AS1

1. VIO 文献阅读

阅读 VIO 相关综述文献如[a]，回答以下问题：

- 视觉与 IMU 进行融合之后有何优势？
- 有哪些常见的视觉 +IMU 融合方案？有没有工业界应用的例子？
- 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？

你也可以对自己感兴趣的方向进行文献调研，阐述你的观点。

[a] Jianjun Gui et al. "A review of visual inertial odometry from filtering and optimisation perspectives". In: *Advanced Robotics* 29.20 (2015), 1289–1301. ISSN: 0169-1864. DOI: {10.1080/01691864.2015.1057616}.

https://raw.githubusercontent.com/MichaelBeechan/VO-SLAM-Review/master/A review of visual inertial odometry from filtering and optimisation perspectives.pdf

- 视觉与imu融合的优势: 互相取长补短

方案	IMU	视觉
优势	快速响应 不受成像质量影响 角速度普遍比较准确 可估计绝对尺度	不产生漂移 直接测量旋转与平移
劣势	存在零偏 低精度 IMU 积分位姿发散 高精度价格昂贵	受图像遮挡、运动物体干扰 单目视觉无法测量尺度 单目纯旋转运动无法估计 快速运动时易丢失

- 常见的视觉+imu融合方案,工业界例子

VINS : 紧耦合, Harris+光流+回环

OKVIS (单目+IMU、双目+IMU)

ORB_SLAM-IMU（单目+IMU）

MSCKF

工业界:

AR: 增强现实,在物理场景中加入虚拟信息,实现现实和虚拟的交互

VR: 虚拟现实, 通过虚拟现实头盔投射虚拟信息,将使用者现实的空间位移映射到虚拟空间,人机交互.

- 学术界中,VIO的新进展; 将学习方法用到VIO的例子.

新进展: 加入深度学习,比如在传统slam上加入语义信息,构建语义地图,或者用深度学习替换传统slam的某个模块

举例:

Selective Sensor Fusion for Neural Visual-Inertial Odometry.

端到端选择性传感器融合框架，该框架融合了单眼图像和惯性测量，以估计轨迹，同时提高对现实生活问题的鲁棒性，如应对丢失和损坏的数据或不良的传感器同步。

2. 四元数和李代数更新

课件提到了可以使用四元数或旋转矩阵存储旋转变量。当我们用计算出来的 ω 对某旋转更新时，有两种不同方式：

$$\begin{aligned} &\mathbf{R} \leftarrow \mathbf{R} \exp\left(\boldsymbol{\omega}^{\wedge}\right) \\ \text{或} \quad &\mathbf{q} \leftarrow \mathbf{q} \otimes \left[1, \frac{1}{2}\boldsymbol{\omega}\right]^{\top} \end{aligned} \tag{20}$$

请编程验证对于小量 $\boldsymbol{\omega} = [0.01, 0.02, 0.03]^{\mathrm{T}}$，两种方法得到的结果非常接近，实践当中可视为等同。因此，在后文提到旋转时，我们并不刻意区分旋转本身是 $\mathbf{q}$ 还是 $\mathbf{R}$，也不区分其更新方式为上式的哪一种。

```
#include <iostream>
#include <eigen3/Eigen/Core>
#include <eigen3/Eigen/Dense>
#include <unsupported/Eigen/MatrixFunctions>
#include <sophus/so3.hpp>

Eigen::Matrix3d toSkewMatrix(Eigen::Vector3d x){
    Eigen::Matrix3d res;
    res<< 0, -x[2], x[1],
          x[2], 0, -x[0],
          -x[1], x[0], 0;
    return res;
}

std::complex<double> expfn(std::complex<double> x, int)
{
```

```
    return std::exp(x);
}

int main(){
    Eigen::Vector3d omega = {0.01, 0.02, 0.03};
    Eigen::Vector3d v(1,2,3);
    Eigen::AngleAxisd angleaxis(M_PI/4, v/v.norm());
    Eigen::Matrix3d R = angleaxis.toRotationMatrix();
    std::cout << "R :"<< "\n" << R <<std::endl;
    Eigen::Quaterniond q(R);     //R->q
    std::cout << "q :"<< "\n" << q.coeffs().transpose() <<std::endl;
    Sophus::SO3d SO3_R(R);    //R->SO3
    std::cout << "so3 :"<< "\n" << SO3_R.log().transpose() <<std::endl;

    //calculate q'
    Eigen::Quaterniond qv = Eigen::Quaterniond(1, 0.005, 0.01, 0.015);
    Eigen::Quaterniond q_new = q * qv.normalized();
    std::cout << "q_new : " <<  q_new.coeffs().transpose() << std::endl;
    std::cout << "q_new to Matrix : " << "\n" << q_new.toRotationMatrix()
    <<std::endl;


    //calculate with Eigen
    Eigen::Matrix3d R_new = R * toSkewMatrix(omega).matrixFunction(expfn);
    std::cout << "R_new = R * toSkewMatrix(omega).matrixFunction(expfn): "
    << "\n" << R_new <<std::endl;

    //with Sophus
    Sophus::SO3d SO3_updated = SO3_R * Sophus::SO3d::exp(0.5 * omega);
    std::cout<<"SO3_new = "<< "\n" << SO3_updated.matrix() <<std::endl;

    std::cout<<"error (SO3_new & result with Eigen)= "<< "\n"
    <<SO3_updated.matrix()-q_new.toRotationMatrix()<<std::endl;

}
```

```
R :
  0.728028  -0.525105   0.440727
  0.608789   0.790791 -0.0634566
 -0.315202   0.314508   0.895395
q :
0.102276 0.204553 0.306829  0.92388
so3 :
0.209906 0.419813 0.629719
q_new : 0.106877 0.213754 0.320631  0.91656
q_new to Matrix :
  0.703009  -0.542065   0.460374
  0.633447   0.771546 -0.0588459
 -0.323301   0.332991   0.885773
R_new = R * toSkewMatrix(omega).matrixFunction(expfn):
  0.703006  -0.542067   0.460376
   0.63345   0.771543 -0.0588453
 -0.323302   0.332993   0.885772
SO3_new =
   0.71563  -0.533704   0.450593
  0.621203   0.781254 -0.0612367
 -0.319345   0.323732   0.890627
error (SO3_new & result with Eigen)=
  0.0126205  0.00836067 -0.00978062
 -0.0122439  0.00970809 -0.00239076
 0.00395576 -0.00925895  0.00485405
```

用Eigen计算出來的误差更小,用Sophus的大一点

3. 其他导数

使用右乘 $\mathfrak{so}(3)$，推导以下导数：

$$\frac{\mathrm{d}\left(\mathbf{R}^{-1}\mathbf{p}\right)}{\mathrm{d}\mathbf{R}} \tag{21}$$

$$\frac{\mathrm{d}\ln\left(\mathbf{R}_1\mathbf{R}_2^{-1}\right)^{\vee}}{\mathrm{d}\mathbf{R}_2} \tag{22}$$

李群性质:

$$C = \exp(\phi^\wedge) \equiv \sum_{n=0}^{\infty} \frac{1}{n!}(\phi^\wedge)^n$$
$$\equiv \cos\phi \cdot \mathbf{1} + (1-\cos\phi)\, a a^T + \sin\phi\, a^\wedge$$
$$\approx \mathbf{1} + \phi^\wedge$$

$$C^{-1} \equiv C^T \equiv \sum_{n=0}^{\infty} \frac{1}{n!}(-\phi^\wedge)^n \approx \mathbf{1} - \phi^\wedge$$

$$(Cu)^\wedge \equiv C u^\wedge C^T$$

$$\exp((Cu)^\wedge) \equiv C \exp(u^\wedge) C^T$$

$$\ln(\exp(\phi_1^\wedge)\exp(\phi_2^\wedge))^\vee \approx \begin{cases} J_l(\phi_2)^{-1}\phi_1 + \phi_2 & \text{当}\ \phi_1\ \text{为小量} \\ J_r(\phi_1)^{-1}\phi_2 + \phi_1 & \text{当}\ \phi_2\ \text{为小量.} \end{cases}$$

$$\frac{d(R^{-1}p)}{dR} = \lim_{\varphi \to 0} \frac{(R\exp(\varphi^\wedge))^{-1}p - R^{-1}p}{\varphi}$$
$$= \lim_{\varphi \to 0} \frac{(\exp(\varphi^\wedge))^{-1}R^{-1}p - R^{-1}p}{\varphi}$$
$$\approx \lim_{\varphi \to 0} \frac{(I - \varphi^\wedge)R^{-1}p - R^{-1}p}{\varphi}$$
$$= \lim_{\varphi \to 0} \frac{-\varphi^\wedge R^{-1}p}{\varphi}$$
$$= \lim_{\varphi \to 0} \frac{(R^{-1}p)^\wedge \varphi}{\varphi} = (R^{-1}p)^\wedge$$

$C^{-1} \equiv C^T \equiv \sum_{n=0}^{\infty} \frac{1}{n!}(-\phi^\wedge)^n \approx 1-\phi^\wedge$

$\exp(\phi^\wedge))^{-1} = \exp(-\phi^\wedge)$

转换 $\wedge$ 位置.

$$
\begin{aligned}
\frac{d\ln(R_1R_2^{-1})^\vee}{dR_2} &= \lim_{\phi\to 0}\frac{\ln(R_1(R_2\exp(\phi^\wedge))^{-1})^\vee-\ln(R_1R_2^{-1})^\vee}{\phi}\\
&= \lim_{\phi\to 0}\frac{\ln(R_1\underset{\exp(-\phi^\wedge)}{(\exp(\phi^\wedge))^{-1}}\underset{R_2^T}{R_2^{-1}})^\vee-\ln(R_1R_2^{-1})^\vee}{\phi}\\
&= \lim_{\phi\to 0}\frac{\ln(R_1R_2^{-1}R_2(\exp(\phi^\wedge))^{-1}R_2^{-1})^\vee-\ln(R_1R_2^{-1})^\vee}{\phi}\\
&= \lim_{\phi\to 0}\frac{\ln(R_1R_2^{-1}\cdot\exp((-R_2\phi)^\wedge))^\vee-\ln(R_1R_2^{-1})^\vee}{\phi}
\end{aligned}
$$

ㄱ 호.

$$
\begin{aligned}
&= \lim_{\phi\to 0}\frac{\ln(\exp((\ln(R_1R_2^{-1})^\vee)^\wedge)\cdot\exp((-R_2\phi)^\wedge))^\vee-\ln(R_1R_2^{-1})^\vee}{\phi}\\
&= \lim_{\phi\to 0}\frac{J_r(\ln(R_1R_2^{-1})^\vee)^{-1}\cdot(-R_2\phi)+\ln(R_1R_2^{-1})^\vee-\ln(R_1R_2^{-1})^\vee}{\phi}\\
&= \lim_{\phi\to 0}\frac{J_r(\ln(R_1R_2^{-1})^\vee)^{-1}\cdot(-R_2\phi)}{\phi}\\
&= \lim_{\phi\to 0} -J_r(\ln(R_1R_2^{-1})^\vee)^{-1}R_2
\end{aligned}
$$