深蓝学院
shenlanxueyuan.com

第一章作业分享

主讲人 梁谨栋

1 VIO文献阅读

深蓝学院 shenlanxueyuan.com

LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping

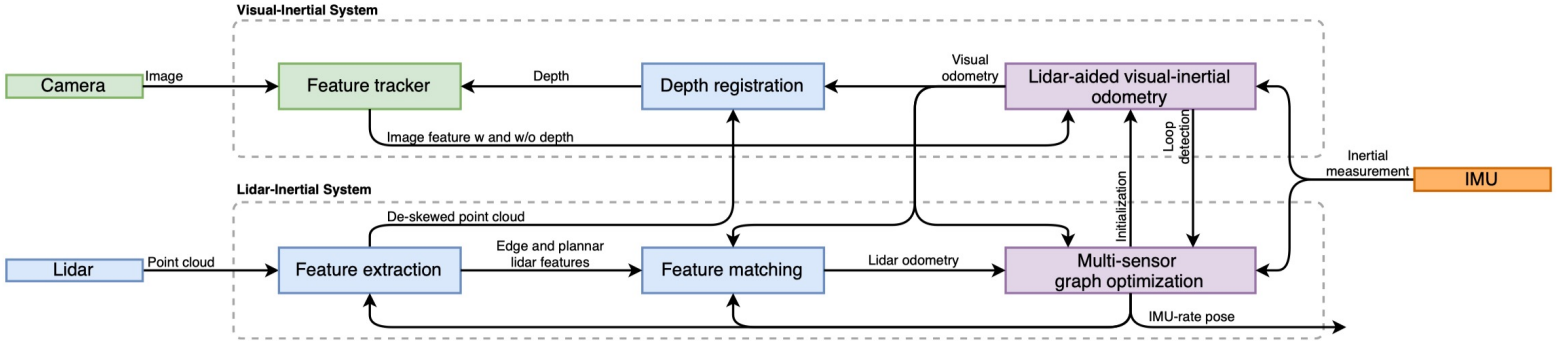


Fig. 1: The system structure of LVI-SAM. The system, which receives input from a 3D lidar, a camera and an IMU, can be divided into two sub-systems: a visual-inertial system (VIS) and a lidar-inertial system (LIS). The VIS and LIS can function independently while using information from each other to increase system accuracy and robustness. The system outputs pose estimates at the IMU rate.

主要贡献：

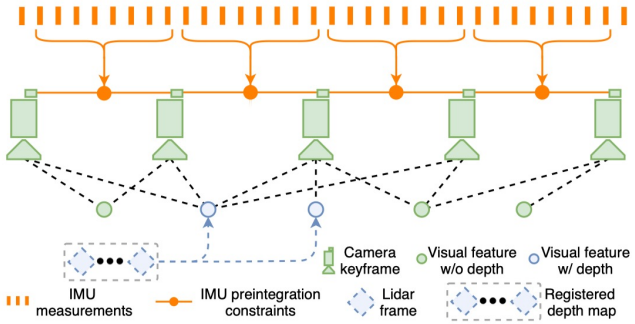
1.实现了一个激光-视觉-惯性的紧耦合系统，通过因子图同时实现多传感器融合，全局优化和回环检测

2.通过故障检测机制，避免单一子系统故障导致系统不能使用的情况，提高了整个系统的鲁棒性

1 VIO文献阅读

深蓝学院 shenlanxueyuan.com

LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping



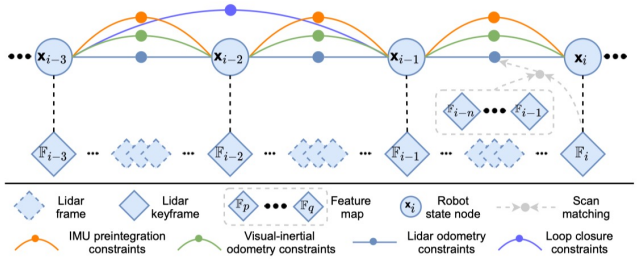


Fig. 5: The framework of our lidar-inertial system. The system maintains a factor graph that has four types of constraints.

[TixiaoShan/LVI-SAM: LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping (github.com)]

1 VIO文献阅读

R2LIVE: A Robust, Real-time, LiDAR-Inertial-Visual tightly-coupled state Estimator and mapping

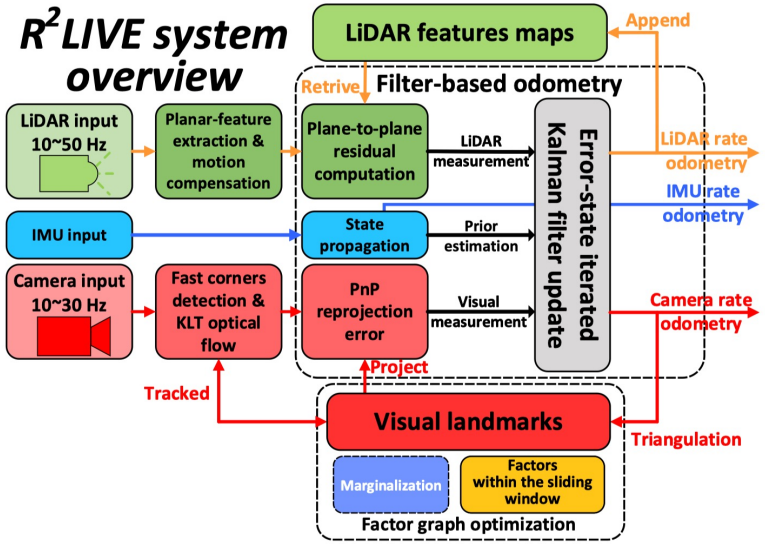


Fig. 2: The overview of our system.

该框架由两部分组成：
基于滤波器里程计和因子图优化。为了保证实时性能， 我们在误差状态迭代卡尔曼滤波器的框架内估计状态，并通过因子图优化进一步提高整体精度

1 VIO文献阅读

R2LIVE: A Robust, Real-time, LiDAR-Inertial-Visual tightly-coupled state Estimator and mapping

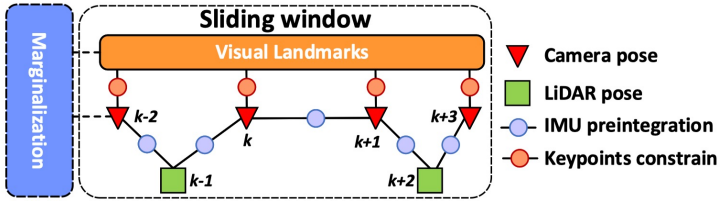


Fig. 5: Our factor graph optimization is a windowed visual bundle adjustment with LiDAR poses constraints.

为了进一步改善视觉测量，利用因子图优化来优化局部滑动窗口内的视觉landmarks.

比起VINS-Mono中的因子图,该因子图进一步包括了由激光雷达测量而造成的姿态约束,但是为了保持后端优化的轻量级,激光雷达的姿态只作为约束项,不对其进行优化.

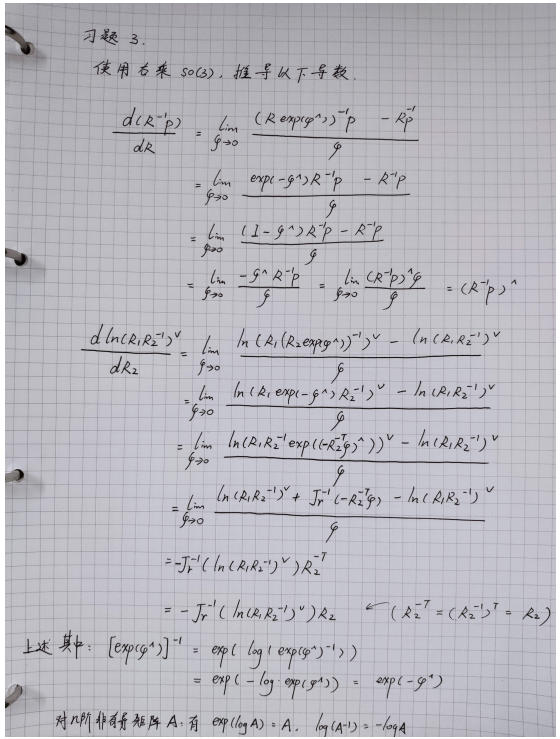
hku-mars/r2live: R2LIVE is a robust, real-time tightly-coupled multi-sensor fusion framework, which fuses the measurement from the LiDAR, inertial sensor, visual camera to achieve robust, accurate state estimation. (github.com)

2 四元数和李代数更新

深蓝学院
shenlanxueyuan.com

```cpp
int main()
{
    Vector3d w(0.01,0.02,0.03);

    Matrix3d R_I = Matrix3d::Identity();
    Sophus::SO3d SO3_R_I(R_I);               // 构建旋转矩阵并初始化为单位阵
    Quaterniond q_I = Quaterniond(R_I);      // 用旋转矩阵来构建四元数q


    //输出在更新之前的R和q
    cout << "before update ,R is : " << endl <<SO3_R_I.matrix() << endl;
    cout << endl;
    cout << "before update ,q is :" << q_I.coeffs().transpose() << endl;

    //update
    cout << endl<< "update ..." << endl;
    cout << endl;

    Sophus::SO3d SO3_updated = SO3_R_I * Sophus::SO3d::exp(w); //对R进行右乘更新
    cout << "updated, R is :" << endl << SO3_updated.matrix() << endl;
    cout << endl;

    Quaterniond q_w(1.0, 0.005, 0.01,0.015);  // 用四元数更新的公式来构建更新变量
    q_w.normalize();  // 要记得对其初始化为单位四元数

    q_I = q_I * q_w;
    cout << "updated , q is :" << q_I.coeffs().transpose() << endl;
    cout << endl;
    cout << "updated , from q to R is :" <<endl <<q_I.toRotationMatrix() << endl; //转化成旋转矩阵的形式方便对比

    return 0;
}
```

3 求导



主要注意两点：

1.右乘扰动模型和左乘扰动模型的区别

2.旋转矩阵的逆矩阵和其对应的李代数之间的关系

深蓝学院
shenlanxueyuan.com

感谢各位聆听

Thanks for Listening