

第一讲-习题

姜帆

2019 年 6 月 16 日

目录

1	文献阅读题	2
1.1	视觉与 IMU 进行融合之后有何优势?	2
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?	2
1.3	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?	4
2	编程验证题	5
2.1	代码	5
2.2	运行结果截图	6
3	公式推导题	7
3.1	第一小题	7
3.2	第二小题	7

1 文献阅读题

1.1 视觉与 IMU 进行融合之后有何优势？

视觉和 IMU 进行融合符合了当前多传感器融合的一个趋势，实际场景非常复杂，单一传感器不一定适用于所有场景，所以需要多传感器来实现互补，来达到尽可能适用于各类场景的效果。比如最常见的有 GPS-IMU 进行组合导航，激光雷达 +IMU 实现恶劣天气的定位。

从视觉上来说，存在以下问题：1. 成像容易受到环境干扰，比如场景中动态物体的存在，天气影响，场景纹理单一、缺失，光照变化；2. 单目情况下无法估计测量尺度；3. 当运动较快时由于运动模糊容易发生跟踪丢失的现象。4. 对于单目，存在初始化过程，对于双目，计算量较大，对于 RGBD 相机，测量范围有限，易受光照影响。当然，视觉也自身的优势：1. 可以获取场景纹理，利于后面的地图构建；2. 自身没有漂移。

对于 IMU，优势为：1. 相对视觉来说频率高，可以对快速运动作出及时反应。2. 可以估计得到绝对尺度信息。3. IMU 测量与外界环境没有关系，当相机在面对动态场景场景纹理缺失的情况时，IMU 仍然可以继续工作。缺点为：1. IMU 虽然可以测得角速度和加速度，但低廉的 IMU 这些量都存在明显的漂移，而精度较高的 IMU 则过于昂贵。2. 存在零漂。

根据上面所述视觉和 IMU 的优势与劣势，可以发现这两种传感器在某种程度上存在互补的特性。因此视觉 +IMU 的组合方案能够实现取长补短的效果。当视觉在快速运动，环境干扰情况下将要失效时，IMU 的快速相应，短时间内精度较高的优势能够解决视觉缺点；而相机数据可以有效地估计并修正 IMU 读数中的漂移，使得在慢速运动后的位姿估计依然有效。

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 +IMU 融合可以分为松耦合和紧耦合两种。

1. 松耦合是指视觉和 IMU 分别单独估计当前运动状态，然后对两个状态估计结果进行融合的过程。一般使用卡尔曼滤波进行融合。

2. 紧耦合是指将图像特征与 IMU 预积分得到的变量一起构建运动与观测方程，进行运动状态的求解的过程。一般有基于滤波的方法和基于优化的方法。基于滤波的方法主要有 MSCKF (Multi-State Constraint KF)。基于优化的方法主要有 vins 为代表。其中关键的技术主要有滑动窗口 (slide window) 优化，利用舒尔补进行边缘化等。

尽管在纯视觉中，基于优化的方案已经基本取代了基于滤波的方法，但是在 vio 中由于 IMU 的数据频率非常高，对状态进行优化需要的计算量就更大，非线性优化相比于基于滤波的方法并没有展现特别的优势，因此目前是两种方案并存的阶段。

具体的 VIO 框架有：MSCKF, OKVIS, ROVIO, VINS-Mono, SVO+MSF 等。MSCKF (Multi-State Constraint Kalman Filter) 是 2007 年提出的，是较为经典的基于滤波作为后端的 VIO 算法，前段上 MSCKF 采用的是光流跟踪特征点的方法，特征点使用的是 FAST 特征；OKVIS 采用紧耦合方案，使用非线性优化作为后端，使用 ceres 框架，前端使用 Harris 角点，BRISK descriptors；ROVIO 同样采用紧耦合方案，后端使用扩展 Kalman 滤波，前端使用 Fast 角点，优化光度误差；VINS-Monos 整体采用紧耦合方案，使用基于滑动窗口的非线性优化作为后端，具体采用 ceres 框架但是在视觉和 IMU 初始化时采用的是松耦合方案，前端使用 Harris 特征并利用光流进行跟踪；此外 MSF 是一种多传感器融合的松耦合方案，可以将 svo 与 imu 融合，实现 VIO 的作用。在众多 VIO 方案中，带闭环的 VINS

是精度最高的方案。图 1 为各个 VIO 方案对比图。

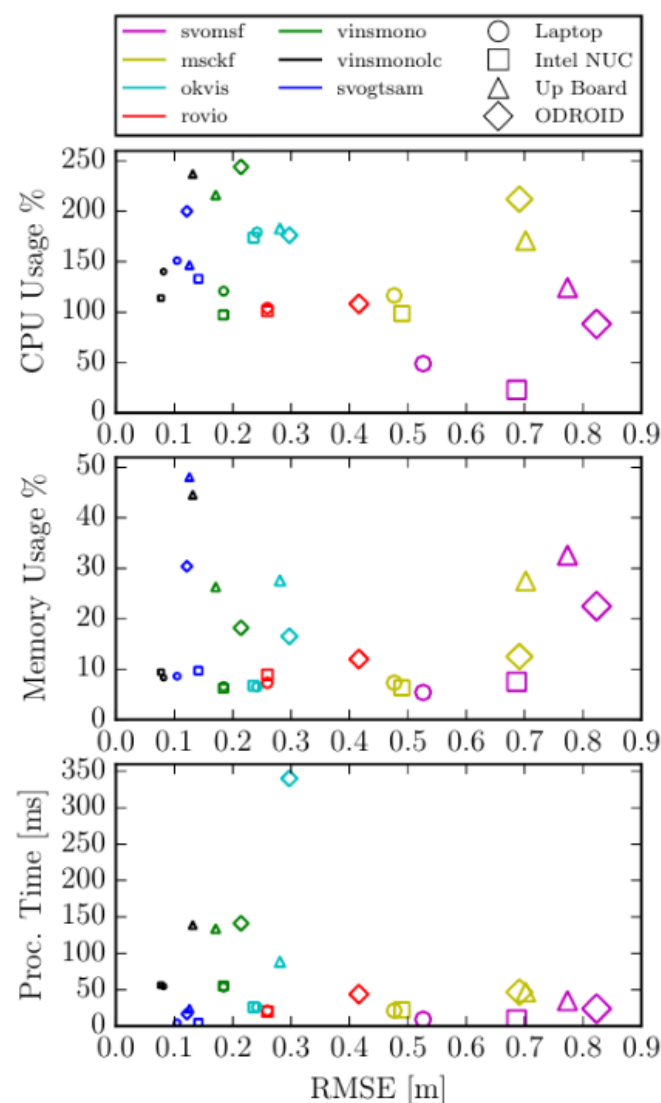


图 1: VIO 方案对比

工业界，高巨创新 MARK 无人机采用的 VIO（视觉惯性里程计）技术，革新定位方式，通过机身前端向下倾斜 45° 的鱼眼广角镜头与机身内部 IMU 结合实现精准定位，可在室内无 GPS 的环境中飞行，实现 MARK 室内外来回自由穿梭，并且悬停更稳定。飞行器通过图像比对进行定位导航，无需校准磁场便可直接起飞，也不受复杂磁场环境干扰，自动对下方地面障碍物或积水等，降低定位误差，大幅提升降落精度。



图 2: VIO 工业应用

1.3 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 预积分技术极大推动了 VIO 的发展进程。IMU 的数据通过积分，可以获取当前位姿（ p 位置， q 四元数表达的姿态）、瞬时速度等参数。在 VINS 中便使用了预积分技术。预积分是 Christian Forster (SVO 作者) 在 15 年做的工作，发表在 2015 年的 RSS (Robotics: Science and Systems) 上。预积分给出了一种比较优雅的，在 $SO(3)$ 流形上处理 IMU 运动的方法，并给出了实现代码 (gtsam 中)。因此，了解预积分，会对 VIO 的研究有重要的意义。

首次使用 DL 的框架解决 VIO 问题是 VINet。传统 VIO 的框架一般可分为三个过程，基于图像序列的光流估计、基于惯性数据的积分操作以及基于滤波和优化的运动融合。这种网络使用 FlowNet 来建模视觉运动特征，用 LSTM 来建模 IMU 的运动特征，最后通过李群李代数中的 $SE(3)$ 流行来建模位姿，用帧间堆叠的 LSTM 网络来预测位姿。同时，VINet 在面对时间不同步和外参标定不准确的多视觉惯导数据时，表现出了一定的优势。

最新论文 Shamwell, E. Jared, et al. "Unsupervised Deep Visual-Inertial Odometry with Online Error Correction for RGB-D Imagery." IEEE transactions on pattern analysis and machine intelligence (2019). 学会了在没有惯性测量单元 (IMU) 内在参数或 IMU 和摄像机之间的外部校准的情况下执行视觉惯性里程计 (VIO)。

2 编程验证题

2.1 代码

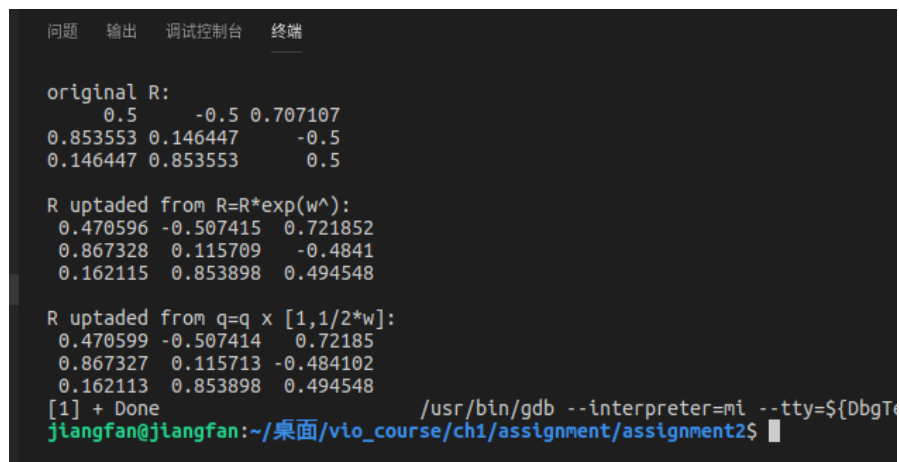
```

1  #include <iostream>
2  #include <Eigen/Dense>
3
4  using namespace std;
5  using namespace Eigen;
6
7  int main()
8  {
9      Eigen::Matrix3d R0=Eigen::AngleAxisd(M_PI/4,Eigen::
        Vector3d(1,0,0)).toRotationMatrix()
10         *Eigen::AngleAxisd(M_PI/4,Eigen::Vector3d
        (0,1,0)).toRotationMatrix()
11         *Eigen::AngleAxisd(M_PI/4,Eigen::Vector3d
        (0,0,1)).toRotationMatrix();
12     cout<<"original_R:"<<endl<<R0<<endl<<endl;
13     Eigen::Vector3d omiga;
14     omiga<<0.01,0.02,0.03;
15
16     // update rotation matrix
17     // compute exp(w_hat)
18     double theta=sqrt(omiga(0)*omiga(0)+omiga(1)*omiga(1)+
        omiga(2)*omiga(2));
19     Eigen::AngleAxisd w(theta,omiga/theta);
20     Eigen::Matrix3d deltaR=w.toRotationMatrix();
21     //cout<<"w_R:"<<deltaR<<endl;
22     Eigen::Matrix3d Rnew;
23     Rnew=R0*deltaR;
24     cout<<"R_updated_from_R=R*exp(w^):"<<endl<<Rnew<<endl;
25
26     cout<<endl;
27
28     // update quaternnion
29     Eigen::Quaterniond q(R0);
30     Eigen::Quaterniond deltaq(1,0.5*omiga(0),0.5*omiga(1)
        ,0.5*omiga(2));
31     //cout<<"deltaq:"<<deltaq.coeffs()<<endl;
32     deltaq=deltaq.normalized();
33     //cout<<"deltaq:"<<deltaq.coeffs()<<endl;
34     //cout<<"q_R:"<<deltaq.toRotationMatrix()<<endl;

```

```
35     Eigen::Quaterniond qnew;  
36     qnew=q*deltaq;  
37     qnew.normalized();  
38     cout<<"R uptaded from q=q x [1,1/2*w]:"<<endl<<qnew.  
        toRotationMatrix()<<endl;  
39  
40     return 0;  
41 }
```

2.2 运行结果截图



```
问题 输出 调试控制台 终端  
  
original R:  
0.5      -0.5 0.707107  
0.853553 0.146447 -0.5  
0.146447 0.853553 0.5  
  
R uptaded from R=R*exp(w^):  
0.470596 -0.507415 0.721852  
0.867328 0.115709 -0.4841  
0.162115 0.853898 0.494548  
  
R uptaded from q=q x [1,1/2*w]:  
0.470599 -0.507414 0.72185  
0.867327 0.115713 -0.484102  
0.162113 0.853898 0.494548  
[1] + Done /usr/bin/gdb --interpreter=mi --tty=${DbgTe  
jiangfan@jiangfan:~/桌面/vio_course/ch1/assignment/assignment2$
```

图 3: 运行结果截图

根据程序运行结果, 可以发现使用李代数更新旋转矩阵和使用四元数更新四元数这两种方法得到的结果是非常接近的。

3 公式推导题

3.1 第一小题

$$\begin{aligned}
\frac{d(R^{-1}p)}{dR} &= \lim_{\phi \rightarrow 0} \frac{(Exp(\phi^\wedge))^T p - R^T p}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{(exp(\phi^\wedge))^T R^T p - R^T p}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{(I + \phi^\wedge)^T R^T p - R^T p}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{(\phi^\wedge)^T R^T p}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{(-\phi)^\wedge R^T p}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{(-R^T p)^\wedge (-\phi)}{\phi} \\
&= (R^T p)^\wedge
\end{aligned} \tag{1}$$

3.2 第二小题

$$\begin{aligned}
\frac{d \ln(R_1 R_2^{-1})}{dR_2} &= \lim_{\phi \rightarrow 0} \frac{\ln(R_1 R_2 exp(\phi^\wedge)^T) - \ln(R_1 R_2^T)}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{\ln(R_1 (exp(\phi^\wedge))^T R_2^T) - \ln(R_1 R_2^T)}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{\ln(R_1 R_2^T R_2 (exp(\phi^\wedge))^T R_2^T) - \ln(R_1 R_2^T)}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{\ln(R_1 R_2^T (R_2 exp(\phi^\wedge) R_2^T)^T) - \ln(R_1 R_2^T)}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{\ln(R_1 R_2^T (exp((R_2 \phi)^\wedge))^T) - \ln(R_1 R_2^T)}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{\ln(R_1 R_2^T exp((-R_2 \phi)^\wedge)) - \ln(R_1 R_2^T)}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{\ln(R_1 R^T) + J_r^{-1}(-R_2 \phi) - \ln(R_1 R_2^T)}{\phi} \\
&= \lim_{\phi \rightarrow 0} \frac{-J_r^{-1}(R_2 \phi)}{\phi} \\
&= -J_r^{-1}((\ln(R_1 R_2^{-1}))^\vee) R_2
\end{aligned} \tag{2}$$

参考文献

- [1] Jianjun Gui, Dongbing Gu, Sen Wang , Huosheng Hu . A review of visual inertial odometry from filtering and optimisation perspectives[J], Advanced Robotics, 29:20, 1289-1301,2015.
- [2] J. Delmerico. A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots[J],ICRA. 2018.
- [3] Joan Solà.Quaternion kinematics for the error-state Kalman filter[M], 2017.
- [4] 高翔, 视觉 SLAM 十四讲 [M], 北京: 电子工业出版社,2017.
- [5] Shamwell, E. Jared, et al. "Unsupervised Deep Visual-Inertial Odometry with Online Error Correction for RGB-D Imagery." IEEE transactions on pattern analysis and machine intelligence (2019).