Jindong_Shi
2021-12-06

第 10 节课习题

1. VIO 文献阅读

1) 视觉和 IMU 进行融合后，有何优势?

普通 VO 存在的缺点: 1. 单目视觉 SLAM 存在尺度的问题，单目 SLAM 处理的图像帧丢失了环境的深度信息，即使通过对极约束和三角化恢复了空间路标点的三维信息，但是这个过程的深度恢复的刻度是任意的，并不是实际的物理尺度，导致的结果就是单目 SLAM 估计出的运动轨迹即使形状吻合但是尺寸大小却不是实际轨迹尺寸。2. 由于基于视觉特征点进行三角化的精度和帧间位移是有关系的，当相机进行近似旋转运动的时候，三角化算法会退化导致特征点跟踪丢失。 3. 视觉 SLAM 一般采取第一帧作为世界坐标系，这样估计出的位姿是相对于第一帧图像的位姿，而不是相对于地球水平面 (世界坐标系) 的位姿，后者却是导航中真正需要的位姿，换言之，视觉方法估计的位姿不能和重力方向对齐。[3]

VIO 的优点:

优势 1: 通过引入 IMU 信息可以很好地解决 VO 的问题，首先通过将 IMU 估计的位姿序列和相机估计的位姿序列对齐可以估计出相机轨迹的真实尺度，而且 IMU 可以很好地预测出图像帧的位姿以及上一时刻特征点在下帧图像的位置，提高特征跟踪算法匹配速度和应对快速旋转的算法鲁棒性，最后 IMU 中加速度计提供的重力向量可以将估计的位置转为实际导航需要的世界坐标系中。[3]

优势 2: 相机和 IMU 融合有很好的互补性。首先通过将 IMU 估计的位姿序列和相机估计的位姿序列对齐可以估计出相机轨迹的真实尺度，而且 IMU 可以很好地预测出图像帧的位姿以及上一时刻特征点在下帧图像的位置，提高特征跟踪算法匹配速度和应对快速旋转的算法鲁棒性，最后 IMU 中加速度计提供的重力向量可以将估计的位置转为实际导航需要的世界坐标系中。[2]

优势 3: 随着 MEMS 器件的快速发展，智能手机等移动终端可以便捷地获取 IMU 数据和摄像头拍摄数据，融合 IMU 和视觉信息的 VINS 算法可以很大程度地提高单目 SLAM 算法性能，是一种低成本高性能的导航方案，在机器人、AR/VR 领域得到了很大的关注。[2]

With the increased interest in applying these techniques to small-sized platforms, such as small-sized unmanned aerial vehicles (UAVs) or handholdmobile devices, the research focus of localisation and mapping is shifted towards the use of cameras and inertial measurement unit (IMU) sensor. These sensors are made available nowadays with high accuracy, miniaturised size, and low cost because of the fast-developing manufacturing of chips and microelectromechanical systems (MEMS) devices. And they are complimentary with one another in a way which would be able to compensate for the errors made by each of them via the redundant information they provided.[1]

2) 有哪些常见的视觉+IMU 融合方案，有没有工业界应用的例子?

视觉惯性里程计根据融合框架的不同分为松耦合和紧耦合。松耦合中视觉运动估计和惯导运动估计系统是两个独立的模块，将每个模块的输出结果进行融合,但也因此只能达到局部最优而无法达到全局最优的位姿。紧耦合则是使用两个传感器的原始数据共同估计一组变量，传感器噪声也是相互影响的。紧耦合算法比较复杂，但充分利用了传感器数据，可以实现更好的效果，是目前研究的重点。常见的 VIO 方案有 VINS-Mono, OKVIS, MSCKF 等。前两个是基于非线性优化的方案而且框架比较相似，后者是基于滤波优化的方案，也是 Google Tango 上使用的方法。值得注意的是，虽然在纯视觉 SLAM 中，学界已经公认基于非线性优化方法的 SLAM 方法效果要好于滤波的方法，但在 VIO 中，非线性优化和滤波方法目前还没有很明显的优劣之分。

工业界的应用主要集中在机器人，无人机和 AR/VR 领域。主要罗列一下 AR/VR 方向:

Apple 在 2017 年 推出的 ARKit 实现了在移动设备上的 AR 效果。当使用者通过移动设备向设备拍摄到的场景中添加虚拟信息时，为了实现很好的逼真效果，第一步就是需要固定该虚拟物体相对真实环境中的位置，即移动设备而视角发生变化时，虚拟物体和真实环境的相对位置不应该发生变化，而虚拟物体实际是存在于移动设备上的，换言之，我们需要精确估计出移动设备相对空间的位置变化。 ARKit 是通过 VIO 实现移动设备在空间中的精确定位。

VR(Virtual Reality) 虚拟现实技术则是通过虚拟现实头盔投射虚拟信息，给人身临其境的感觉，将使用者在实际空间中的移动反映到虚拟空间上可以很大程度地提高交互体验，通过追踪使用者佩戴头盔的位置可以实现这一效果，实现头盔空间定位的方式可以分为 Outside-in 方案和 Inside-out 方案，前者使用过外部辅助设备实现定位，后者则使用头盔自身传感器实现定位，显然后者的使用场景不受限制是一种更好的定位方式，而 Inside-out 定位目前被广泛采用的方案就是视觉惯性传感器融合实现 SLAM 的 VINS 算法。[3]

3) 在学术界 VIO 研究有哪些新进展? 有没有将学习方法用到 VIO 中的例子?

VIO 学术界研究新进展(2020 年以来)[4]:

1. The UMA-VI dataset: Visual–inertial odometry in low-textured and dynamic illumination environments
2. An Online Initialization and Self-Calibration Method for Stereo Visual-Inertial Odometry
3. Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry
4. Camera intrinsic parameters estimation by visual–inertial odometry for a mobile phone with application to assisted navigation
5. Information sparsification for visual-inertial odometry by manipulating Bayes tree
6. OrcVIO: Object residual constrained Visual-Inertial Odometry
7. Lightweight hybrid visual-inertial odometry with closed-form zero velocity update
8. DVIO: An optimization-based tightly coupled direct visual-inertial odometry
9. Leveraging planar regularities for point line visual-inertial odometry

使用学习方法的例子:

1. CodeVIO: Visual-Inertial Odometry with Learned Optimizable Dense Depth

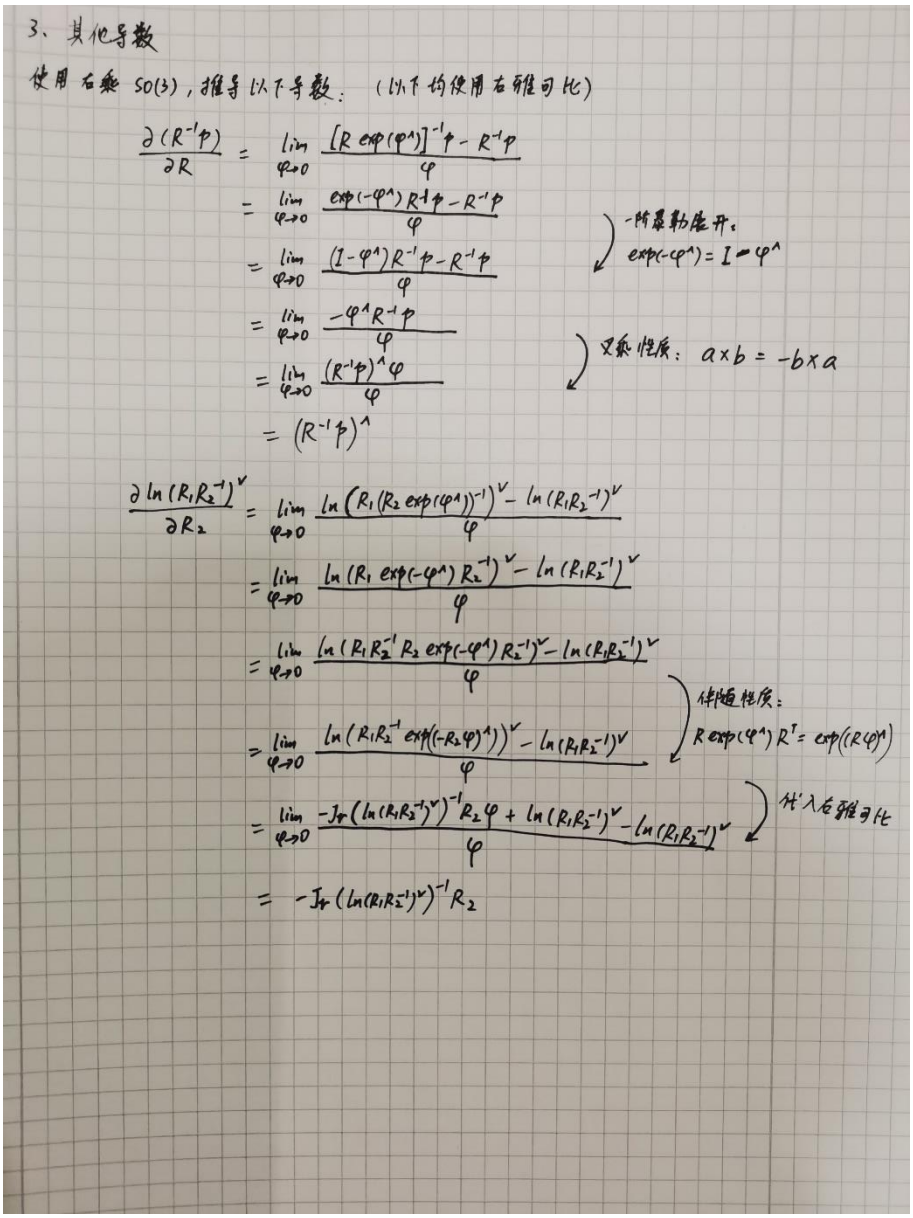
2. 四元数和李代数更新

源代码位于文件夹 L10_code 内。我们以围绕 z 轴旋转 90 度创建旋转矩阵 R 和对应的四元数 q，并且分别对它们右乘一个小量 $w = [0.01, 0.02, 0.03]^T$ 进行更新，两者更新结果相近，见下图。

```cpp
#include <sophus/so3.hpp>
#include <iostream>
#include <Eigen/Core>
#include <Eigen/Geometry>

int main()
{
    // create a rotation matrix R and corresponding quaternion q
    Eigen::Matrix3d R = Eigen::AngleAxisd( angle: M_PI/2,  axis: Eigen::Vector3d( x: 0, y: 0, z: 1)).toRotationMatrix();
    Eigen::Quaterniond q( other: R);

    Eigen::Vector3d w_so3( x: 0.01,  y: 0.02,  z: 0.03);

    Sophus::SO3d R_updated = Sophus::SO3d(R) * Sophus::SO3d::exp( omega: w_so3);
    Eigen::Quaterniond q_updated = q * Eigen::Quaterniond( w: 1,  x: 0.5 * w_so3( index: 0),  y: 0.5 * w_so3( index: 1),  z: 0.5 * w_so3( index: 2));

    std::cout << "R_updated = " << R_updated.matrix() << std::endl << std::endl
              << "q_updated = " << q_updated.matrix() << std::endl;

}
```

```
/home/jindong/VIO/L10/L10_code/OUTPUT/quaternion_lie_algebra_update
R_updated =  -0.030093    -0.9995  0.0096977
   0.99935  -0.029893  0.0201453
-0.0198454  0.0102976    0.99975

q_updated = -0.03045 -0.99985   0.0097
  0.9997 -0.03025  0.02015
-0.01985   0.0103  0.99975
```

3. 其他导数



参考文献:

[1] A review of visual inertial odometry from filtering and optimisation perspectives, Jianjun Gui*, Dongbing Gu, SenWang and Huosheng Hu.

[2] 计算机视觉方向简介 | 视觉惯性里程计(VIO)
https://tianchi.aliyun.com/forum/postDetail?postId=77482

[3] SLAM 中 VIO 的优势及入门姿势
https://mp.weixin.qq.com/s?spm=5176.21852664.0.0.7ede6e10IkFI6g&__biz=MzIxOTczOTM4NA==&mid=2247487473&idx=1&sn=4c1b2cef84bbb4e04f12c1190aef2eff&chksm=97d7ea66a0a063703c857483cb501b911d9cafc43583ec2c901153f39853818d385d2616cb1c&scene=21#wechat_redirect

[4] https://scholar.google.com/