

一步步深入了解S-MSCKF

作者:钟心亮

联系方式:xinliangzhong@foxmail.com

一步步深入了解S-MSCKF

0.前言

0.1 我为什么写这个?

0.2 可爱的您能得到什么?

0.3 我希望可爱的您反馈什么?

1.简介

符号说明

2.前端

2.1 基本数据结构说明

A.特征点检测和跟踪的参数

B. 特征点数据

2.2 跟踪流程

2.2.1 Initialization

2.2.2 trackFeatures

2.2.3 addNewFeatures & pruneGridFeatures

2.2.4 publish

2.3 显示

3.基于四元数的 error-state Kalman Filter

4.S-MSCKF

4.1 基本原理

A.运动模型

B. 增广

C.观测模型

4.2 三角化

4.3 能观测性分析

4.4 滤波器更新机制

5.S-MSCKF代码流程

6.总结

附录

A. F矩阵和G矩阵的推导

B. J_I 的计算

C. H矩阵

D. 三种常用数值积分方式:欧拉,中值,RK4积分

7.参考文献

0.前言

0.1 我为什么写这个?

一方面是对自己的东西做一个简单的总结,之前也看过一些其他的,但都没有系统的成文,因为时间比较紧张,科研压力比较大.另外一方面是向各位大佬们学习,希望能得到大佬们的指导,同时希望对刚入门SLAM和已经入门一段时间的SLAM小伙伴一点干货.结交一些志同道合的小伙伴!

0.2 可爱的您能得到什么?

如果您正在研究MSCKF或者对VIO很感兴趣,那么您将得到以下几件东西:

(1) [一份看了不会亏的MSCKF的入门指南](#)

(2) 基本我自己推导过的一本书(或者叫比较长的论文) [Quaternion kinematics for the error-state Kalman filter](#)

(3) [一份注释过的开源版本的代码](#)

如果您是已经最MSCKF及其熟悉的大佬,那么您将得到:

其实我也还没想好您能得到什么,我倒是很想从您那得到一些东西...

0.3 我希望可爱的您反馈什么?

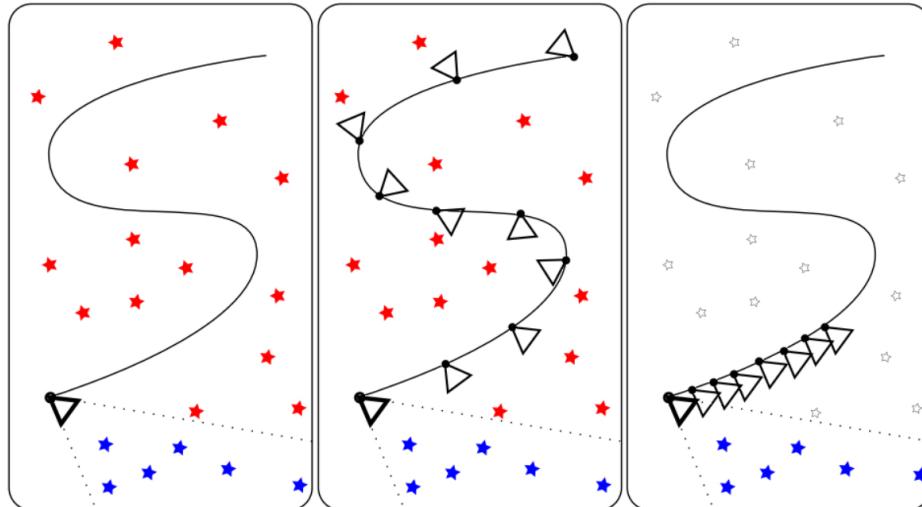
(1) 全文一些描述有误的地方,因为自己写东西比较随意,难免有大家理解起来有问题或者本身就描述不对的地方.

(2) 反馈网址戳 [这里](#), 可爱的您一定会随手在github留下小星星对不对? 不方便公共留言的话可以邮箱私聊我.

(3) 交个朋友? 江湖有缘相见. 祝可爱的您科研顺利,一夜暴富!!!

1.简介

MSCKF (Multi-State Constraint Kalman Filter),从2007年提出至今,一直是filter-based SLAM比较经典的实现.据说这也是谷歌tango里面的算法,这要感谢Mingyang Li博士在MSCKF的不懈工作.在传统的EKF-SLAM框架中,特征点的信息会加入到特征向量和协方差矩阵里,这种方法的缺点是特征点的信息会给一个初始深度和初始协方差,如果不正确的话,很容易导致后面不收敛,出现inconsistent的情况. MSCKF维护一个pose的FIFO,按照时间顺序排列,可以称为滑动窗口,一个特征点在滑动窗口的几个位姿都被观察到的话,就会在这几个位姿间建立约束,从而进行KF的更新.如下图所示,左边代表的是传统EKF SLAM,红色五角星是old feature,这个也是保存在状态向量中的,另外状态向量中只保存最新的相机姿态;中间这张可以表示的是keyframe-based SLAM,它会保存稀疏的关键帧和它们之间关联的地图点;最右边这张则可以代表MSCKF的一个基本结构,MSCKF中老的地图点和滑窗之外的相机姿态是被丢弃的,它只存了滑窗内部的相机姿态和它们共享的地图点.



符号说明

$\dot{\mathbf{x}}$ 表示微分

$\tilde{\mathbf{x}}$ 表示误差

$\hat{\mathbf{x}}$ 表示估计值

\mathbf{x}_m 表示测量值

G 惯性系

C 相机坐标系

I IMU坐标系

${}^G \mathbf{p}$ G 系下的一个点

${}^A_B \mathbf{R}$ 从 A 系到 B 系的旋转矩阵

2. 前端

本文主要针对2017年Kumar实验室开源的S-MSCKF进行详细分析,其实这篇文章整体思路与07年提出的基本上是一脉相承的.作为一个VIO的前端,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,另外这是MSCKF双目的一个实现,双目之间的特征点匹配采用的也是光流,这与传统的基于descriptor匹配的方法不同.前端部分其实相对简单,整个前端部分基本在 [image_processor.cpp](#) 中实现.

2.1 基本数据结构说明

A. 特征点检测和跟踪的参数

```
struct ProcessorConfig {
    int grid_row;           //划分图像网格的行数
    int grid_col;           //划分图像网格的列数
    int grid_min_feature_num; //每个网格特征点的最小个数
    int grid_max_feature_num; //每个网格特征点的最大个数

    int pyramid_levels;      //金字塔层数
    int patch_size;
    int fast_threshold;
    int max_iteration;
    double track_precision;
    double ransac_threshold;
    double stereo_threshold;
```

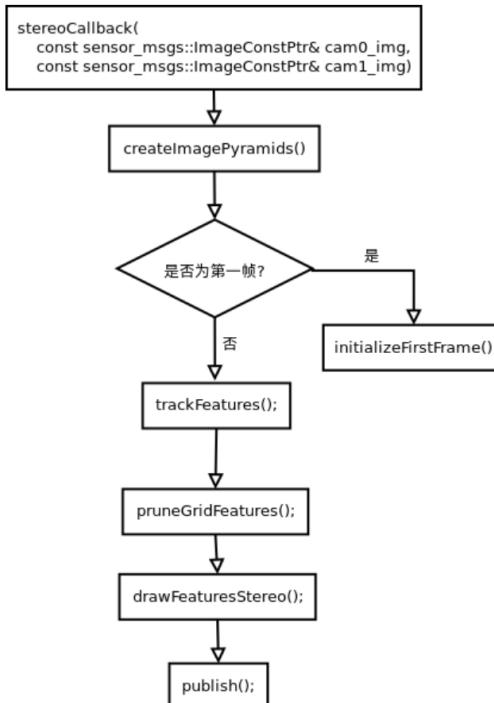
```
};
```

B. 特征点数据

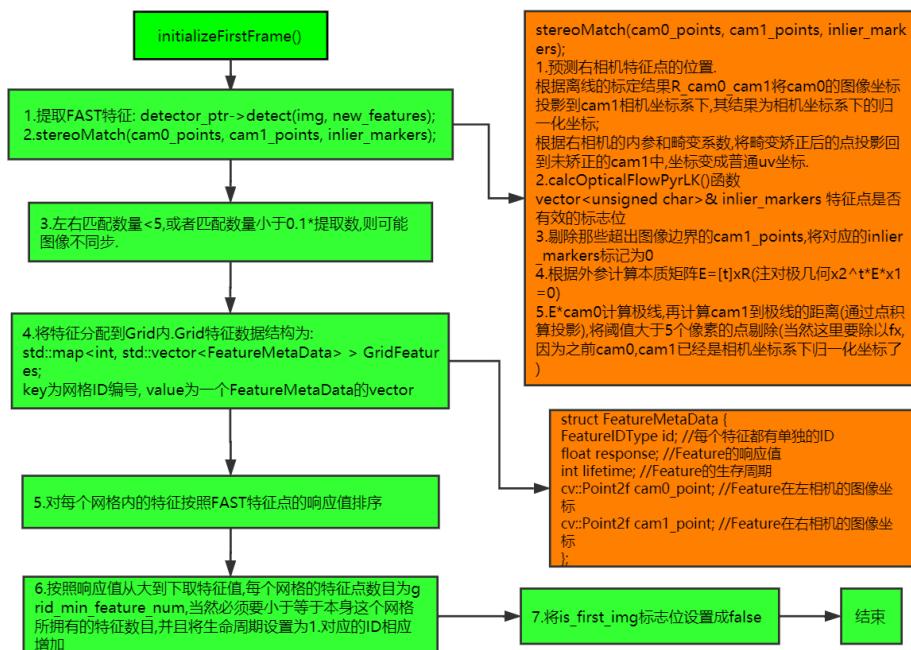
```
struct FeatureMetaData {  
    FeatureIDType id; //unsigned long long int 每个特征都有单独的ID  
    float response; //Feature的响应值  
    int lifetime; //Feature的生存周期  
    cv::Point2f cam0_point; //Feature在左相机的图像坐标  
    cv::Point2f cam1_point; //Feature在右相机的图像坐标  
};
```

2.2 跟踪流程

整体框架如下面的流程图所示：

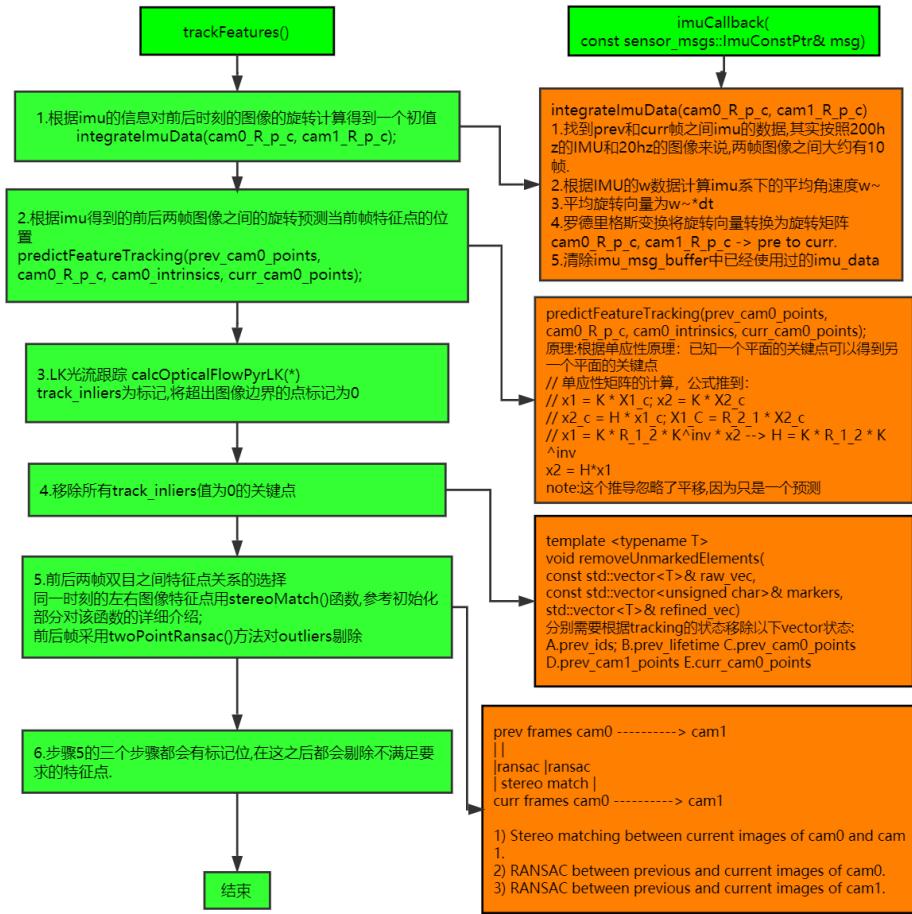


2.2.1 Initialization



2.2.2 trackFeatures

当第一帧初始化完成之后, 后面帧则只需要进行跟踪第一帧的特征点, 并且提取新的特征点, 整个流程如下:



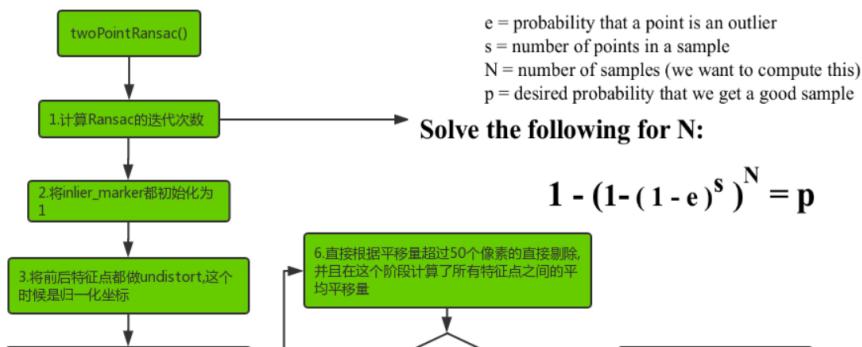
整个流程还算比较清晰,但是有一个步骤需要单独说明一下,也就是作者在论文中提到的twoPointRansac的方法.我们先来看一下函数原型:

```

/**
 * @brief 计算原图像帧关键点对应的矫正位置
 * @param pts1: 上一时刻的关键点位置
 * @param pts2: 当前时刻跟踪匹配到的关键点位置
 * @param R_p_c: 根据IMU信息计算得到的两个时刻相机的相对旋转信息
 * @param distortion_model,intrinsics: 相机内参和畸变模型
 * @param inlier_error: 内点可接受的阈值 (关键点距离差)
 * @param success_probability: 成功的概率
 * @return inlier_markers: 内点标志位
 */
void ImageProcessor::twoPointRansac(
    const vector<Point2f>& pts1, const vector<Point2f>& pts2,
    const cv::Matx33f& R_p_c, const cv::Vec4d& intrinsics,
    const std::string& distortion_model,
    const cv::Vec4d& distortion_coeffs,
    const double& inlier_error,
    const double& success_probability,
    vector<int>& inlier_markers)

```

整个函数的基本流程如下:





下面我们来详细讲解一下RANSAC模型及原理依据: 我们由对极几何可以知道有以下约束:

$$p_2^T \cdot [t]_x \cdot R \cdot p_1 = 0 \quad (2.1)$$

我们假设前后帧对应点归一化坐标分别为,

$$R \cdot p_1 = [x_1 \ y_1 \ 1]^T, p_2 = [x_2 \ y_2 \ 1]^T \quad (2.2)$$

其中R为根据IMU的平均角速度得到的,此时坐标系都统一到一个坐标系下.

$$[x_2 \ y_2 \ 1] \cdot \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = 0 \quad (2.3)$$

将式子(2.3)展开之后我们可以得到:

$$\begin{bmatrix} y_1 - y_2 & -(x_1 - x_2) & x_1 y_2 - x_2 y_1 \\ y_3 - y_4 & -(x_3 - x_4) & x_3 y_4 - x_4 y_3 \end{bmatrix} \cdot \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = 0 \quad (2.4)$$

其中绿色部分在代码中对应这一块:

```

vector<Point2d> pts_diff(pts1_undistorted.size());
for (int i = 0; i < pts1_undistorted.size(); ++i)
    pts_diff[i] = pts1_undistorted[i] - pts2_undistorted[i];
...
...
MatrixXd coeff_t(pts_diff.size(), 3);
for (int i = 0; i < pts_diff.size(); ++i) {
    coeff_t(i, 0) = pts_diff[i].y;
    coeff_t(i, 1) = -pts_diff[i].x;
    coeff_t(i, 2) = pts1_undistorted[i].x * pts2_undistorted[i].y -
        pts1_undistorted[i].y * pts2_undistorted[i].x;
}

```

至于这个模型是怎么选出来的呢? 假设一共有N个inliers点对,那么根据式(2.4)势必会得到一个N*3 * 3*1 = N(0)的等式.但事实上由于误差和outliers的存在,最终结果不可能为零,我们取两个点将式子分块,并且只考虑两个点的情况,那么将会有:

$$\begin{bmatrix} y_1 - y_2 & -(x_1 - x_2) & x_1 y_2 - x_2 y_1 \\ y_3 - y_4 & -(x_3 - x_4) & x_3 y_4 - x_4 y_3 \end{bmatrix} \cdot \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}^T \cdot \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.5)$$

那我们可以分别得到以下三个式子:

$$\begin{aligned} \begin{bmatrix} A_x \\ A_y \end{bmatrix}^T \cdot \begin{bmatrix} t_x \\ t_y \end{bmatrix} &\approx A_z \cdot t_z \\ \begin{bmatrix} A_x \\ A_z \end{bmatrix}^T \cdot \begin{bmatrix} t_x \\ t_z \end{bmatrix} &\approx A_y \cdot t_y \\ \begin{bmatrix} A_y \\ A_z \end{bmatrix}^T \cdot \begin{bmatrix} t_y \\ t_z \end{bmatrix} &\approx A_x \cdot t_x \end{aligned} \quad (2.6)$$

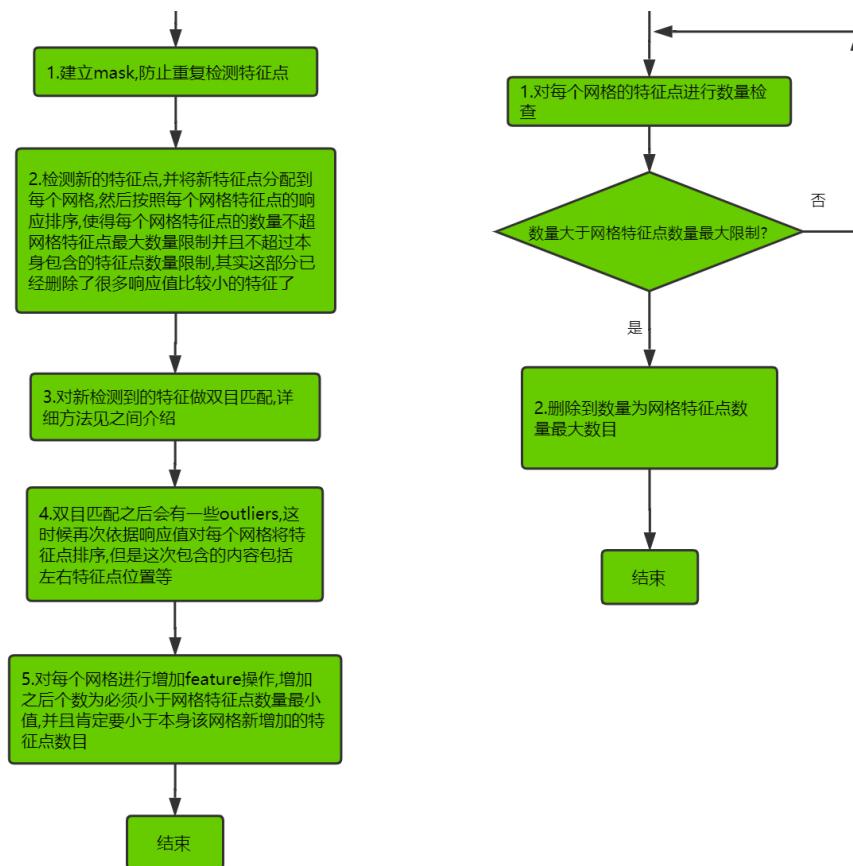
我们的目标当然是使得误差最小,所以作者的做法是比较式子(2.6)绿色部分的大小,取最小的并令模型的平移为1,进而直接求逆然后得到最初的模型假设,之后要做的步骤跟常规RANSAC就十分接近了,找出适应当前模型的所有inliers,然后计算误差并不断迭代找到最好的模型. 至此我们已经完成了整个feature的tracking过程!

2.2.3 addNewFeatures & pruneGridFeatures

如果一直tracking的话那么随着时间流逝,有些特征会消失,有些可能也会有累计误差,所以我们势必要添加一些新的特征,这个步骤正是在跟踪上一帧特征之后要做的,因为stereoMatching 和twoPointRansac都会剔除一些outliers,那我们需要提取新的特征保证能一直运行下去.

addNewFeatures()

pruneGridFeature
s()



2.2.4 publish

主要发布了当前帧特征点的数据, 每个特征的数据结构为FeatureMeasurement如下:

```

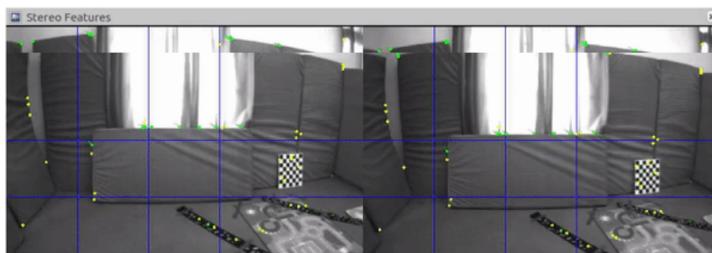
uint64 id
# Normalized feature coordinates (with identity intrinsic matrix)
float64 u0 # horizontal coordinate in cam0
float64 v0 # vertical coordinate in cam0
float64 u1 # horizontal coordinate in cam0
float64 v1 # vertical coordinate in cam0

```

发布的其实是CameraMeasurement, 那其实就是一个包含上面特征数据FeatureMeasurement的一个数组.

2.3 显示

其实前端基本上可以说是非常简单了, 也没有太多的trick, 最后我们来看一下前端的跟踪效果的动图:



3. 基于四元数的 error-state Kalman Filter

其实原理部分相当重要, 包括你对error-state Kalman Filter的理解, 但是如果要从头讲起的话可以说篇幅太长, 考虑到做SLAM的同学们基本上都应该知道这本书 [Quaternion kinematics for the error-state Kalman filter](#), 这本书会让你在四元数, SO3, IMU的模型以及基于IMU的ESKF(error-state Kalman Filter)都会有比较深刻的理解, 对应这本书我也做了很多注释, 关于基于四元数原理部分由于篇幅限制, 我这里不想做太多的说明, 但是接下来在讲解S-MSCKF原理部分我们会将参考公式附上. 下面是一些我在这本参考资料中的注释图.



别代表IMU的bias,而 $\hat{\mathbf{q}}$ 和 $\hat{\mathbf{p}}_C$ 分别代表从相机坐标系到IMU坐标系的旋转与平移,其中以左相机为准.但是我们注意到旋转实际上只有三个自由度,而且四元数必须是单位四元数,那这样额外的约束会使得协方差矩阵奇异,所以我们定义error IMU状态如下:

$$\tilde{\mathbf{x}}_I = \begin{bmatrix} {}^I_G\tilde{\theta}^T & \tilde{\mathbf{b}}_g^T & {}^G\tilde{\mathbf{v}}_I^T & \tilde{\mathbf{b}}_a^T & {}^G\tilde{\mathbf{p}}_I^T & {}^I_C\tilde{\mathbf{q}}^T & {}^I\tilde{\mathbf{p}}_C^T \end{bmatrix}^T \quad (4.2)$$

所以IMU的状态一共是 $3 + 7 = 21$ 维向量,后6维度其实是相机与IMU的外参,这6个维度在MSCKF1.0版本是不在状态向量里边的. MSCKF的状态向量还包含另外一个组成部分,那就是N个相机的姿态,那么每个相机的姿态误差向量定义为 $\tilde{\mathbf{x}}_{C_i} = \begin{pmatrix} {}^C_G\tilde{\theta}^T & {}^G\tilde{\mathbf{p}}_{C_i}^T \end{pmatrix}^T$,所以当滑窗里边有N个相机姿态的时候,整个误差状态向量为:

$$\tilde{\mathbf{x}} = (\tilde{\mathbf{x}}_I^T \quad \tilde{\mathbf{x}}_{C_1}^T \quad \dots \quad \tilde{\mathbf{x}}_{C_N}^T)^T \quad (4.3)$$

我们以标准error-state KF为准,其过程包含运动模型和观测模型.

A.运动模型

我们知道对于IMU的状态连续时间的运动学有:

$$\begin{aligned} {}^I_G\dot{\mathbf{q}} &= \frac{1}{2} \cdot {}^I_G\dot{\mathbf{q}} \otimes \hat{\mathbf{w}} = \frac{1}{2} \Omega(\hat{\mathbf{w}}) {}^I_G\dot{\mathbf{q}}, \\ \dot{\mathbf{b}}_g &= \mathbf{0}_{3 \times 1}, \\ {}^G\dot{\mathbf{v}} &= C({}^I_G\dot{\mathbf{q}})^T \hat{\mathbf{a}} + {}^G\mathbf{g}, \\ \dot{\mathbf{b}}_a &= \mathbf{0}_{3 \times 1}, \\ {}^G\dot{\mathbf{p}}_I &= {}^G\dot{\mathbf{v}}, \\ {}^I_C\dot{\mathbf{q}} &= \mathbf{0}_{3 \times 1}, \\ {}^I\dot{\mathbf{p}}_C &= \mathbf{0}_{3 \times 1} \end{aligned} \quad (4.4)$$

其中 $\hat{\mathbf{w}}$ 和 $\hat{\mathbf{a}}$ 分别为角速度和加速度的估计值(测量值减去bias),即有:

$$\hat{\mathbf{w}} = w_m - \hat{\mathbf{b}}_g, \hat{\mathbf{a}} = a_m - \hat{\mathbf{b}}_a \quad (4.5)$$

其中有几点要说明,其中,

$$\Omega(\hat{\mathbf{w}}) = \Omega(\hat{\mathbf{w}}) = \begin{pmatrix} -[\hat{\mathbf{w}}_\times] & \mathbf{w} \\ -\mathbf{w}^T & 0 \end{pmatrix} \quad (4.6)$$

这个直接参考四元数乘法就可以了,然后 $[\hat{\mathbf{w}}_\times]$ 是 $\hat{\mathbf{w}}$ 的反对称矩阵; $C(\cdot)$ 表示四元数到旋转矩阵的转换,这个可以参照[1]和[2]. 那按照S-MSCKF的论文所述,我们可以得到以下式子,

$$\dot{\tilde{\mathbf{x}}}_I = \mathbf{F}\tilde{\mathbf{x}}_I + \mathbf{G}\mathbf{n}_I \quad (4.7)$$

其中

$$\mathbf{n}_I^T = (\mathbf{n}_g^T \quad \mathbf{n}_{wg}^T \quad \mathbf{n}_a^T \quad \mathbf{n}_{wa}^T)^T \quad (4.8)$$

\mathbf{n}_g 和 \mathbf{n}_a 分别代表角速度和加速度的测量噪声,服从高斯分布; \mathbf{n}_{wg} 和 \mathbf{n}_{wa} 分别代表角速度和加速度的bias的随机游走噪声. \mathbf{F} 是 21×21 大小矩阵, \mathbf{G} 是 21×12 大小的矩阵,其详细推到见附录A.

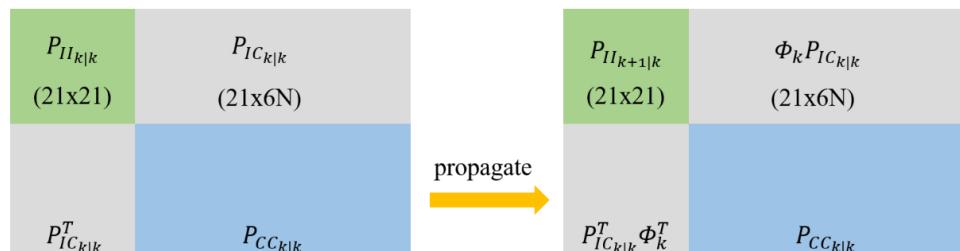
对于IMU的状态来说,我们可以采用RK4的积分方法根据(4.4)求得IMU的状态. 那么对于IMU的协方差矩阵呢,我们需要事先求取状态转移矩阵和离散的运动噪声协方差矩阵,如下:

$$\begin{aligned} \Phi_k &= \Phi(t_{k+1}, t_k) = \exp\left(\int_{t_k}^{t_{k+1}} \mathbf{F}(\tau) d\tau\right) \\ \mathbf{Q}_k &= \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) \mathbf{G} \mathbf{Q} \mathbf{G} \Phi(t_{k+1}, \tau)^T d\tau \end{aligned} \quad (4.9)$$

关于这个状态转移矩阵 Φ_k 的求法,其实式子4.9是一个指数,指数项是一个积分项,当 t_{k+1} 与 t_k 间 Δt 较小时,可以得到这样的式子:

$$\Phi_k = \Phi(t_{k+1}, t_k) = \exp\left(\int_{t_k}^{t_{k+1}} \mathbf{F}(\tau) d\tau\right) = \exp(\mathbf{F}\Delta t) = I + \mathbf{F}\Delta t + \frac{1}{2!}(\mathbf{F}\Delta t)^2 + \frac{1}{3!}(\mathbf{F}\Delta t)^3 + \dots \quad (4.10)$$

整个状态(IMU+Camera)的covariance传播过程如图所示:





那么对于左上角IMU的covariance的传播有:

$$\mathbf{P}_{II_{k+1|k}} = \Phi_k \mathbf{P}_{II_{k|k}} \Phi_k^T + \mathbf{Q}_k \quad (4.11)$$

其中Camera的covariance暂时还没有变化是因为这个时间段图像还没有到来,只有IMU的影响,但是会影响到IMU与Camera协方差,即上图灰色矩形块.

B. 增广

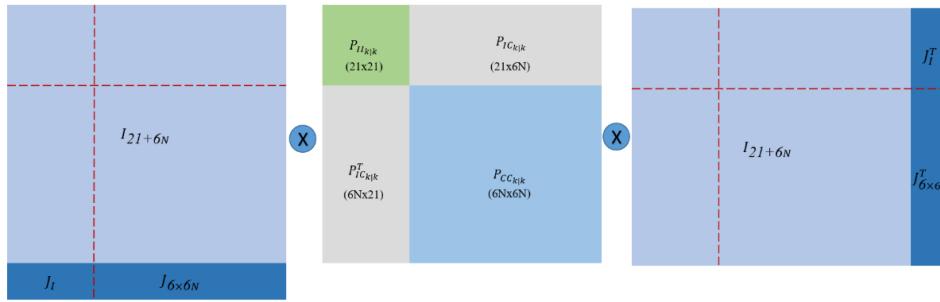
当图像到来时,需要对当前相机姿态做增广,这个时刻的相机姿态是由上一时刻的IMU propagate的结果结合外参得到的:

$$\begin{aligned} {}_G^C \hat{\mathbf{q}} &= {}_I^C \hat{\mathbf{q}} \otimes {}_G^I \hat{\mathbf{q}} \\ {}^G \hat{\mathbf{p}}_C &= {}^G \hat{\mathbf{p}}_I + C({}_G^I \hat{\mathbf{q}})^T \cdot {}^I \hat{\mathbf{p}}_C \end{aligned} \quad (4.12)$$

假设上一时刻共有N个相机姿态在状态向量中,那么当新一帧图像到来时,这个时候整个滤波器的状态变成了 $21 + 6(N + 1)$ 的向量,那么它对应的covariance维度变为 $(21 + 6(N + 1)) \times (21 + 6(N + 1))$.其数学表达式为:

$$\mathbf{P}_{k|k} = \begin{pmatrix} \mathbf{I}_{21+6N} \\ \mathbf{J} \end{pmatrix} \mathbf{P}_{k|k} \begin{pmatrix} \mathbf{I}_{21+6N} \\ \mathbf{J} \end{pmatrix}^T \quad (4.13)$$

这个过程对应如下图过程:



其中 \mathbf{J} 的详细推导过程见附录B.

C. 观测模型

MSCKF的观测模型是以特征点为分组的,我们可以知道一个特征(之前一直处于跟踪成功状态)会拥有多个 Camera State.所有这些对于同一个3D点的Camera State都会去约束观测模型. 那这样其实隐式的将特征点位置从状态向量中移除,取而代之的是Camera State. 我们考虑单个feature f_j , 假设它所对应到 M_j 个相机姿态 $({}_G^C \mathbf{q}, {}^G \mathbf{p}_{C_i})$, $i \in j$.当然双目版本的包含左目和右目两个相机姿态, $({}_G^{C_{i,1}} \mathbf{q}, {}^G \mathbf{p}_{C_{i,1}})$ 和 $({}_G^{C_{i,2}} \mathbf{q}, {}^G \mathbf{p}_{C_{i,2}})$ 右相机很容易能通过外参得到. 其中双目的观测值可以表示如下:

$$\mathbf{z}_i^j = \begin{pmatrix} u_{i,1}^j \\ v_{i,1}^j \\ u_{i,2}^j \\ v_{i,2}^j \end{pmatrix} = \begin{pmatrix} \frac{1}{c_{i,1} X_j} & \mathbf{0}_{2 \times 2} \\ \frac{1}{c_{i,1} Z_j} & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & \frac{1}{c_{i,2} Z_j} \\ \mathbf{0}_{2 \times 2} & \frac{1}{c_{i,2} X_j} \end{pmatrix} \begin{pmatrix} \frac{1}{c_{i,1} X_j} \\ \frac{1}{c_{i,1} Y_j} \\ \frac{1}{c_{i,2} X_j} \\ \frac{1}{c_{i,2} Y_j} \end{pmatrix} \quad (4.14)$$

而特征点在两个相机坐标系下可以分别表示为:

$$\begin{aligned} {}_{C_{i,1}} \mathbf{p}_j &= \begin{pmatrix} c_{i,1} X_j \\ c_{i,1} Y_j \\ c_{i,1} Z_j \end{pmatrix} = C({}_G^{C_{i,1}} \mathbf{q})({}^G \mathbf{p}_j - {}^G \mathbf{p}_{C_{i,1}}) \\ {}_{C_{i,2}} \mathbf{p}_j &= \begin{pmatrix} c_{i,2} X_j \\ c_{i,2} Y_j \\ c_{i,2} Z_j \end{pmatrix} = C({}_G^{C_{i,2}} \mathbf{q})({}^G \mathbf{p}_j - {}^G \mathbf{p}_{C_{i,2}}) \\ &= C({}_G^{C_{i,2}} \mathbf{q})({}_{C_{i,1}} \mathbf{p}_j - {}_{C_{i,1}} \mathbf{p}_{C_{i,2}}) \end{aligned} \quad (4.15)$$

其中 ${}^G \mathbf{p}_j$ 是特征点在惯性系下的坐标,这个是通过这个特征点的对应的所有camera state三角化得到的结果. 将观测模型在当前状态线性化可以得到如下式子:

$$\mathbf{r}_i^j = \mathbf{z}_i^j - \hat{\mathbf{z}}_i^j = \mathbf{H}_{C_i}^j \tilde{\mathbf{x}}_{C_i} + \mathbf{H}_{f_i}^j \tilde{\mathbf{p}}_j + \mathbf{n}_i^j \quad (4.16)$$

其中 \mathbf{n}_i^j 是观测噪声, $\mathbf{H}_{C_i}^j$ 和 $\mathbf{H}_{f_i}^j$ 是对应的雅克比矩阵. 其详细推导和解析见附录C. 式子(4.16)对应到的是单个特征点对应的其中某一个相机姿态, 但是这个特征点会对应到很多相机姿态, 我们直接将它贴在后边可以得到一个特征点的残差模型为:

$$\mathbf{r}^j = \mathbf{H}_{\mathbf{x}}^j \tilde{\mathbf{x}} + \mathbf{H}_{f_i}^j \tilde{\mathbf{p}}_j + \mathbf{n}^j \quad (4.17)$$

但是这个其实并不是一个标准的EKF观测模型, 因为我们知道 $\tilde{\mathbf{p}}_j$ 并不在我们的状态向量里边, 所以做法是将式子(4.17)中红色部分投影到零空间, 假设 \mathbf{H}_f^j 的 left null space 为 \mathbf{V}^T , 即有 $\mathbf{V}^T \mathbf{H}_f^j = \mathbf{0}$, 所以式(4.17)可有写成:

$$\begin{aligned} \mathbf{r}_o^j &= \mathbf{V}^T \mathbf{H}_{\mathbf{x}}^j \tilde{\mathbf{x}} + \mathbf{V}^T \mathbf{n}^j = \\ &\mathbf{H}_{\mathbf{x}, \mathbf{o}}^j \tilde{\mathbf{x}} + \mathbf{n}_o^j \end{aligned} \quad (4.18)$$

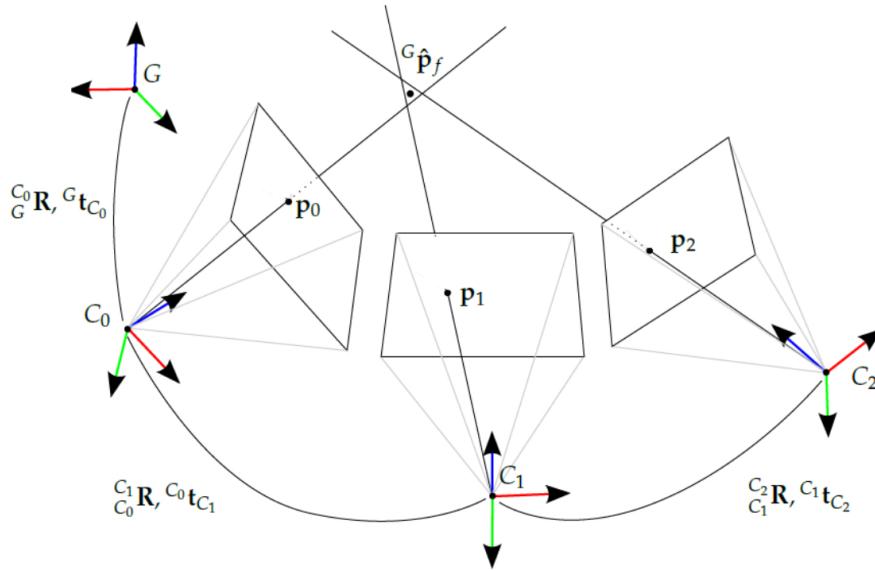
式(4.18)则是一个标准的EKF观测模型了, 下面简单分析一下维度. 分析时针对单个特征点, 我们知道 \mathbf{H}_f^j 的维度是 $4M_j \times 3$, 那么它的left null space的维度即 \mathbf{V}^T 的维度为 $(4M_j - 3) \times 4M_j$, 则最终 $\mathbf{H}_{\mathbf{x}, \mathbf{o}}^j \tilde{\mathbf{x}}$ 的维度变为 $(4M_j - 3) \times 6$, 残差的维度变为 $(4M_j - 3) \times 1$, 假设一共有L个特征的话, 那最终残差的维度会是 $L(4M_j - 3) \times 1$. 更多详细的代码细节见给到的注释版代码, 然后H矩阵的详细推导见附录C.

4.2 三角化

三角化是通过多帧相机对同一个点的观测计算出特征点在世界坐标系下的绝对3D坐标, 或者你可以认为是恢复出一个比较可靠的3D点. 在讲这个之前, 我们先来简单过一下相机的投影模型, 假设相机图像已经去畸变了, 那么我们很容易得到这样一个模型:

$$\begin{bmatrix} u \\ v \end{bmatrix} = h \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \quad (4.19)$$

其中(X,Y,Z)为相机坐标系下的一个点. 我们再来看下图:



其中在惯性系下 ${}^G \hat{\mathbf{p}}_f$ 被多帧相机观测到, 其中在每个相机下的坐标表示为 ${}^{C_i} \mathbf{p}_f = ({}^{C_i} X, {}^{C_i} Y, {}^{C_i} Z)^T$, 假设该特征第一个观测为 ${}^{C_0} \mathbf{p}_f$, 余数在第 i 个相机帧中可以表示为:

$${}^{C_i} \mathbf{p}_f = {}^{C_0} \mathbf{R} {}^{C_0} \mathbf{p}_f + {}^{C_i} \mathbf{p}_{C_0} \quad (4.20)$$

我们将这个转换为逆深度的表达形式, 可以得到下面一系列式子:

$$\begin{aligned} {}^{C_i} \mathbf{p}_f &= {}^{C_0} \mathbf{R} \begin{bmatrix} {}^{C_0} X \\ {}^{C_0} Y \\ {}^{C_0} Z \end{bmatrix} + {}^{C_i} \mathbf{p}_{C_0} \\ &= {}^{C_0} Z \begin{pmatrix} {}^{C_0} X / {}^{C_0} Z \\ {}^{C_0} Y / {}^{C_0} Z \\ 1 \end{pmatrix} + \frac{1}{{}^{C_0} Z} \cdot {}^{C_i} \mathbf{p}_{C_0} \\ &= \frac{1}{\rho} \mathbf{g}_i \begin{pmatrix} \alpha \\ \beta \\ 1 \end{pmatrix} + \rho {}^{C_0} \mathbf{p}_{C_0} \\ &= \frac{1}{\rho} \mathbf{g}_i \begin{pmatrix} \alpha \\ \beta \\ \rho \end{pmatrix} = \frac{1}{\rho} \mathbf{g}_i(\theta) \end{aligned} \quad (4.21)$$

共十项参数, $\alpha = \dots$, $\beta = \dots$, $\rho = \dots$, $\theta = \dots$.

并且这个假设我们的 $C_i \mathbf{p}_f$ 为 $(C_i X / C_i Z, C_i Y / C_i Z)^T$, 那么就是说 $g_i(\theta)$ 是一个3维输入, 二维输出的函数. 所以误差函数可以写成:

$$f_i(\theta) = \mathbf{z}_i - h(g_i(\theta)) \quad (4.22)$$

假设一共有 N 个相机观测, 那么我们可以构建一个最小二乘问题, 形如下式:

$$\arg \min \sum_{i=1}^n \|f_i(\theta)\|_2 \quad (4.23)$$

其中对应于单个特征点的 *Jacobian* 形式如下:

$$J_f = \frac{\partial f}{\partial \theta} = \frac{\partial h}{\partial g} \frac{\partial g}{\partial \theta} \quad (4.24)$$

其中第一项 $\frac{\partial f}{\partial g}$ 非常简单, 就是参考式(4.19), 得到的结果第比较简单, 第二部分 $\frac{\partial g}{\partial \theta}$ 根据式子(4.21)可以很容易得到

$$\begin{aligned} \frac{\partial g}{\partial \theta} &= \left[\frac{\partial g_i}{\partial \alpha} \quad \frac{\partial g_i}{\partial \beta} \quad \frac{\partial g_i}{\partial \rho} \right] \\ &= \left[\begin{array}{c} 1 \\ 0 \\ 0 \end{array} \right], \left[\begin{array}{c} 0 \\ 1 \\ 0 \end{array} \right], \left[\begin{array}{c} 0 \\ 0 \\ 1 \end{array} \right] \end{aligned} \quad (4.25)$$

然后用高斯-牛顿法就可以很容易解决这个最小二乘问题. 最后能得到 $\hat{\theta} = [\hat{\alpha}, \hat{\beta}, \hat{\rho}]$, 那么其实也就是特征点在首个观测到它的相机帧下的坐标, 根据下面的式子则很容易恢复出惯性系下的特征点的位置:

$${}^G \hat{\mathbf{p}}_f = \frac{1}{\hat{\rho}} {}^G {}_{C_0} \mathbf{R} \begin{bmatrix} \hat{\alpha} \\ \hat{\beta} \\ 1 \end{bmatrix} + {}^G \mathbf{p}_{C_0} \quad (4.26)$$

注意, 代码中的实现和现在这个推到稍微有点出入, 它的实现主要参考的是文献7, 不过基本大同小异, 大家阅读起来应该也不会有太大的阻碍.

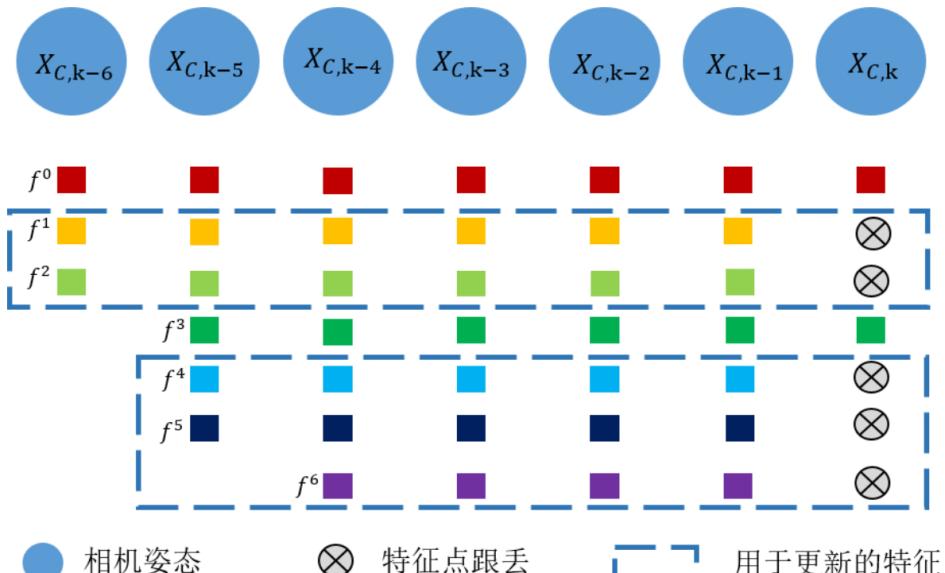
4.3 能观测性分析

关于能观测性分析, 我个人感觉公式太多了, 并且没有想到一个很好的描述方式, 理解的也不算太透彻, 所以这里还希望有大佬能把这部分写一下, 或者单独写一个专题, 我觉得那是极好的.

另外开源版本的 S-MSCKF 用的是 OC-EKF, 这个主要参考了这两篇论文 [Consistency Analysis and Improvement of Vision-aided Inertial Navigation](#) 和 [On the consistency of Vision-aided Inertial Navigation](#). 代码中的实现主要参考了第二篇给出的公式, 我在注释里应该都有注明.

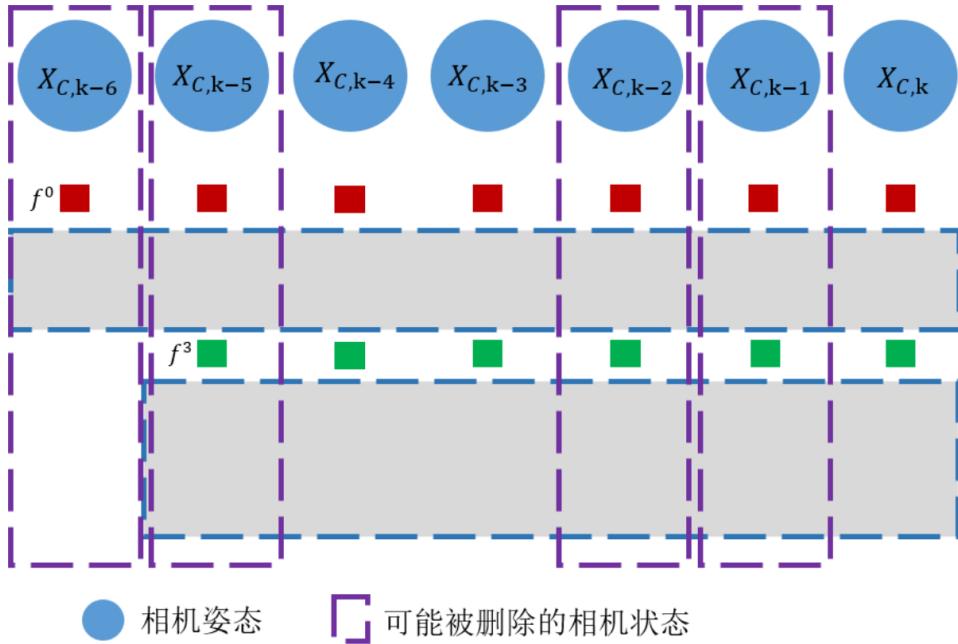
4.4 滤波器更新机制

大致有两种更新策略, 假设新进来一帧图像, 这个时候会丢失一些特征点, 这个时候丢失的特征点(且三角化成功)用于滤波器更新, 如下图所示:



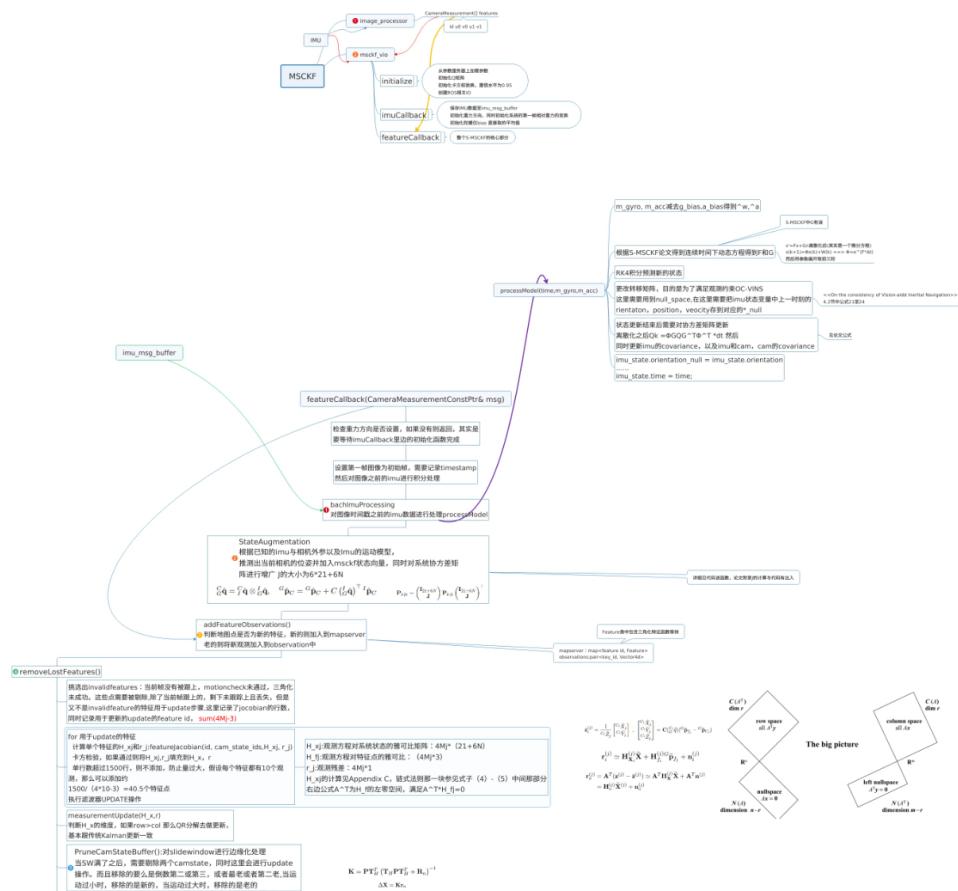
那当然随着时间的推移, 相机状态会越来越多, 这个时候呢, 相机状态会有一个阈值, 也就是滑窗的上限, S-MSCKF 与 msckf1.0 有稍微不同, 它是当满了之后每次迭代的清除两个, 最新的这个状态肯定保持, 清除依据就是帧间的旋转

跟位移大小,如下图所示,假设Slide Window的大小正好为7,且已经经过了上面的update过程,那么这个时候还会再update一次,这个时候它的所有特征都会用于更新.因为要移除两个camera state,所以对应的状态和covariance也需要剔除掉,所以删掉的两个状态其实肯定处于紫色框其中的两个.



5.S-MSCKF代码流程

这里放一张之前做的图,清晰图片从[这里下载](#),另外中文注释版本的代码在[这里](#)



6. 总结

全文以S-MSCKF为依托, 主要对MSCKF的理论基础和实现原理及细节进行了讲述, 并且对论文公式进行了详细的推导(很多手写的地方实在是不想敲了, 太费时间了), 然后还对Quaternion kinematics for the error-state Kalman filter这本书进行了详细的注释, 同时对开源版本的S-MSCKF的代码进行了注释. 由于笔者水平有限, 有些地方理解难免不到位, 其中就包括能观性分析这部分还没有做比较到位的解释, 最后希望读者批评指正文中不足的地方.

附录

A. F矩阵和G矩阵的推导

对子式子(4.7)

$$\dot{\tilde{x}}_I = F \tilde{x}_I + G n_I \quad \text{展开得到如下式子}$$

(我们先看结果再证明)

$$\begin{aligned} \textcircled{1} \quad & \dot{\tilde{x}}_I = \begin{pmatrix} -\hat{I}_3 & -I_3 & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & \tilde{I}_3 \tilde{\theta} \\ 0_{3 \times 3} & \tilde{g}_y \\ -C(\tilde{q} \tilde{\dot{q}})^T \tilde{a}_x & 0_{3 \times 3} & 0_{3 \times 3} & -C(\tilde{q} \tilde{\dot{q}})^T & 0_{3 \times 3} & 0_{3 \times 3} & \tilde{q} \tilde{v}_I \\ 0_{3 \times 3} & \tilde{b}_a \\ 0_{3 \times 3} & 0_{3 \times 3} & I_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & \tilde{g} \tilde{P}_I \\ 0_{3 \times 3} & \tilde{c} \tilde{P}_I \\ 0_{3 \times 3} & \tilde{I} \tilde{P}_I \end{pmatrix} \\ \textcircled{2} \quad & \dot{\tilde{g}}_y \\ \textcircled{3} \quad & \tilde{q} \tilde{v}_I \\ \textcircled{4} \quad & \tilde{b}_a \\ \textcircled{5} \quad & \tilde{g} \tilde{P}_I \\ \textcircled{6} \quad & \tilde{c} \tilde{P}_I \\ \textcircled{7} \quad & \tilde{I} \tilde{P}_I \\ + & \begin{pmatrix} -I_3 & 0_{3 \times 3} & n_g \\ 0_{3 \times 3} & I_3 & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & n_w \\ 0_{3 \times 3} & 0_{3 \times 3} & -C(\tilde{q} \tilde{\dot{q}})^T & 0_{3 \times 3} & 0_{3 \times 3} & I_3 & n_a \\ 0_{3 \times 3} & n_w \\ 0_{3 \times 3} & n_w \end{pmatrix} \end{aligned}$$

\tilde{x}	x 的误差项
\tilde{x}_m	x 的测量值
$\dot{\tilde{x}}$	x 的导数
\hat{x}	x 的估计值

一共需要证明7个式子, 其中, 式②和④这个是陀螺仪和加速度计 bias 漏差的导数, 这个直接对在 random walk 即 n_w 和 n_w . 式③和⑦是非漏差的导数, 一直是0.
式⑤ $\dot{\tilde{q}}_I = \tilde{q} \tilde{v}_I$ 这个也比较简单. 那么剩下 ① ⑥ ⑧ 证明如下.

式①证明: Hamilton 格式, JPL 的专利文献[2]

$$\text{真实状态 } \dot{\tilde{q}}_{t+} = \frac{1}{2} \tilde{q}_{t+} \otimes \tilde{w}_t.$$

$$\text{估计值 } \dot{\tilde{q}}_{\tilde{q}} = \frac{1}{2} \tilde{q}_{\tilde{q}} \otimes \tilde{w}.$$

$$\text{又: } \tilde{w} = \tilde{w}_t - b_g \quad \leftarrow \text{无偏}$$

$$\text{且有 } \tilde{w} = -\tilde{b}_g - n_g \quad \leftarrow \text{无偏}$$

$$\tilde{w}_t = \tilde{w} + \tilde{w} = \tilde{w}_t - b_g - \tilde{b}_g - n_g$$

真实值 bias 偏差 noise

然后对于真实的 $\dot{\tilde{q}}_{t+}$, 之后省略 \tilde{q} .

$$\begin{aligned} (\tilde{q} \otimes \tilde{q}) \Leftrightarrow \boxed{\dot{\tilde{q}}_t} & \Rightarrow \frac{1}{2} \tilde{q}_t \otimes \tilde{w}_t \\ \dot{\tilde{q}} \otimes \tilde{q} + \tilde{q} \otimes \dot{\tilde{q}} & = \frac{1}{2} \tilde{q} \otimes \tilde{q} \otimes \tilde{w}_t \\ \frac{1}{2} \tilde{q} \otimes \tilde{w} \otimes \tilde{q} + \tilde{q} \otimes \tilde{w} & = \\ \Rightarrow 2 \tilde{q} \otimes \tilde{q} & = \tilde{q} \otimes \tilde{q} \otimes \tilde{w}_t - \tilde{q} \otimes \tilde{w} \otimes \tilde{q} \\ \Rightarrow 2 \tilde{q} & = \begin{bmatrix} 0 \\ \tilde{q} \end{bmatrix} = \tilde{q} \otimes \tilde{w}_t - \tilde{w} \otimes \tilde{q} \end{aligned}$$

$$\begin{aligned}
&= q_R(w_0) \cdot \tilde{q} - q_L(w) \cdot \tilde{q} \\
&= [q_R(w_0) - q_L(\tilde{w})] \cdot \tilde{q} \\
&= \left(\begin{bmatrix} 0 & -w_0^T \\ w_0 & -[w_0]_x \end{bmatrix} - \begin{bmatrix} 0 & -\tilde{w}^T \\ \tilde{w} & -[\tilde{w}]_x \end{bmatrix} \right) \begin{bmatrix} 1 \\ \tilde{\theta}/2 \end{bmatrix} + O(\|\tilde{\theta}\|) \\
&= \begin{bmatrix} 0 & -(w_0 - \tilde{w})^T \\ w_0 - \tilde{w} & -[w_0 + \tilde{w}]_x \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{\theta}/2 \end{bmatrix} + O(\|\tilde{\theta}\|^2) \\
&= \begin{bmatrix} 0 & -\tilde{w}^T \\ \tilde{w} & -[2\tilde{w} + w]_x \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{\theta}/2 \end{bmatrix} \\
\Rightarrow \dot{\tilde{q}} &= \tilde{w} - [\tilde{w}]_x \tilde{\theta} - \frac{1}{2} [\tilde{w}]_x \tilde{\theta} + O(\|\tilde{\theta}\|^2) \quad \text{高阶} \\
\Rightarrow \dot{\tilde{\theta}} &= -[\tilde{w}]_x \tilde{\theta} + \tilde{w} \\
&= -[w_m - b_g] \tilde{\theta} - \tilde{b}_g - n_g
\end{aligned}$$

式②证明

$$\begin{aligned}
&\text{由题或(4.4)知} \quad R \\
&q_j = c\left(\begin{smallmatrix} I \\ \hat{a} \end{smallmatrix}\right)^T \hat{a} + {}^q g, \quad c\left(\begin{smallmatrix} I \\ \hat{a} \end{smallmatrix}\right)^T = c\left(\begin{smallmatrix} I \\ \hat{a} \end{smallmatrix}\right) \cdot c\left(\begin{smallmatrix} I \\ \hat{a} \end{smallmatrix}\right)^T \\
&\hat{a} = a_m - b_a \\
&\text{同理有 } \tilde{a} = -\tilde{b}_a - n_a \\
&\text{于是 } {}^q a_{\hat{a}} = c\left(\begin{smallmatrix} I \\ \hat{a} \end{smallmatrix}\right)^T (\hat{a} + \tilde{a}) + {}^q g \\
&= R(I + [\tilde{\theta}]_x)(\hat{a} + \tilde{a}) + {}^q g \\
&\Rightarrow \underbrace{{}^q \dot{\theta} + {}^q \dot{\tilde{\theta}}}_{R\hat{a} + {}^q g + {}^q \tilde{v}} = {}^q \dot{v}_t = \underbrace{\quad}_{\text{得证}}
\end{aligned}$$

$$\begin{aligned}
R\hat{a} + {}^q g + {}^q \tilde{v} &= R\hat{a} + R\tilde{a} + R[\tilde{\theta}]_x \hat{a} + R[\tilde{\theta}]_x \tilde{a} + {}^q g \\
{}^q \tilde{v} &= R(\hat{a} + [\tilde{\theta}]_x \hat{a} + [\tilde{\theta}]_x \tilde{a}) \quad \boxed{[a]_x b = -[b]_x a} \\
&= R(-[\hat{a}]_x \tilde{\theta} + \tilde{a}) \\
&= R(-[\hat{a}]_x \tilde{\theta} - \tilde{b}_a - n_a) \\
&= c\left(\begin{smallmatrix} I \\ \hat{a} \end{smallmatrix}\right)^T (-[\hat{a}]_x \tilde{\theta} - \tilde{b}_a - n_a) \quad \Rightarrow \text{得证.}
\end{aligned}$$

B. J_I 的计算

J_I 的计算与正文有点出入, 但还是先贴上来了, 希望哪位大佬能推导得到论文的结果并告知我一下.

$$\tilde{x} = \begin{bmatrix} {}^z\tilde{\theta} \\ {}^g\tilde{v}_I \\ {}^g\tilde{p}_I \\ {}^g\tilde{p}_{ci} \end{bmatrix}$$

根据式 4.12

$${}^g\tilde{q} = {}^z\tilde{q} \otimes {}^g\tilde{q}, {}^g\tilde{p}_c = {}^g\tilde{p}_I + C({}^z\tilde{q})^T {}^z\tilde{p}_c; \text{ 其中 } \tilde{x}_{ci} = ({}^g\tilde{\theta} \quad {}^g\tilde{p}_{ci})$$

$$J_I = \frac{\partial \tilde{x}_{ci}}{\partial \tilde{x}} = \begin{bmatrix} \frac{\partial {}^g\tilde{\theta}}{\partial {}^z\tilde{\theta}} & 0_{3 \times 3} & \frac{\partial {}^g\tilde{\theta}}{\partial {}^z\tilde{\theta}} & 0_{3 \times 3} \\ \frac{\partial {}^g\tilde{p}_{ci}}{\partial {}^z\tilde{\theta}} & 0_{3 \times 3} & \frac{\partial {}^g\tilde{p}_{ci}}{\partial {}^g\tilde{p}_I} & 0_{3 \times 3} \\ 0 & 0 & 0 & \frac{\partial {}^g\tilde{p}_{ci}}{\partial {}^z\tilde{p}_c} \end{bmatrix}$$

$$\begin{aligned} \text{我们先看 } {}^g\tilde{p}_c &= {}^g\tilde{p}_I + C({}^z\tilde{q})^T {}^z\tilde{p}_c \\ &= {}^g\tilde{p}_I + R {}^z\tilde{p}_c \end{aligned}$$

我们将其展开为递归式

$${}^g\tilde{p}_c + {}^g\tilde{p}_I = {}^g\tilde{p}_I + {}^g\tilde{p}_I + R({}^z\tilde{q} \otimes)({}^z\tilde{p}_c + {}^z\tilde{p}_c)$$

$$\begin{aligned} {}^g\tilde{p}_c &= {}^g\tilde{p}_I + R {}^z\tilde{p}_c + R({}^z\tilde{q} \otimes) {}^z\tilde{p}_c + \underbrace{R({}^z\tilde{q} \otimes) {}^z\tilde{p}_c}_{= \text{多余项}} \\ &= -R[{}^z\tilde{p}_c]_x \tilde{\theta} + {}^g\tilde{p}_I + R {}^z\tilde{p}_c \end{aligned}$$

$$= -C({}^z\tilde{q})^T [{}^z\tilde{p}_c]_x \tilde{\theta} + {}^g\tilde{p}_I + C({}^z\tilde{q})^T \tilde{p}_c$$

↑
多出来的.

$$\text{因此 } {}^g\tilde{q} = {}^z\tilde{q} \otimes {}^g\tilde{q}$$

这个我提出来的与论文中不太相符, 还希望哪位大佬能推一下.

C. H 矩阵

H 矩阵的计算

H 包含两部分, $H_{G_i}^j$ 和 $H_{G_i}^j$.

其中 $H_{G_i}^j$ 是 z_i^j 关于 error state \tilde{x} 的 Jacobian, $H_{G_i}^j$ 则是 z_i^j 关于 ${}^g\tilde{p}_c$ 的 Jacobian.

我们先看 $H_{G_i}^j \Rightarrow H_{G_i}^j$

$$H_{G_i}^j = \frac{\partial z_i^j}{\partial {}^{G_i}P_j} \cdot \frac{\partial {}^{G_i}P_j}{\partial x_{G_i,1}} + \frac{\partial z_i^j}{\partial {}^{G_i}P_j} \cdot \frac{\partial {}^{G_i,2}P_j}{\partial x_{G_i,1}}$$

$$4 \times 6 \quad 4 \times 3 \quad 3 \times 6 \quad 4 \times 3 \quad 3 \times 6$$

我们先看 $\frac{\partial z_i^j}{\partial {}^{G_i}P_j}$, 根据 4.14. 即 提到模型如何知

$$\frac{\partial z_i^j}{\partial {}^{G_i}P_j} = \begin{bmatrix} \frac{\partial u_{i,1}^j}{\partial {}^{G_i}y_1} & \dots & \frac{\partial u_{i,1}^j}{\partial {}^{G_i}y_f} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_{i,2}^j}{\partial {}^{G_i}y_1} & \dots & \frac{\partial u_{i,2}^j}{\partial {}^{G_i}y_f} \end{bmatrix}$$

$$= \frac{1}{c_{i,1} \hat{z}_j} \begin{pmatrix} 1 & 0 & -\frac{c_{i,1} \hat{x}_j}{c_{i,1} \hat{z}_j} \\ 0 & 1 & -\frac{c_{i,1} \hat{y}_j}{c_{i,1} \hat{z}_j} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

同理 $\frac{\partial z_j^i}{\partial c_{i,2} p_j} = \frac{1}{c_{i,2} \hat{z}_j} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & -\frac{c_{i,2} \hat{x}_j}{c_{i,2} \hat{z}_j} \\ 0 & 1 & -\frac{c_{i,2} \hat{y}_j}{c_{i,2} \hat{z}_j} \end{pmatrix}$

那么现在还剩下 $\frac{\partial c_{i,1} p_j}{\partial x_{c_{i,1}}} \text{ 和 } \frac{\partial c_{i,2} p_j}{\partial x_{c_{i,1}}}$

这里主要根据式 4.15.

$${}^g P_j = C \left(\frac{c_{i,1}}{g} q \right) ({}^g p_j - {}^g p_{c,1})$$

我们先看 $\frac{\partial c_{i,1} p_j}{\partial x_{c_{i,1}}}$

$${}^c p = \frac{c}{g} R ({}^g p - {}^g p_c)$$

根据参考文献 [2] 的式 (80) 可得

$$\begin{aligned} {}^c p + {}^c \tilde{p} &= (I - [\tilde{\theta}]_x) {}^g R ({}^g p - ({}^g p_c + {}^g \tilde{p}_c)) \\ &= {}^g R {}^g p - [\tilde{\theta}]_x {}^g R {}^g p - ({}^g R - [\tilde{\theta}]_x {}^g R) ({}^g p_c + {}^g \tilde{p}_c) \\ &= {}^g R {}^g p - [\tilde{\theta}]_x {}^g R {}^g p - ({}^g R {}^g p_c + {}^g R {}^g \tilde{p}_c - [\tilde{\theta}]_x {}^g R {}^g p_c - {}^g \tilde{p}_c) \end{aligned}$$

$$\begin{aligned} {}^c \tilde{p} &= -[\tilde{\theta}]_x {}^g R {}^g p - {}^g R {}^g \tilde{p}_c + [\tilde{\theta}]_x {}^g R {}^g p_c \\ &= -[\tilde{\theta}]_x \left(\frac{{}^g R {}^g p - {}^g R {}^g p_c}{={}^c p} \right) - {}^g R {}^g \tilde{p}_c \\ &= [{}^c p]_x \tilde{\theta} + (-{}^g R) {}^g \tilde{p}_c \end{aligned}$$

所以 $\frac{\partial c_{i,1} p_j}{\partial x_{c_{i,1}}} = ({}^c p)_x, -c \left(\frac{c_{i,1}}{g} \hat{q} \right)$

同理可得

$$\frac{\partial c_{i,2} p_j}{\partial x_{c_{i,1}}} = c \left(\frac{c_{i,1}}{c_{i,2}} q \right)^T \left([{}^c p]_x, -c \left(\frac{c_{i,1}}{g} \hat{q} \right) \right)$$

那么就最后

$$H_x^j = [0_{4 \times 2}, 0_{4 \times 6}, \dots, H_{c_i}^j, \dots]$$

\uparrow
第 i 个 pose.

然后我们再看 H_f^j

$$H_{ij}^j = \left(\frac{\partial z_i^j}{\partial c_{i1} p_j} \right) \frac{\partial c_{i1} p_j}{\partial q p_j} + \left(\frac{\partial z_i^j}{\partial c_{i2} p_j} \right) + \frac{\partial c_{i2} p_j}{\partial q p_j}$$

$$\frac{\partial c_{i1} p_j}{\partial q p_j} = C \left(\begin{matrix} c_{i1} \\ q \end{matrix} \right)$$

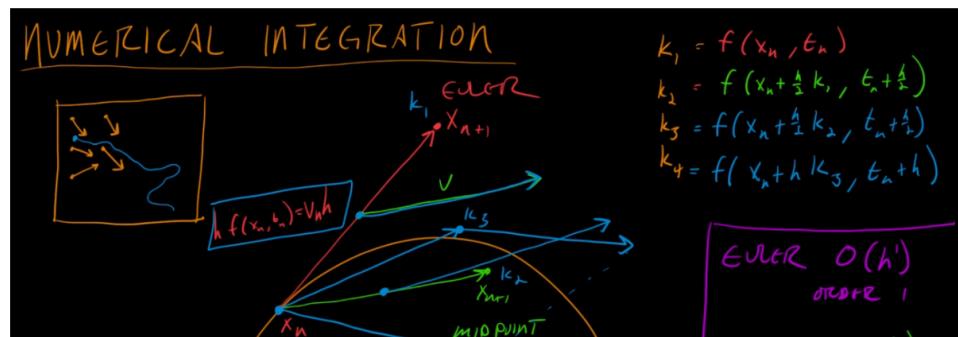
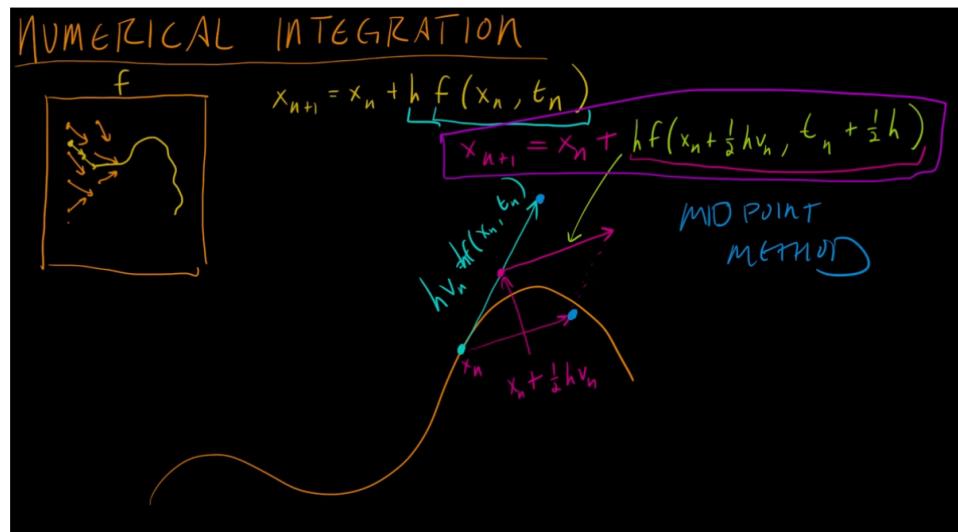
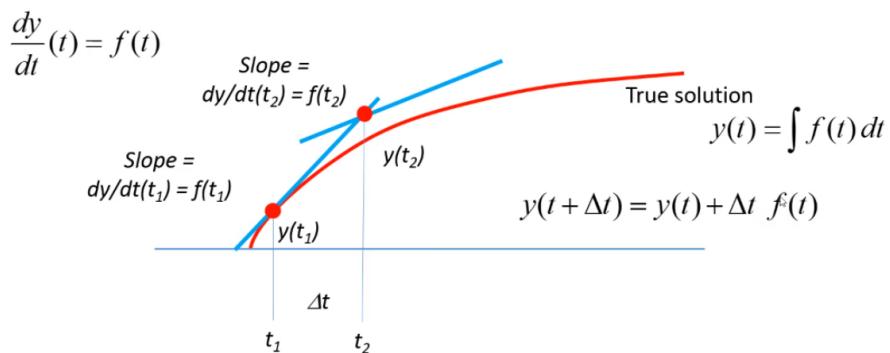
$$\frac{\partial c_{i2} p_j}{\partial q p_j} = C \left(\begin{matrix} c_{i1} \\ c_{i2} \\ q \end{matrix} \right)^T C \left(\begin{matrix} c_{i1} \\ q \end{matrix} \right)$$

然后再去看 S-MSCKF 话之 附录 C 和 D 就很明确了。

D. 三种常用数值积分方式: 欧拉, 中值, RK4 积分

这部分其实比较简单, 大家也可以参考 参考文献1中附录A部分, 也讲的很详细。这里简单附上三张图, 分别对应三种积分方式。

Euler Integration



$$x_{n+1} = x_n + \frac{1}{6} h \left(k_0 + 2k_1 + 2k_3 + k_4 \right)$$

MIDPOINT $O(h^2)$
 $h = 1$ ORDER 2

RK 4 $O(h^4)$
 $h = 1$ ORDER 4

7. 参考文献

- (1) [Quaternion kinematics for the error-state Kalman filter](#)
- (2) [Indirect Kalman Filter for 3D Attitude Estimation-A Tutorial for Quaternion Algebra](#)
- (3) [Consistency Analysis and IMprovement of Vision-aided Inertial Navigation](#)
- (4) [On the consistency of Vision-aided Inertial Navigation.](#)
- (5) [Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight](#)
- (6) [Monocular Visual Inertial Odometry on a Mobile Device](#)
- (7) [A multi-state constraint Kalman filter for vision-aided inertial navigation](#)
- (8) [视觉SLAM十四讲](#)
- (9) [卡尔曼滤波与组合导航原理](#)