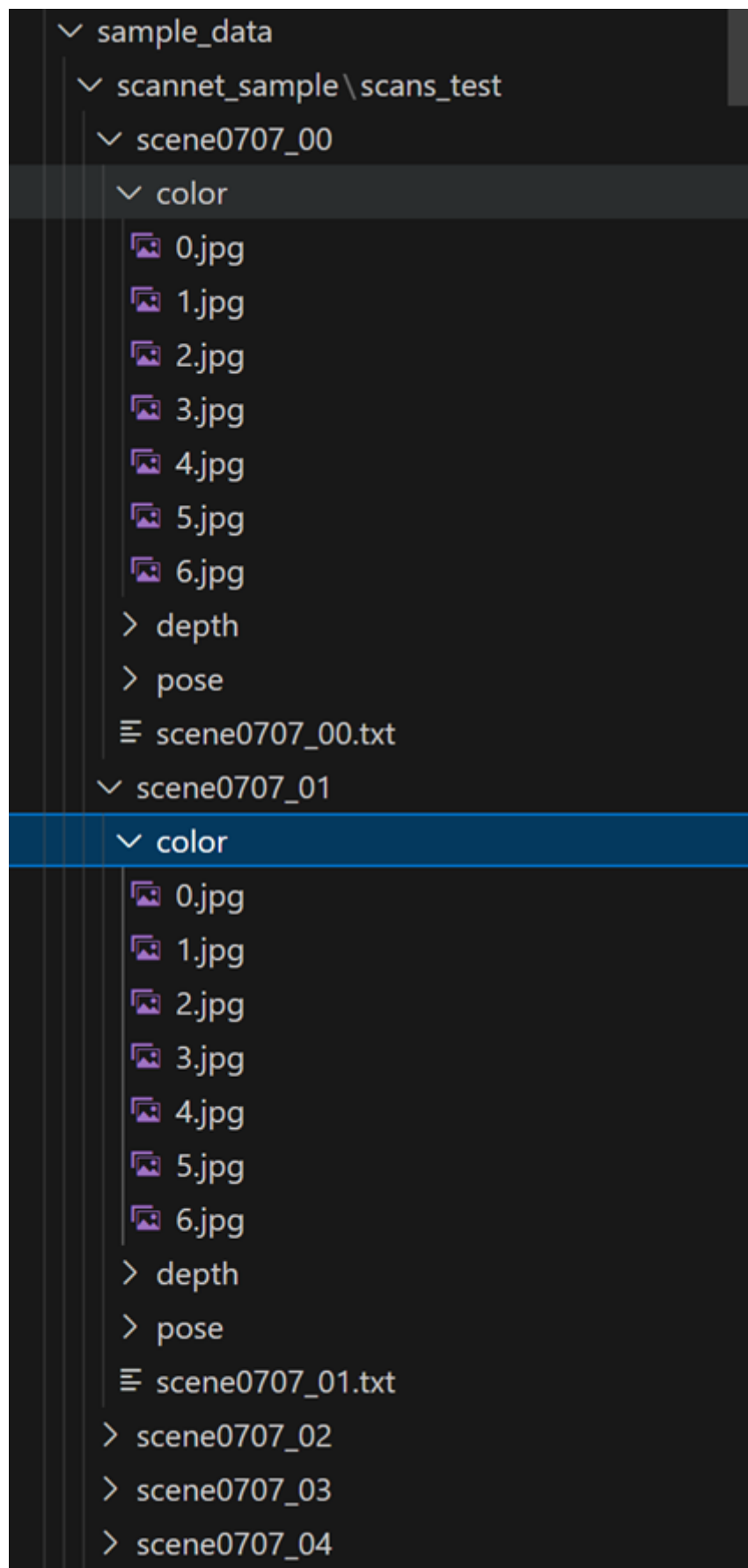
1. **读取TUM数据**：从 TUM 数据集中读取 RGB 图像、深度图像及相应的姿态信息。通过解析文件中的数据，获取每个图像的时间戳、文件路径、平移向量和四元数表示的旋转信息。
2. **四元数转换为旋转矩阵**：将四元数表示的旋转信息转换为旋转矩阵。利用数学公式，将四元数转换为 3x3 的旋转矩阵，方便后续计算。

3. **计算姿态矩阵**：结合平移向量，将旋转矩阵和平移向量组合成 4x4 的姿态矩阵。姿态矩阵包含了从世界坐标系到相机坐标系的变换信息。
4. **保存处理后的图像和姿态数据**：按照每个 scene 包含 7 张图片的方式，将处理后的图像和姿态数据保存到指定的目录中。确保每个 scene 的图像、深度图和姿态数据能够对应，便于后续的深度估计和三角化计算。

TUM (rgb\_d\_dataset\_freiburg1\_xyz)数据集格式

```
.
├─ accelerometer.txt
├─ associate.txt
├─ depth
│   ├── 1305031102.160407.png
│   └─ ...
├─ 1305031128.754646.png
├─ depth.txt
├─ groundtruth.txt
├─ rgb
│   ├── 1305031102.175304.png
│   └─ ...
├─ 1305031128.747363.png
└─ rgb.txt
```

经过处理之后，保存形式如下图：

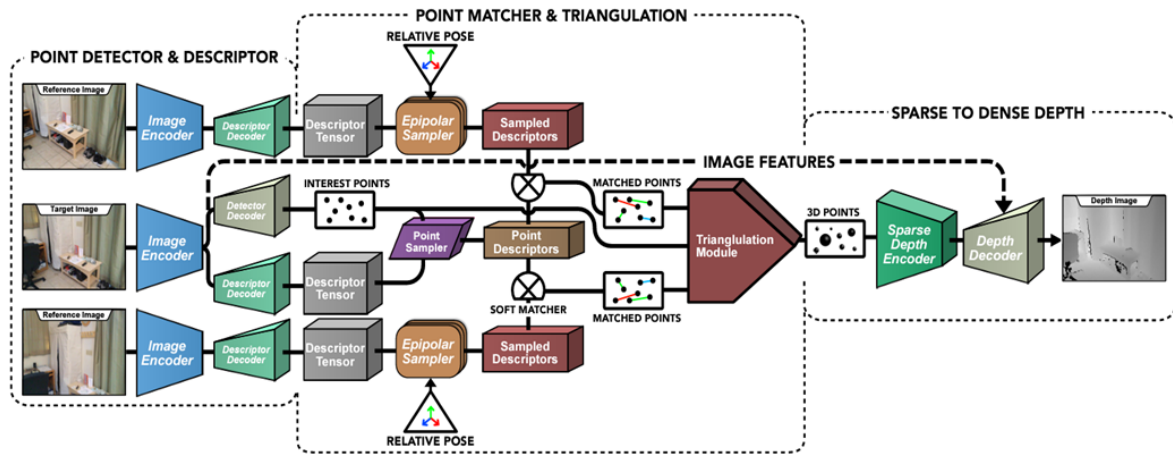


## 网络推理

DELTAS 论文提出了一种有效估计深度的方法，通过学习三角测量和稀疏点的密集化。该方法包括三个主要步骤：

1. **超点检测**：使用 SuperPoint 模型检测图像中的关键点和描述子。
2. **三角化**：使用图像对之间的相对运动和关键点进行三角化。

### 3. 密集化：使用深度学习方法对稀疏的三角化结果进行密集化。



图：DELTAS 网络架构图

该方法通过感兴趣点检测和描述符学习来补充姿态估计。网络由一个编码器-解码器结构组成，其中包括两个编码器（RGB 图像和稀疏深度图像）和三个解码器（感兴趣点检测、描述符和密集深度预测）。此外，引入了一个可微分模块，用于使用几何先验高效地三角化点，将感兴趣点解码器、描述符解码器和稀疏深度编码器连接起来进行端到端训练。

以下是实现该方法的详细步骤：

1. **修改基础信息配置文件**：根据实验需求修改配置文件，确保数据路径和模型参数正确。
2. **修改depth尺度信息**：根据数据集提供的深度信息，调整模型的 depth 尺度。
3. **执行推理代码**：运行推理代码，生成稠密的深度图。

```
python test_learnabledepth.py
```

```
Python > DELTAS > assets > sample_data > scannet_sample > scans_test > scene0707_00 > scene
1  colorHeight = 480
2  colorToDepthExtrinsics = 1.000000 0.000000 0.000000 0.000000 1.000
3  colorWidth = 640
4  depthHeight = 480
5  depthWidth = 640
6  fx_color = 517.3
7  fx_depth = 517.3
8  fy_color = 516.5
9  fy_depth = 516.5
10 mx_color = 318.6
11 mx_depth = 318.6
12 my_color = 255.3
13 my_depth = 255.3
14 numColorFrames = 5
15 numDepthFrames = 5
16
```

# 实验

## 评价指标

在本次实验中，使用以下三个指标来评估深度估计的性能：

**绝对误差 (Abs)：**绝对误差衡量估计深度与真实深度之间的平均差异，计算公式为：

$$\text{Abs} = \frac{1}{N} \sum_{i=1}^N |d_i - \hat{d}_i|$$

其中， $d_i$  是第  $i$  个像素的真实深度， $\hat{d}_i$  是第  $i$  个像素的估计深度， $N$  是像素总数。

**均方根误差 (RMSE)：**均方根误差衡量估计深度与真实深度之间的平方差的均值，计算公式为：

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (d_i - \hat{d}_i)^2}$$

**对数均方根误差 (RMSE log)：**对数均方根误差衡量估计深度与真实深度之间的对数差的平方均值，计算公式为：

$$\text{RMSE log} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(d_i + 1) - \log(\hat{d}_i + 1))^2}$$

其中，对数操作可以减小深度值较大时的误差影响，+1 是为了避免对零值取对数。

## 必做实验

	Abs	RMSE	RMSE log
ORB	13.81	14.37	2.59
SIFT	13.25	13.99	2.53
SURF	13.44	14.60	2.57

表：Portland\_hotel 数据集 5000 张图片对平均结果

结果显示，SIFT 在所有误差指标上表现最佳，其次是 SURF 和 ORB。值得注意的是，SURF 产生的特征点数量要多于 SIFT 和 ORB，但其深度估计误差略高于 SIFT。三者在特征点匹配上三者效果都差不多，但在深度估计方面三者的效果其实都不理想。

## 拓展实验

	2 Frames	3 Frames	4 Frames	5 Frames
Abs	0.250	0.226	0.194	0.157
RMSE	0.233	0.210	0.187	0.154
RMSE log	0.119	0.108	0.095	0.080

表：TUM 数据集不同 sequences 长度平均结果

从结果可以看出，随着额外视图的增加，深度估计的误差在所有指标上均稳步降低。这表明在多视图条件下，点匹配器和三角化模块能够从更多视图中受益，从而提高深度估计的精度。特别是在使用5帧图像进行估计时，误差显著降低，显示了多视图方法在提高深度估计精度方面的潜力。



## 总结

---

在本实验中，通过对 Portland\_hotel 数据集和 TUM 数据集的深度估计研究，验证了传统三角化方法和学习三角化方法在不同条件下的性能表现。实验结果表明，传统的 SIFT 特征检测器在深度估计中表现最佳，而 SURF 和 ORB 在特征匹配上表现良好但在深度估计上效果较差。通过复现 DELTAS 论文中的方法，发现多视图条件下的深度估计能够显著提高精度，特别是在使用更多视图时，误差显著降低。这表明，结合深度学习和多视图信息的方法在 SLAM 系统中具有广阔的应用前景。未来工作中，可以进一步优化特征检测和匹配算法，结合更多的视图和深度学习方法，进一步提高深度估计的准确性和鲁棒性。

## 参考文献

---

<https://cloud.tencent.com/developer/article/1852832>

<https://github.com/magicleap/DELTAS>

[https://cvg.cit.tum.de/data/datasets/rgbd-dataset/file\\_formats](https://cvg.cit.tum.de/data/datasets/rgbd-dataset/file_formats)

Ayan Sinha, Zak Murez, James Bartolozzi, Vijay Badrinarayanan, and Andrew Rabinovich. DELTAS: Depth Estimation by Learning Triangulation and Densification of Sparse Points. In ECCV, 2020.