一. VIO 文献阅读

阅读 VIO 相关文献综述，回答以下问题

1. 视觉与 IMU 进行融合之后有何优势?

   答:

   (a) 低成本 IMU 帧率快，适合短时间快速运动追踪，但是存在零漂，容易发散。视觉定位方法不会发生漂移，适合长时间慢速运动追踪，但是有时（速度过快、lowtexture env 或遮挡等）会丢失定位跟踪。两者优势互补。同时，IMU 的引入可以为单目相机提供尺度信息。

   (b)VIO 成本相对 LiDar 或 RGB-D 相机等传感器低很多，同时体积小，功耗低，可以用于手机或手持 AR 设备上。而 LiDar、RGB-D 相机等则无法满足实际使用需求。

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?

   答:

   (a) 常见的视觉 +IMU 融合方案如表 1 所示，其中的这些方法，每一个都有各自独特的优点，最早的 MSCKF 是 filter-based method, 随着研究的深入，大家发现 SLAM 中也可以应用优化方法，并且优化方法得到的效果比滤波的方法更好，因此得到了更加广泛的应用。而多传感器融合是一个好的趋势，因为不同传感器之间可以互相取长补短，因此许多优秀的 SLAM 框架下都可以加入 IMU，以增强系统的鲁棒性，下面的 SVO+MSF 以及 VIORB 就是这样的例子。

项目名称	耦合	前端	后端	误差定义	闭环	developer
MSCKF[1]	紧耦合	Fast 光流	EKF	重投影	no	UCR Mourikis 2007
ROVIO[2]	紧耦合	Fast 光流	IEKF	光度	no	ETHasl Bloesch
SVO+MSF	松耦合	Fast 光流	no	光度	no	U of Zurich Foster
VINS[3]	紧耦合	Harris 光流	优化	重投影	no	HKUST Shaojie
VIORB[4][5]	紧耦合	ORB 特征点	优化	重投影	yes	U of Zaragoza RaulMur
OKVIS[6]	紧耦合	Harris BRISK	优化	重投影	no	ETHasl Leutenegger

表 1: 常见视觉 +IMU 融合方案

   (b) 在工业界应用的例子如表 2 所示, 目前 VIO 主要应用场景是移动设备和 AR 应用 (因为 IMU + 视觉的优势就在于小型化、轻量化吧)，无人机上也有应用。自动驾驶

汽车上也有一些，但是对于自动驾驶，目前使用较多的是 LiDar，可能精度和可靠性更高。Tango 是 Google 的一个研发项目，其目的是将人类对周遭空间与动态的感知能力赋予移动设备，根据 19 年温焕宇的作业，里面采用的方法是 MSCKF。ARkit 是 Apple 推出的 AR 工具，主要应用在手机上，配合苹果公司广泛的手机市场占有率，如果好用的话，应该是爆炸式的。但是根据网上回答，经常跟丢，可能用用户毫无游戏体验，因此还有很多工作需要做。我想后面的 HoloLens 也是一样，因此不再进行详细的介绍。

感觉 Quora 上的一个回答对 VIO 落地情况分析得比较好，这里简单介绍一下他的观点：SLAM 目前刚刚走出实验室，而完成一个 VIO 项目 (在移动设备上)，有巨大的工作量，包括开发、测试、部署等。而世界上能够对于 VIO 技术非常熟悉的人目前还不多，如表 1 中所示，主要的优秀项目就这么几个，而每一个项目都是好几个博士才能完成的成果，一般的开发工作显然不能全部用博士，因此还没有太多成熟的工业界应用。美国作为灯塔国，Google、Apple 作为世界上最大的科技企业尚且没有成熟的产品落地，其他公司就更没办法、也没钱搞这样的项目了。最后，情况应该就是这么个情况，如果有没有考虑到的方面，还请助教老师指出，我还是比较想具体地了解行业发展以及技术落地之后的情况，谢谢。

项目名称	公司	采用方法	应用场景
Tango	Google	MSCKF	移动设备
ARkit	Apple	不详细	移动设备
HoloLens	MicroSoft	todo	移动设备

表 2: VIO 在工业界中的应用

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？

答：

主要进展分为几个方面来进行阐述，(a)VIO+ 其他传感器 (GPS、轮速计...); (b)deep VIO, 通过 deep learning 的方式，将传感器数据直接输出为里程计的轨迹推算; (c) 语义信息 or 环境理解;

(a)VIO+ 其他传感器，多传感器融合的趋势。特别在自动驾驶领域，我想要形成完善的自动驾驶解决方案，保证所有场景的安全行驶，多种传感器的融合以保证安全其安全性是技术发展的趋势。其中比较有代表性的工作举例如下：VIO 与 LiDar 相

结合 [7], 在 VINS-Mono 的基础上进行扩展 [8](我不知道这是否就是他们说的 VINS-Fusion)，VINS 与轮速计相结合 [9]，这种组合在 UGV 上应用更广泛，可以降低无人小车成本等。

(b)deep-VIO，实际上 Deep-SLAM 的工作有一些，比较早的应该是这个用 CNN 来做 motion estimation 的工作 [10]，以及后面的 VINet[11]、DeepVO[12] 等工作。

(c) 语义信息以及环境理解相关的 SLAM or VIO，其中语义 SLAM 比较早的提出应该是在一篇综述 past, present, and future of SLAM[13] 中提出，而 VIO 方面与语义 SLAM 相关的工作，比较著名的是 MIT 的 kimera 系列 [14]。

如上面所述，有将学习的方法应用到 VIO 中的，比如 (b) 中是直接端到端地估计机器人的位姿或运动。另一种趋势就是通过神经网络进行目标检测或者语义分割，并将检测到的物体作为路标 (landmark)，参与局部 BA 或者后端优化，又或者是闭环检测。

4. 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。

   我比较感兴趣的方向也是上面提到的，将语义信息加入到 SLAM 中。我想最成熟的机器人应该要能够理解环境，因为我们人类看任何东西都是有语义信息的。也是在语义信息的加持下，机器人才能够规划更高阶的任务。

二. 四元数和李代数的更新

题目：课件提到了可以使用四元数或者旋转矩阵储存旋转变量。当我们计算出来的 ω 对某旋转更新时，有两种不同的方式：

$$\begin{aligned} &1)\mathbf{R} \leftarrow \mathbf{R}exp(\omega^{\wedge}) \\ &2)\mathbf{q} \leftarrow \mathbf{q} \otimes [1, \frac{1}{2}\omega]^{\top} \end{aligned} \tag{1}$$

请编程验证对于小量 $\omega = [0.01, 0.02, 0.03]^{\top}$，两种方法得到的结果非常接近, 实践当中可以视为等同。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 $\mathbf{R}$，也不区分更新方式为上式的哪一种。

答：我写了两个文件，一个调用 Eigen 以及 sophus 等库，另一个只调用了 Eigen，其中四元数和李代数的更新我都按照罗德里格斯公式或者向量、矩阵实现了一遍。

只调用 Eigen 库的代码如下

1. main.cpp

```cpp
// 通过四元数和旋转矩阵两种不同的方式更新旋转
// 结果应该相差不大
#include <iostream>
#include "Eigen/Core"
#include "Eigen/Geometry"
#include "sophus/se3.hpp"
#include <math.h>
using namespace std;

/************************Eigen*******************
 * eigen 中各种旋转表示方式
 * 1. rotation matrix(3x3) -----> Eigen::Matrix3d
 * 2. 旋转向量(3x1) -----> Eigen::AngleAxis 每个轴，对应转多少度
 * 3. quterniond(4x1) ------------> Eigen::Qaterniond
 * 4. translation vector(3x1)-----> Eigen::Vector3d
 * 5. transformation matrix(4x4)---> Eigen::Isometry3d
*/

/************************ STEPS**************************
 * 1. 定义 q & R
 * 2. define rotaion vector w = [0.01, 0.02, 0.03]T
 * 3. 分别用两种方法更新旋转
 *        q = q * [1, 1/2 * w]T
 *        R = R * exp(w^)
 *        eigen中无法做矩阵乘法，或许用罗德里格斯公式
 *        exp(theta a) = cos(theta)I + (1 - cos(theta))aaT + sin(theta)a^
 * 4. 两种方法都改为旋转矩阵显示 / 或者用欧拉角显示？？
*/


int main(){
    Eigen::Matrix3d R;  // rotation matrix
```

```
33      R = Eigen::AngleAxisd(M_PI/4, Eigen::Vector3d(0,  0,  1)). toRotationMatrix ();
34      Eigen :: Quaterniond q_eigen = Eigen :: Quaterniond(R);    // quaterniond
35      double trace_R = R(0,0) + R(1,1) + R(2,2);
36      std :: cout << "the trace of R is " << trace_R << endl;
37      double w_quaterniond = sqrt(trace_R + 1) / 2;
38      double z = (R(1,0) - R(0,1)) / (4 * w_quaterniond);
39      Eigen :: Quaterniond q_handmade( w_quaterniond, 0, 0, z );
40      // 为什么构造函数和显示的时候顺序不一样
41      // 因为如果构造函数是Eigen::Quaterniond q(w, x, y, z) 顺序这样
42      // 如果构造函数是 Eigen::Quaterniond q(Eigen::vector4d(x, y, z, w)) 顺序这样
43
44      Sophus::SO3d SO3_R(R); //SO3李群，由旋转矩阵R构造
45
46      cout << "R is = " << endl << R << endl;
47      // 实部在后，虚部在前
48      cout << "q transformed by Eigen is = " << q_eigen.coeffs().transpose() << endl;
49      // 实部在后，虚部在前
50      cout << "q transformed by handmade is = "
51       << q_handmade.coeffs().transpose() << endl;
52      // 李群创建的旋转矩阵
53      cout << "R from SO3_R is = " << endl << SO3_R.matrix() << endl;
54
55      Eigen :: Vector3d so3 = SO3_R.log();
56
57      // 旋转向量微小扰动用w表示
58      Eigen :: Vector3d w(0.01,  0.02,  0.03);
59
60      // 向量的模长
61      double theta = sqrt(0.01*0.01 + 0.02*0.02 + 0.03*0.03);
62      std :: cout << "theta is : " << theta << endl;
63      // 向量的方向
64      Eigen :: Vector3d a(0.01/theta ,  0.02/theta ,  0.03/theta );
```

```
65
66      // ********************罗德里格斯公式实现 update************************//
67      Eigen :: Matrix3d update_matrix;
68      // Rodrigues Formula:
69      // R = cos(theta)I + (1 - cos(theta)aaT + sin(theta)a^)
70      // 罗德里格斯公式 part 1
71      Eigen :: Matrix3d cos_theta_I;
72      cos_theta_I << cos(theta), 0,                0,
73                                 0,           cos(theta),      0,
74                                 0,                    0,    cos(theta);
75      // 罗德里格斯公式 part 2
76      Eigen :: Matrix3d aaT;
77      aaT << a(0)*a(0),  0,                  0,
78                         0,           a(1)*a(1),       0,
79                         0,                  0,     a(2)*a(2);
80      Eigen :: Matrix3d second_part = (1-cos(theta)) * aaT;
81
82      // 罗德里格斯公式 part 3
83      Eigen :: Matrix3d a_hat;
84      a_hat << 0,    -a(2), a(1),
85               a(2),      0,    -a(0),
86               -a(1),  a(0),    0;
87
88      Eigen :: Matrix3d w_hat_exp = cos_theta_I + second_part + sin(theta) * a_hat;
89      std :: cout << " w_hat_exp is \n"<< w_hat_exp << endl;
90
91      // 用handmade李代数演示更新
92      R = R * w_hat_exp;
93      cout << "========\n" << "手写 R by Rodrigues' Formula = \n" << R << endl;
94
95      //**********************sophus库实现*****************************
96      // cout << "so3 hat = \n" << Sophus::SO3d::hat(so3) << endl;
```

```
    // cout << "w hat = \n" << Sophus::SO3d::hat(w) << endl;
    // 用李代数演示更新
    Sophus::SO3d updated_SO3_R = SO3_R * Sophus::SO3d::exp(w);
    cout << "============\n" << "updated_SO3_R = \n"
    << updated_SO3_R.matrix() << endl;

    //**********************四元数实现*****************************
    // 用四元数演示更新
     // 使用q方式储存
    Eigen::Quaterniond q_ = Eigen::Quaterniond (1, 0.5*w(0), 0.5*w(1), 0.5*w(2));
    // 更新q
    q_eigen = q_eigen * q_;
    q_eigen.normalize (); // 归一化
    cout << "============\n" << "updated_q = \n"
    << q_eigen.toRotationMatrix() << endl;

    // // 三种方法之间的差值
    // // 1. sophus 和 四元数
    // Eigen::Matrix3d diff = q_eigen.toRotationMatrix() - updated_SO3_R.matrix();
    // cout << "============\n" << " the difference between 2 methods is: \n"
    // << diff << endl;

    // 肉眼可以看出三者之间的差值较小,
    // 因此就不分别做差了, 否则还要两两比对, 输出三个矩阵
    return 0;
}
```

2. CMakeLists.txt

```
cmake_minimum_required(VERSION 3.1.2)
project (ch1)

set(CMAKE_CXX_ STANDARD 11)
```

```
5
6  find_package(Sophus REQUIRED)
7
8   include_directories ("/usr/include/eigen3")
9   include_directories (${Sophus_INCLUDE_DIRS})
10
11 add_executable( ch1_rotation_use_libs  ch1_rotation_use_libs .cpp)
12 add_executable(ch1_rotation_handmade ch1_rotation_handmade.cpp)
13
14   target_link_libraries  ( ch1_rotation_use_libs  Sophus::Sophus)
15   target_link_libraries  (ch1_rotation_handmade Sophus::Sophus)
```

程序运行结果如下

```
1  the  trace  of  R  is   2.41421
2  R  is  =
3   0.707107 -0.707107          0
4   0.707107   0.707107          0
5           0           0          1
6  q  transformed  by  Eigen  is  =          0          0 0.382683   0.92388
7  q  transformed  by  手写  is  =          0          0 0.382683   0.92388
8  R   from  SO3_R  is  =
9   0.707107 -0.707107          0
10  0.707107   0.707107          0
11          0           0          1
12 theta  is  :  0.0374166
13  w_hat_exp is
14      0.99935    -0.029993   0.0199953
15    0.029993        0.9995 -0.00999767
16  -0.0199953  0.00999767       0.99975
17 ============
18 R by 手写 Rodrigues' Formula =
19   0.685439 -0.727962 0.0212083
```

```
  0.727855  0.685545 0.00706942
-0.0199953 0.00999767 0.99975
============
updated_SO3_R =
  0.685368 -0.727891 0.0211022
  0.727926  0.685616 0.00738758
-0.0198454 0.0102976  0.99975
============
updated_q =
  0.685371 -0.727888 0.0210998
  0.727924  0.685618 0.00738668
-0.0198431 0.0102964  0.99975
```

可以看到三种更新方法之间的差值很小，验证了题目中所说的不同更新方法之间差距很小，因此实际中不区分具体采用哪种更新方式。

三. 其他导数

使用右乘 $\mathfrak{so}(3)$ 推导以下导数：

答：要用到的一些性质：

$\mathbf{R}^{-1} = \mathbf{R}^{\top}$

$a^{\wedge}b = -b^{\wedge}a$

$\mathbf{R}^{\top}exp(\varphi^{\wedge})\mathbf{R} = exp(\mathbf{R}\varphi)^{\wedge}$

$ln(\mathbf{R}exp(\varphi^{\wedge}) = ln(\mathbf{R}) + J_r^{-1}\varphi$

(a) $\frac{d(\mathbf{R}^{-1}\mathbf{p})}{d\mathbf{R}}$

$$
\begin{aligned}
&\frac{d(\mathbf{R}^{-1}\mathbf{p})}{d\mathbf{R}} \\
&= \lim_{\varphi \to 0} \frac{(\mathbf{R}exp(\varphi^{\wedge}))^{-1} - \mathbf{R}^{-1}p}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{(exp(\varphi^{\wedge})^{-1}\mathbf{R}^{-1}p - \mathbf{R}^{-1}p}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{(\mathbf{I} - \varphi^{\wedge})\mathbf{R}^{-1}p - \mathbf{R}^{-1}p}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{(-\varphi^{\wedge})\mathbf{R}^{-1}p}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{(\mathbf{R}^{-1}p)^{\varphi}}{\varphi} \\
&= (\mathbf{R}^{-1}p)^{\wedge}
\end{aligned}
\tag{2}
$$

(b) $\frac{dln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{d\mathbf{R}_2}$

$$
\begin{aligned}
&\frac{dln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{d\mathbf{R}_2} \\
&= \lim_{\varphi \to 0} \frac{ln(\mathbf{R}_1(\mathbf{R}_2exp(\varphi^{\wedge}))^{-1})^{\vee} - ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{ln(\mathbf{R}_1exp(\varphi^{\wedge})^{-1}\mathbf{R}_2^{-1})^{\vee} - ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{ln(\mathbf{R}_1\mathbf{R}_2^{-1}(\mathbf{R}_2exp(\varphi^{\wedge})^{-1}\mathbf{R}_2^{-1}))^{\vee} - ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{ln(\mathbf{R}_1\mathbf{R}_2^{-1}(\mathbf{R}_2exp(-\varphi^{\wedge})\mathbf{R}_2^{-1}))^{\vee} - ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{ln(\mathbf{R}_1\mathbf{R}_2^{-1}exp((\mathbf{R}_2(-\varphi))^{\wedge}))^{\vee} - ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{\varphi} \\
&= \lim_{\varphi \to 0} \frac{ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee} - J_r^{-1}\mathbf{R}_2\varphi - -ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee}}{\varphi} \\
&= -J_r^{-1}(ln(\mathbf{R}_1\mathbf{R}_2^{-1})^{\vee})\mathbf{R}_2
\end{aligned}
\tag{3}
$$

参考文献

[1] Mourikis, A. I., Roumeliotis, S. I. (2007, April). A multi-state constraint Kalman filter for vision-aided inertial navigation. In Proceedings 2007 IEEE International Conference on Robotics and Automation (pp. 3565-3572). IEEE.

[2] Bloesch, M., Burri, M., Omari, S., Hutter, M., Siegwart, R. (2017). Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. The International Journal of Robotics Research, 36(10), 1053-1072.

[3] Qin, T., Li, P., Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics, 34(4), 1004-1020.

[4] Mur-Artal, R., Tardós, J. D. (2017). Visual-inertial monocular SLAM with map reuse. IEEE Robotics and Automation Letters, 2(2), 796-803.

[5] Mur-Artal, R., Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE transactions on robotics, 33(5), 1255-1262.

[6] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 34(3), 314-334.

[7] Zuo, X., Geneva, P., Yang, Y., Ye, W., Liu, Y., Huang, G. (2019). Visual-inertial localization with prior LiDAR map constraints. IEEE Robotics and Automation Letters, 4(4), 3394-3401.

[8] Qin, T., Cao, S., Pan, J., Shen, S. (2019). A general optimization-based framework for global pose estimation with multiple sensors. arXiv preprint arXiv:1901.03642.

[9] Wu, K. J., Guo, C. X., Georgiou, G., Roumeliotis, S. I. (2017, May). Vins on wheels. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 5155-5162). IEEE.

[10] Costante, G., Mancini, M., Valigi, P., Ciarfuglia, T. A. (2015). Exploring representation learning with cnns for frame-to-frame ego-motion estimation. IEEE robotics and automation letters, 1(1), 18-25.

[11] Clark, R., Wang, S., Wen, H., Markham, A., Trigoni, N. (2017, February). Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).

[12] Wang, S., Clark, R., Wen, H., Trigoni, N. (2017, May). Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2043-2050). IEEE.

[13] Chen, C., Rosa, S., Miao, Y., Lu, C. X., Wu, W., Markham, A., Trigoni, N. (2019). Selective sensor fusion for neural visual-inertial odometry. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10542-10551).

[14] Rosinol, A., Abate, M., Chang, Y., Carlone, L. (2020, May). Kimera: an open-source library for real-time metric-semantic localization and mapping. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1689-1696). IEEE.