

双目视觉里程计

张吉祥

2018 年 5 月 17 日

目录

1 裁剪方案	1
2 地图元素	2
3 词袋模型	3
4 跟踪线程 (前端)	3
4.1 特征提取	4
4.2 双目匹配	4
4.3 姿态预估计	4
4.3.1 跟踪上一帧	4
4.3.2 重定位	5
4.4 跟踪局部地图	5
4.5 确定关键帧	6
5 建图线程 (后端)	6
5.1 插入关键帧	6
5.2 地图点筛选	6
5.3 地图点生成	7
5.4 局部 BA	7
5.5 关键帧筛选	8
6 结果分析	8
6.1 轨迹进度	8
6.1.1 Absolute Trajectory Error	8
6.1.2 Relative Pose Error	9
6.2 运行时间	10
7 改进方向	10

摘要

视觉里程计通过处理一帧帧图像信息来估计相机的运动姿态，广泛运用在 AR、机器人、无人驾驶等领域。双目视觉里程计无需单目视觉里程计的初始化过程，不存在尺度不确定性的问题，应用场景更广。文中首先简述了如何通过**裁剪 ORB-SLAM2** 实现双目视觉里程计的方法；然后结合程序详细分析双目视觉里程计的实现原理；最后讨论实验结果，并提出改进方向。

关键词：双目视觉里程计；词袋模型；同时定位与地图构建

1 裁剪方案

考虑实际需求，双目视觉里程计需含跟踪和局部建图功能。ORB-SLAM2 组成模块为跟踪、局部建图和回环检测，并支持单目、双目和 RGB-D 相机。根据题目要求，我们需要对 ORB-SLAM2 进行裁剪，删除 ORB-SLAM2 的回环检测模块、与单目和 RGB-D 的相关函数等¹，只保留实现双目视觉里程计的跟踪和局部建图的功能，系统框图见图 1。

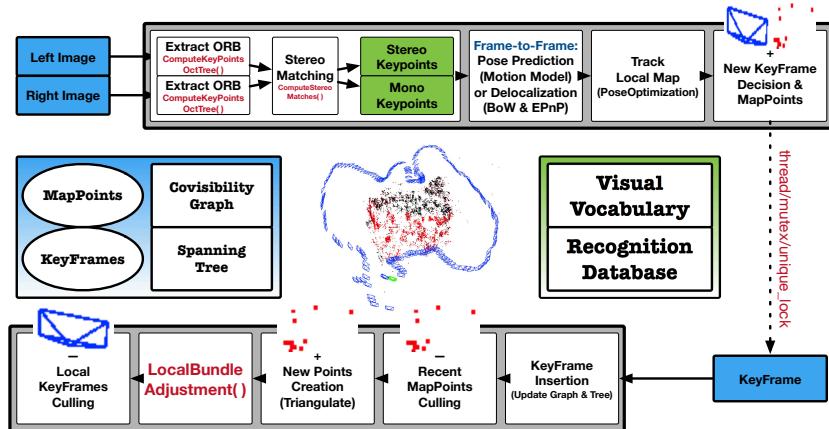


图 1: 双目视觉里程计系统框图

双目视觉里程计采用跟踪和局部建图两线程的方式，保证前端（跟踪）的实时性和控制后端（局部建图）优化的规模。同时，保留了 ORB-SLAM2

¹删除内容：Initializer.h/.c, LoopClosing.h/.c, Sim3Solver.h/.c 等文件；针对单目、RGB-D 与回环的相关函数。

的基于关键帧的地图结构，用来减少误差积累，并给后端优化提供了所需数据。为了提高系统的鲁棒性，保留了词袋模型，BoW 在双目里程计中主要用于重定位。

2 地图元素

采用 `set` 数据结构来存储地图点和关键帧：

```
1 std :: set<MapPoint*> mspMapPoints;  
2 std :: set<KeyFrame*> mspKeyFrames;
```

地图点 存储的部分信息：

- 在世界坐标系下的三维坐标：

```
1 cv :: Mat mWorldPos;
```

- 平均观测方向：

```
1 cv :: Mat mNormalVector;
```

- ORB 描述子：

```
1 cv :: Mat mDescriptor;
```

关键帧 存储的部分信息：

- 相机姿态 T_{CW}

```
1 cv :: Mat Tcw;
```

- 相机内参

- ORB 特征

共视图 关键帧之间的共视信息至关重要，用无向的加权图来表示。每个节点上有一个关键帧，两节点之间存在加权边的前提是，两节点的关键帧至少观测到 15 个相同的地图点。

生成树 共视图的一个子集 (共视地图点数量 ≥ 100)，用于维持关键帧的前后连接关系。

3 词袋模型

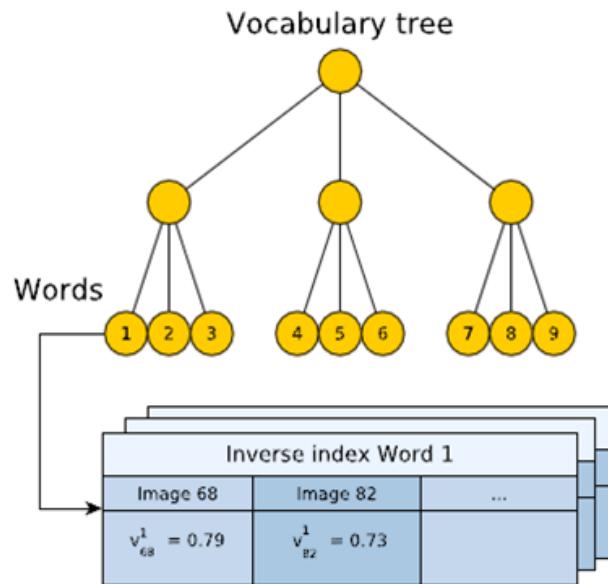


图 2: 词典树

词袋模型 [1](图 2) 在双目视觉里程计中的作用是：

- 跟踪失败时采取**重定位**，重新估计位姿：

```
1 std :: vector<KeyFrame*> DetectRelocalizationCandidates(Frame* F);
```

- 加快特征匹配速度：

```
1 int SearchByBoW();
```

4 跟踪线程 (前端)

前端的功能：计算姿态和确定关键帧 [2]。

4.1 特征提取

首先根据相机的图像构造缩放系数为 1.2 的 8 层图像金字塔。然后对每一层提取特征点，为使得特征点分布更加均匀，采用八叉树和自适应阈值的技巧提取 ORB 特征 [3]：

```
1 // 构建图像金字塔  
2 ComputePyramid( image );  
3 // 计算每层图像的兴趣点  
4 vector < vector<KeyPoint> > allKeypoints;  
5 ComputeKeyPointsOctTree( allKeypoints );
```

4.2 双目匹配

双目立体匹配函数：

```
1 void Frame::ComputeStereoMatches();
```

及其算法流程：

1. 为左目每个特征点建立带状区域搜索表，限定搜索区域
2. 通过描述子进行特征点匹配，得到每个特征点最佳匹配点 **scaleduR0**
3. 通过 SAD 滑窗得到匹配修正量 **bestincR**
4. 拟合抛物线，得到亚像素修正量 **deltaR**
5. 最终匹配点位置： **scaleduR0 + bestincR + deltaR**

4.3 姿态预估计

姿态预估计采用了跟踪上一帧或重定位的方式。

4.3.1 跟踪上一帧

如果成功跟踪上一帧，则采用匀速运动模型：

```
1 TrackWithMotionModel();
```

如果匀速运动模型失效，则跟踪参考值(最近的关键帧):

```
1 TrackReferenceKeyFrame();
```

4.3.2 重定位

假如当前帧与最近邻关键帧的匹配也失败了，意味着此时当前帧已经丢了，无法确定其真实位置。此时，只有去和所有关键帧匹配，看能否找到合适的位置。首先，计算当前帧的 BoW；然后，利用 BoW 词典选取若干关键帧作为备选，寻找有足够多的特征点匹配的关键帧；最后，利用 PnP 求解位姿。如果有关键帧有足够多的内点，那么采用该关键帧优化出的位姿 [4]：

```
1 Relocalization();
```

4.4 跟踪局部地图

跟踪局部地图的任务是最小化重投影误差来优化当前帧的位姿：

$$\{R_{cw}, cPw\} = \underset{R_{cw}, cPw}{\operatorname{argmin}} \sum_j \rho(\|x_c^j - \pi_m(R_{cw}X_W^j + cPw)\|_{\Sigma_j}^2) \quad (1)$$

```
1 // 步骤1：更新局部关键帧和局部地图点  
2 UpdateLocalMap();  
3  
4 // 步骤2：在局部地图中查找与当前帧匹配的 MapPoints  
5 SearchLocalPoints();  
6  
7 // 步骤3：更新局部所有 MapPoints 后对位姿再次优化  
8 Optimizer::PoseOptimization(&mCurrentFrame);
```

4.5 确定关键帧

当前帧想成为新的关键帧需要满足以下条件：

1. 长时间没有插入关键帧
2. 建图线程处于空闲状态
3. 跟踪效果较差即将失败
4. 当前帧跟踪的地图点与参考帧（最近的关键帧）的重复度不高

检测并插入关键帧，对于双目会产生新的地图点。

```
1 if (NeedNewKeyFrame ())  
2     CreateNewKeyFrame ();
```

5 建图线程（后端）

后端的功能：用关键帧优化局部地图。

5.1 插入关键帧

插入新的关键帧，紧接着需要更新共视图和生成树，并计算关键帧的BoW，为三角化生成新的地图点作准备：

```
1 ProcessNewKeyFrame ();
```

5.2 地图点筛选

采用宽进严出的原则，淘汰一些地图点：

```
1 MapPointCulling ();
```

每个地图点需要满足：

1. 跟踪到地图点的帧数相比预计可观测到该地图点的帧数的比例需大于25%

- 从地图点建立开始，到目前已经过了不小于 2 个关键帧，但是观测到该点的关键帧数却不超过阈值帧数，那么该点不合格

5.3 地图点生成

运动过程中和共视程度较高的关键帧通过三角化生成新的地图点：

1 CreateNewMapPoints();

检查并融合当前关键帧与相邻帧（两级相邻）重复的地图点，并更新共视图：

1 SearchInNeighbors();

5.4 局部 BA

构造图优化模型：

- Vertex
 - g2o::VertexSE3Expmap(), **LocalKeyFrames**, 即当前关键帧的位姿、与当前关键帧相连的关键帧的位姿
 - g2o::VertexSE3Expmap(), FixedCameras, 即能观测到 LocalMapPoints 的关键帧（但不属于 LocalKeyFrames）的位姿，在优化中这些关键帧的位姿不变
 - g2o::VertexSBAPointXYZ(), **LocalMapPoints**, 即 LocalKeyFrames 能观测到的所有 MapPoints 的位置
- Edge
 - g2o::EdgeSE3ProjectXYZ(), BaseBinaryEdge
 - * Vertex: 关键帧的 T_{cw} , MapPoint 的 P_w
 - * measurement: MapPoint 在关键帧中的二维位置 (u, v)
 - * InfoMatrix: invSigma2 (与特征点所在的尺度有关)
 - g2o::EdgeStereoSE3ProjectXYZ(), BaseBinaryEdge

- * Vertex: 关键帧的 Tcw, MapPoint 的 Pw
- * measurement: MapPoint 在关键帧中的二维位置 (ul,v,ur)
- * InfoMatrix: invSigma2(与特征点所在的尺度有关)

调用 **g2o** 实现对姿态和地图点的优化:

```
1 Optimizer :: LocalBundleAdjustment ( mpCurrentKeyFrame ,&mbAbortBA , mpMap )
```

5.5 关键帧筛选

采用**宽进严出**的原则，检测并剔除当前帧相邻的关键帧中冗余的关键帧，剔除的标准：**该关键帧的 90% 的 MapPoints 可以被其它至少 3 个关键帧观测到：**

```
1 KeyFrameCulling ();
```

6 结果分析

测试数据集 KITTI 07

硬件配置 MacBook Pro

- 处理器 2.7 GHz Intel Core i5
- 内存 8 GB 1867 MHz DDR3

6.1 轨迹进度

文中采用 **evo**² 来评估实验结果。

6.1.1 Absolute Trajectory Error

实验结果见图 3。

²<https://github.com/MichaelGrupp/evo>

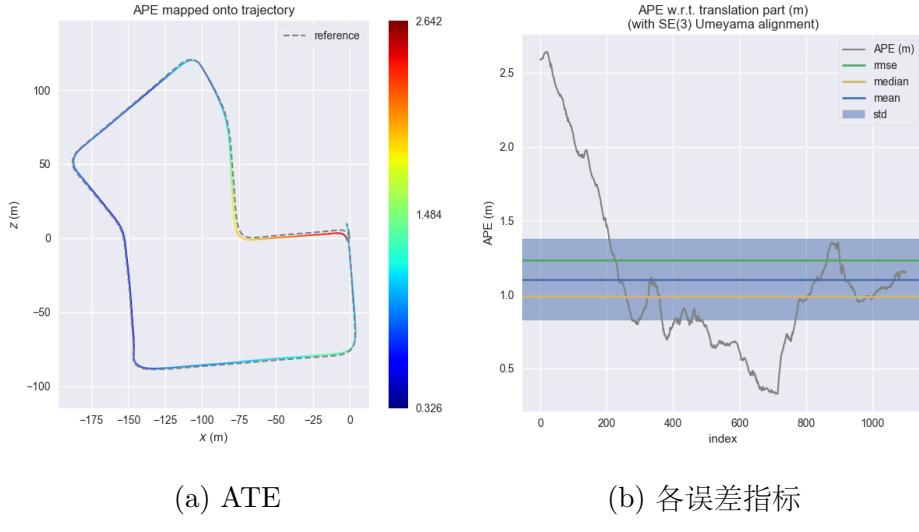


图 3: 绝对轨迹误差 (ATE)

6.1.2 Relative Pose Error

实验结果见图 4。

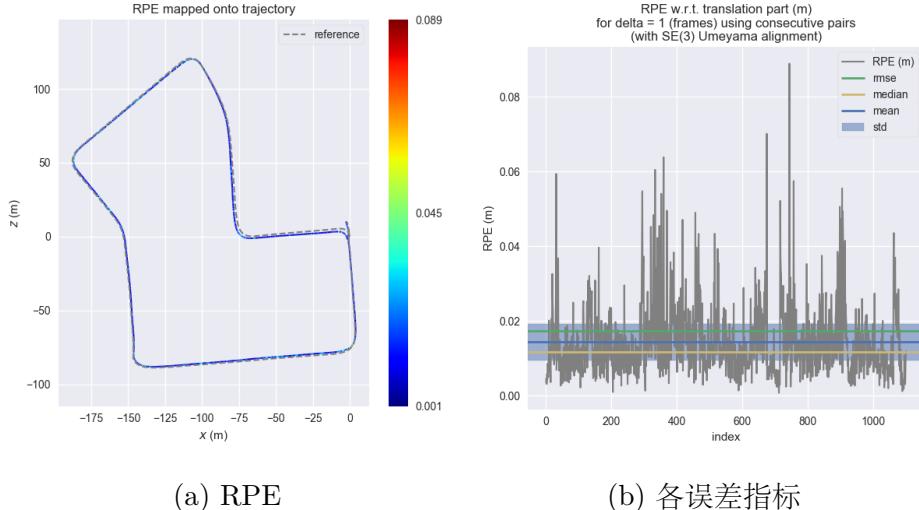


图 4: 相对姿态误差 (RPE)

6.2 运行时间

统计多线程程序的运行时间时,推荐采用 `chrono::steady_clock::now()`。

³

(KITTI 07)	时间 (ms)			平均时间
跟踪	24.1697	20.8701	18.0534	21.0311(ms)
局部建图	216.896	478.959	623.365	439.740(ms)

7 改进方向

1. 采用 GPU 或 FPGA 技术加速特征点提取和双目立体匹配;
2. 在双目视觉里程计的基础上融合 IMU 传感器信息, 提高估计的准确性;
3. 采用光流或者直接法, 克服了特征匹配耗时的缺点。
4. 使用二进制格式字典, 提高字典加载速度。

参考文献

- [1] D. Gálvez-López and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.
- [2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: a versatile and accurate monocular slam system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [3] “Orb-slam2 源码阅读 orb-slam2::orbextractor.” <https://www.cnblogs.com/JingeTU/p/6438968.html>.
- [4] “Orb-slam (四) 追踪.” <http://www.cnblogs.com/luyb/p/5357790.html>.

³如果采用 `clock()`, 则会出错。