深蓝学院
shenlanxueyuan.com

VINS-Mono代码讲解

主讲人 崔华坤

臻迪科技 SLAM负责人
2020.4

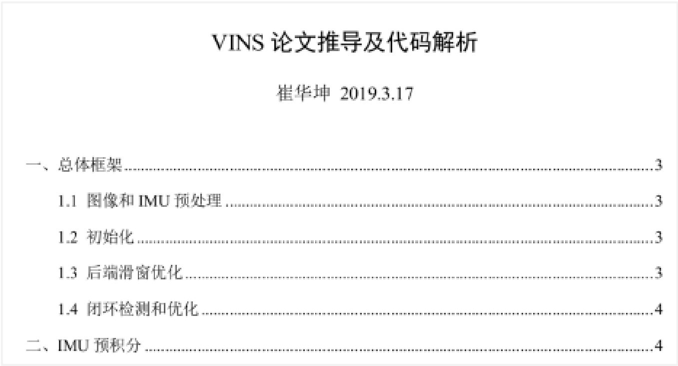
课程说明

1. 图解SLAM：用图让SLAM变得轻松、形象

2. 以介绍代码框架为主，VINS的原理、公式建议参考《从零开始手写VIO》和VINS文档[1]

3. 代码学习建议：不要像课堂式的学习，结合一个项目来实践代码，比如：

   - ❖ 把VINS中的ROS、闭环去掉，实现相对定位
   - ❖ VINS在Android手机上跑通
   - ❖ VINS在实际小车上跑通
   - ❖ 用特征点法替换掉LK光流，提高前端鲁棒性
   - ❖ 优化后端效率，提高帧率至20Hz



VINS 论文推导及代码解析

崔华坤 2019.3.17

[1] VINS论文推导及代码解析_V13_190317： https://github.com/StevenCui/VIO-Doc

目录

前端

- 图解VINS原理
- 代码流程图：从main函数开始说起
- 图解feature_tracker
- getMeasurements干了啥?
- processIMU：IMU预积分

后端

- processImage：每帧都干了什么?
- 谁是FeatureManager：维护路标点与图像
- 后端干了啥：详解因子图
- optimization：优化了谁?
- IMU约束、重投影约束：长什么模样?
- 麻烦的边缘化

初始化、闭环

- Camera与IMU的初始化：如何当一个好红娘?
- SfM与IMU预积分的羁绊
- 怎么知道发生了闭环?
- 闭环优化：拉扯橡皮条
- 位姿图优化与滑窗优化都为哪般?

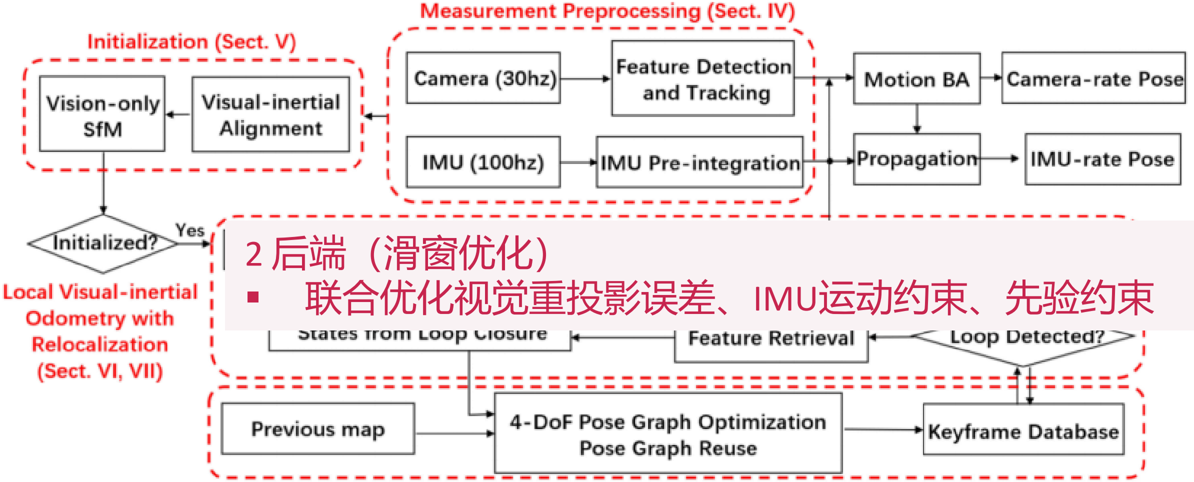
图解VINS原理

1 前端（数据预处理）

- 相邻帧的光流跟踪：建立当前帧与滑窗中其他帧的共视关系，即重投影误差约束
- Shi-Tomas特征点提取：保证一定数量的特征点
- IMU预积分：计算当前帧的位姿作为初始值，计算IMU约束的误差项、协方差和Jacobian

3 初始化

- 基于视觉信息来计算运动轨迹
- 基于IMU信息来计算运动轨迹
- 视觉和IMU松耦合来优化尺度、速度、重力



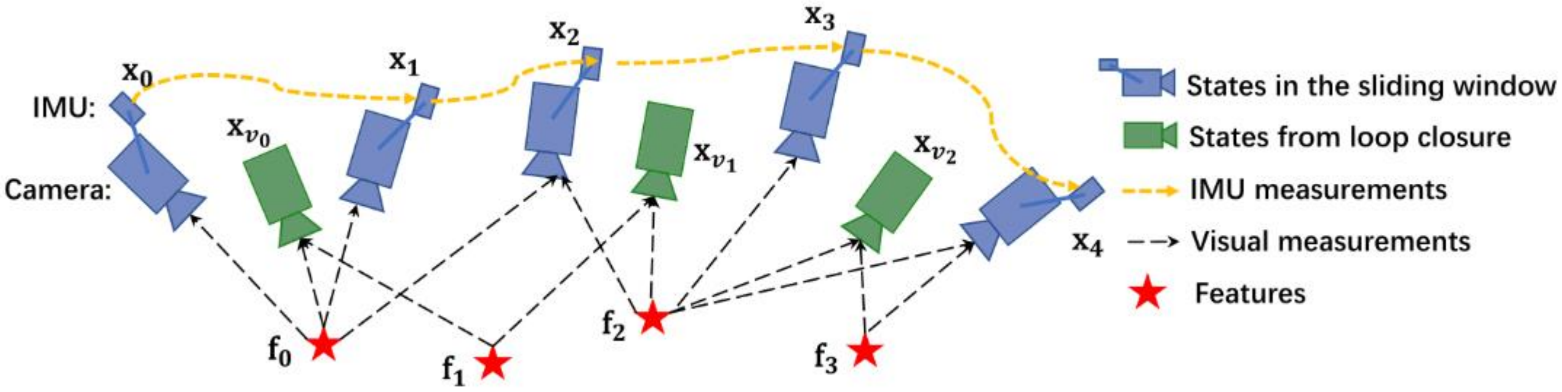
2 后端（滑窗优化）

- 联合优化视觉重投影误差、IMU运动约束、先验约束

4 闭环

- 通过词袋找闭环帧，通过暴力匹配进行特征点匹配，通过滑窗优化闭环帧的相对位姿
- 利用闭环帧的相对位姿进行轨迹的四自由度优化

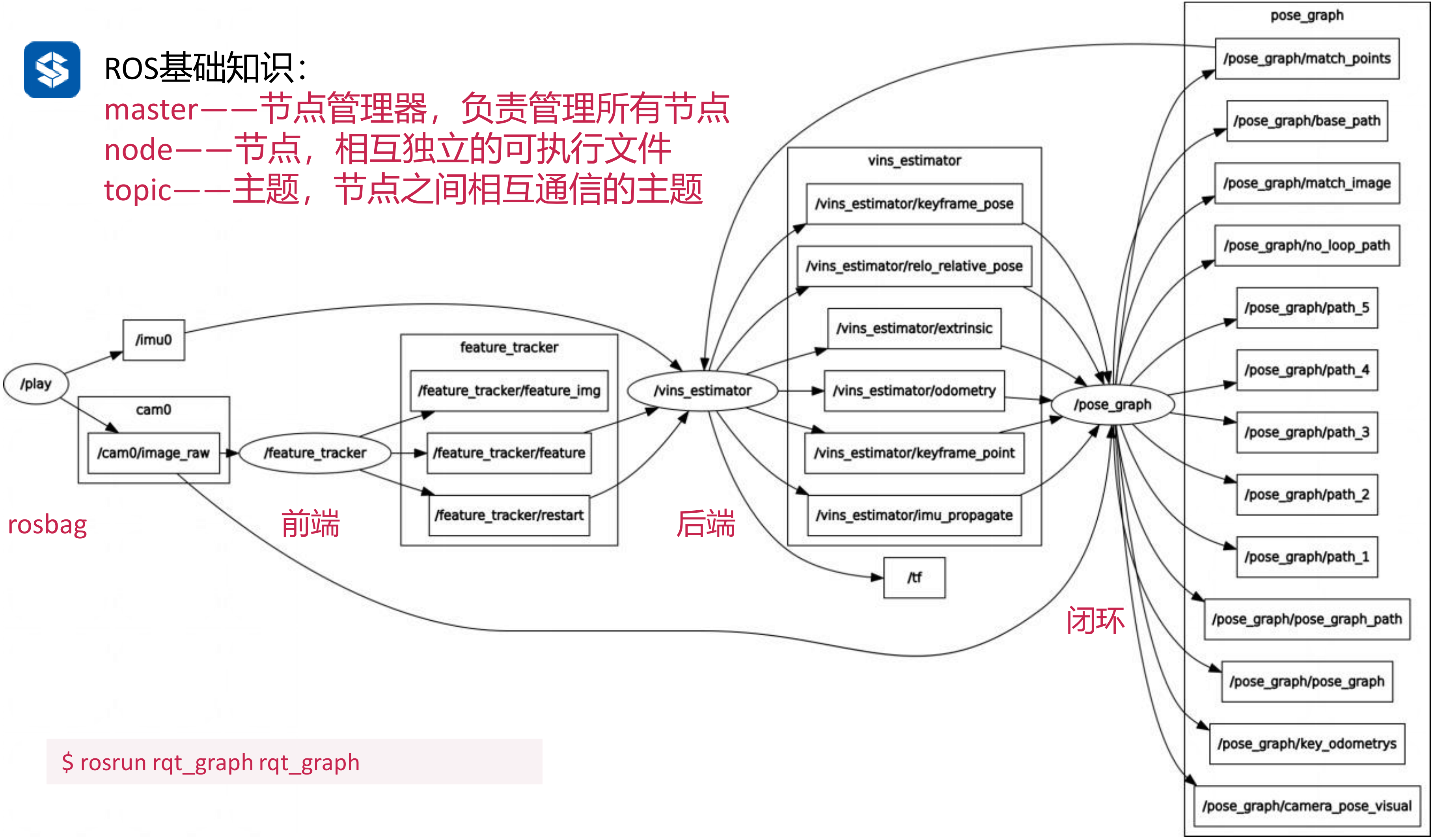
滑窗中的约束关系



1. 蓝色为视觉重投影误差约束（2×1）
2. 黄色为IMU帧间约束（PVQBaBg 15×1）
3. 绿色为闭环帧，用来计算相对位姿，用于闭环优化时使用

ROS基础知识：

master——节点管理器，负责管理所有节点
node——节点，相互独立的可执行文件
topic——主题，节点之间相互通信的主题



$ rosrun rqt_graph rqt_graph

ROS基础知识：rosrun、roslaunch、package、node

rosrun　　一次运行一个node
roslaunch　一次运行多个node
package　　一个功能包，包含多个node
node　　　一个节点就是package中的一个可执行文件

- 运行launch文件，以.launch为后缀名，放在Package的launch目录内。

```
$ roslaunch [PackageName] [LaunchFileName]
```

如何运行VINS-Mono：

```
$ roslaunch vins_estimator euroc.launch
$ roslaunch vins_estimator vins_rviz.launch
$ rosbag play YOUR_PATH_TO_DATASET/MH_01_easy.bag
```

euroc.launch：为了一次运行多个node

```xml
<launch>
    <arg name="config_path" default = "$(find feature_tracker)/../config/euroc/euroc_config.yaml" />
      <arg name="vins_path" default = "$(find feature_tracker)/../config/../" />

    <node name="feature_tracker" pkg="feature_tracker" type="feature_tracker" output="log">
        <param name="config_file" type="string" value="$(arg config_path)" />
        <param name="vins_folder" type="string" value="$(arg vins_path)" />
    </node>

    <node name="vins_estimator" pkg="vins_estimator" type="vins_estimator" output="screen">
       <param name="config_file" type="string" value="$(arg config_path)" />
       <param name="vins_folder" type="string" value="$(arg vins_path)" />
    </node>

    <node name="pose_graph" pkg="pose_graph" type="pose_graph" output="screen">
        <param name="config_file" type="string" value="$(arg config_path)" />
        <param name="visualization_shift_x" type="int" value="0" />
        <param name="visualization_shift_y" type="int" value="0" />
        <param name="skip_cnt" type="int" value="0" />
        <param name="skip_dis" type="double" value="0" />
    </node>

</launch>
```

node1：前端

node2：后端

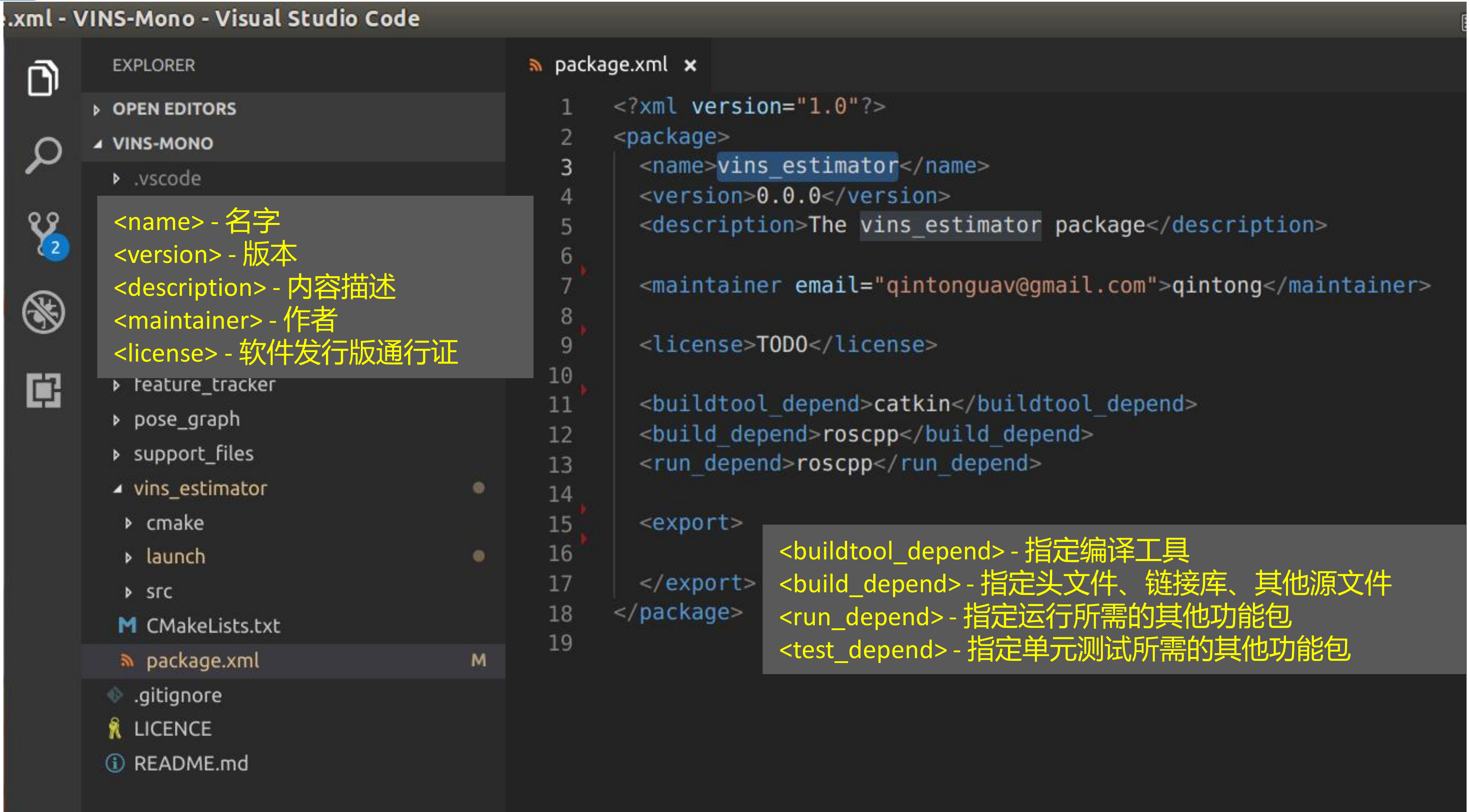
node3：闭环

pkg=package名字（在package.xml定义）

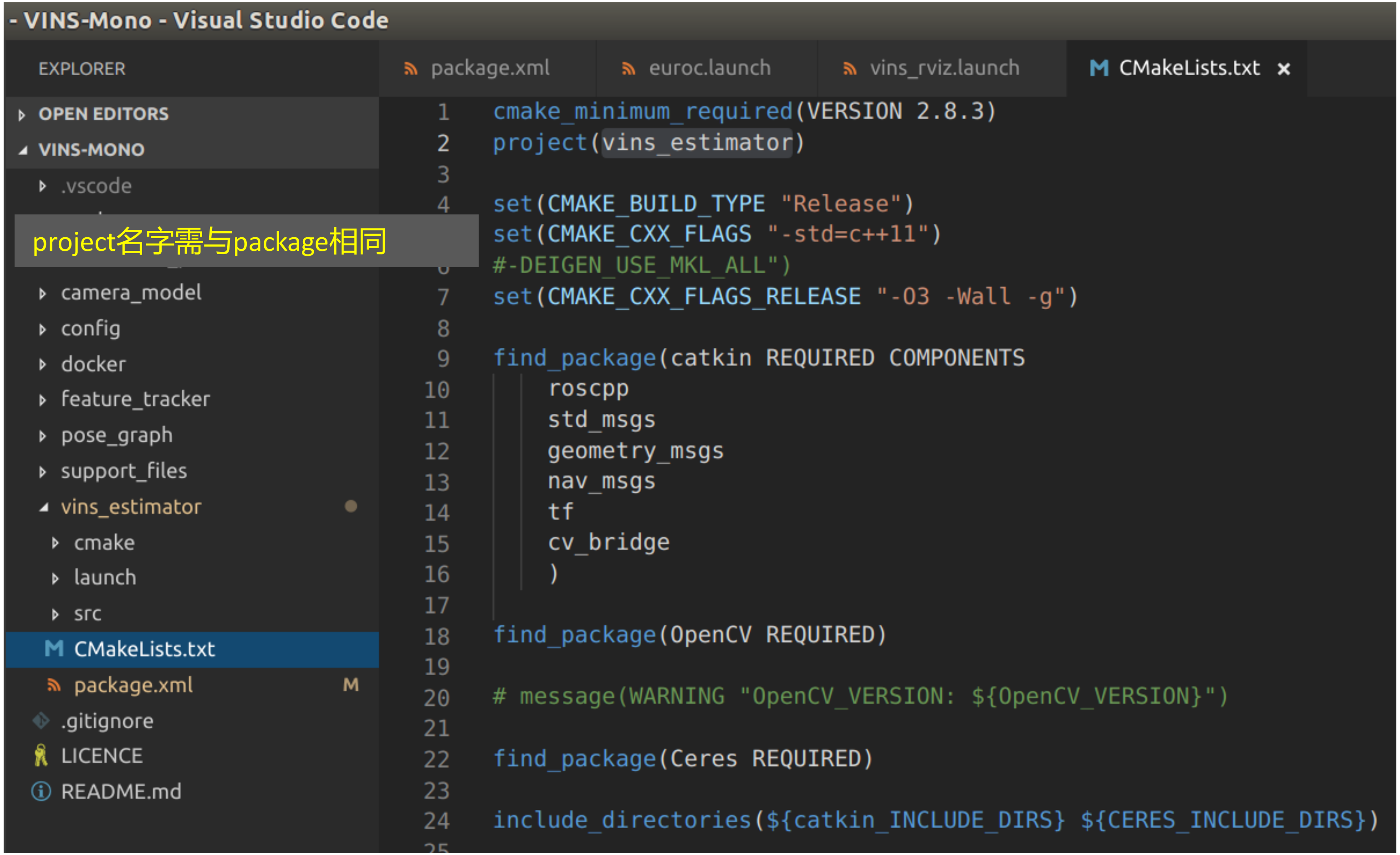
type=可执行文件名字（在CMakeLists.txt定义）

name=node名字，覆盖ros::init()

package.xml 功能包：



CMakeLists.txt：学习一个工程，从CMakeLists开始



project名字需与package相同

CMakeLists.txt

package.xml | euroc.launch | vins_rviz.launch | CMakeLists.txt

```
add_executable(vins_estimator
    src/estimator_node.cpp
    src/parameters.cpp
    src/estimator.cpp
    src/feature_manager.cpp
    src/factor/pose_local_parameterization.cpp
    src/factor/projection_factor.cpp
    src/factor/projection_td_factor.cpp
    src/factor/marginalization_factor.cpp
    src/utility/utility.cpp
    src/utility/visualization.cpp
    src/utility/CameraPoseVisualization.cpp
    src/initial/solve_5pts.cpp
    src/initial/initial_aligment.cpp
    src/initial/initial_sfm.cpp
    src/initial/initial_ex_rotation.cpp
    )


target_link_libraries(vins_estimator ${catkin_LIBRARIES} ${OpenCV_LIBS} ${CERES_LIBRARIES})
```

生成可执行文件

后端

各种约束关系

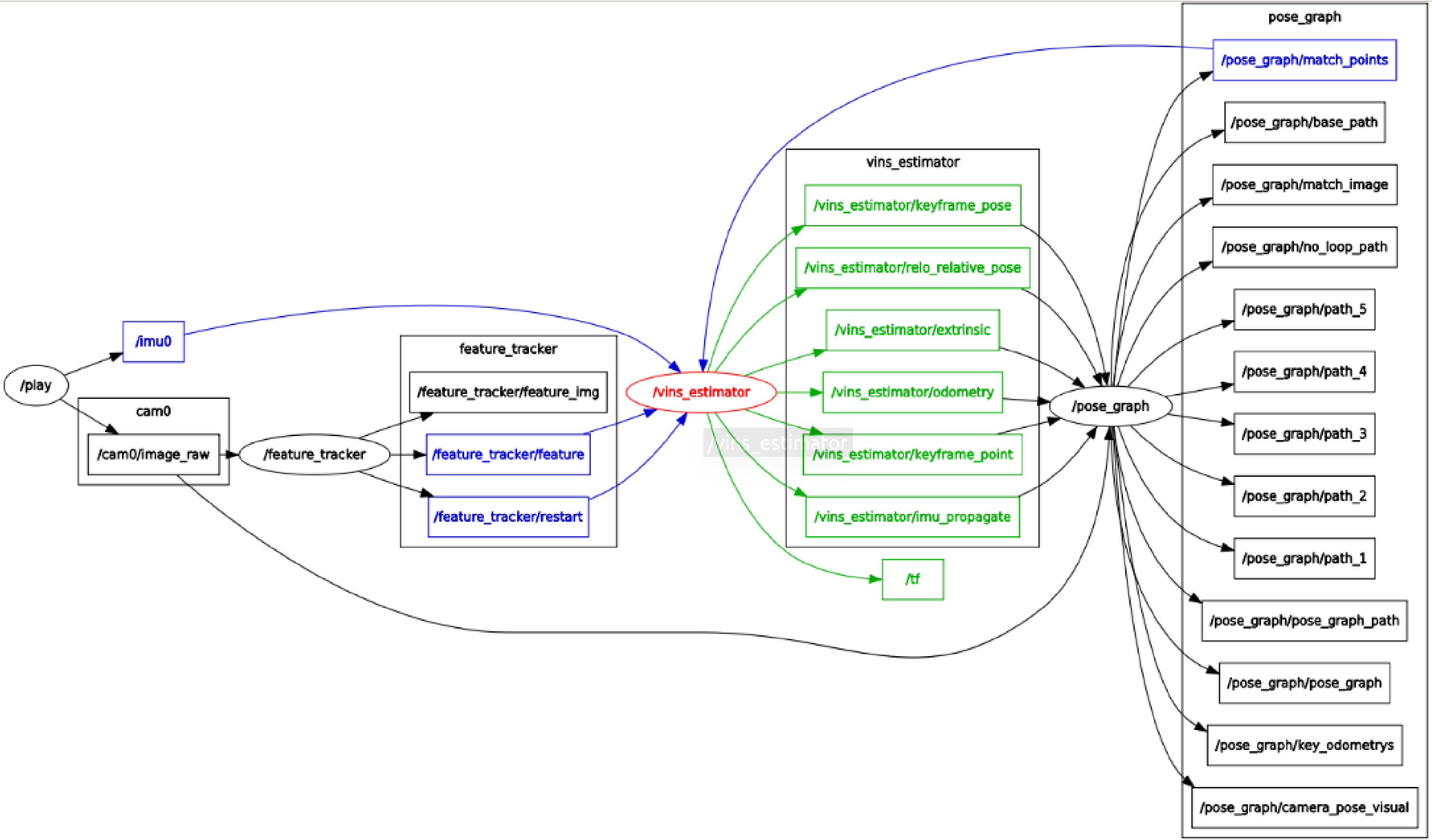
初始化

添加依赖库

配置文件：euroc_config.yaml

```
estimate_extrinsic: 0
max_cnt: 150            最大的点数 number in feature tracking
min_dist: 30            # min distance between two features
freq: 10                # frequence (Hz) of publish tracking result
F_threshold: 1.0        # ransac threshold (pixel)
equalize: 1             # if image is too dark or light, trun on equalize to find enough features
#optimization parameters
max_solver_time: 0.04  # 最大的求解时间 time (ms), to guarantee real time
max_num_iterations: 8   # max solver itrations, to guarantee real time
keyframe_parallax: 10.0 # keyframe selection threshold (pixel)
#imu parameters       The more accurate parameters you provide, the better performance
acc_n: 0.08          # accelerometer measurement noise standard deviation. #0.2   0.04
gyr_n: 0.004         # gy IMU噪声 urement noise standard deviation.     #0.05  0.004
acc_w: 0.00004         # r bias random work noise standard deviation.  #0.02
gyr_w: 2.0e-6       # gyroscope bias random work noise standard deviation.     #4.0e-5

#unsynchronization parameters
estimate_td: 0          IMU和Cam的时间戳同步 time offset between camera and imu
td: 0.0                                   time offset. unit: s. readed image clock +
td = real image clock (IMU clock)

#rolling shutter parameters
rolling_shutter: 0                  # 0: global shutter camera, 1: rolling shutter camera
rolling_shutter_tr: 0               # unit: s. rolling shutter read out time per frame (from data
sheet).
```

代码流程图：从后端的main函数开始说起



代码流程图：从后端的main函数开始说起

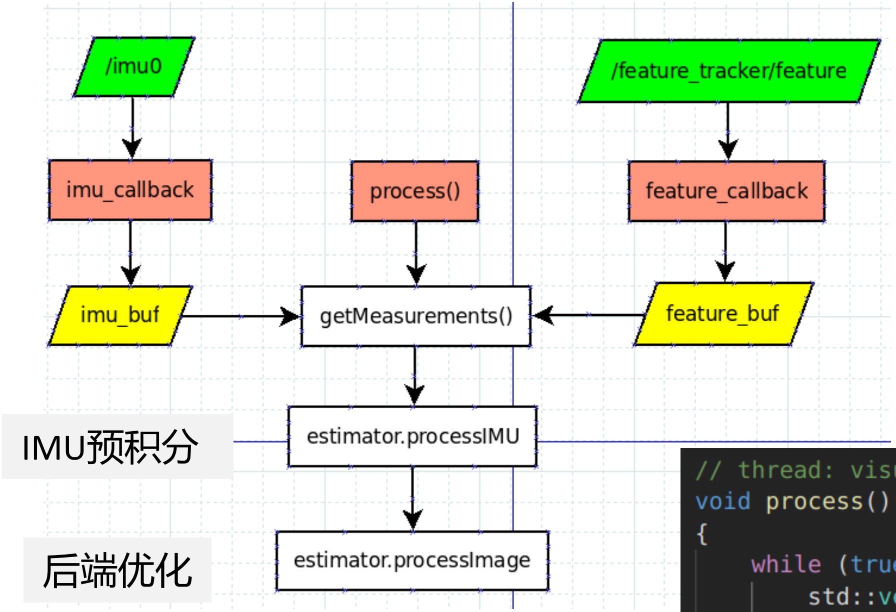
```
int main(int argc, char **argv)
{
    ros::init(argc, argv, "vins_estimator");
    ros::NodeHandle n("~");
    ros::console::set_logger_level(ROSCONSOLE_DEFAULT_NAME, ros::console::levels::Info);
    readParameters(n);
    estimator.setParameter();
#ifdef EIGEN_DONT_PARALLELIZE
    ROS_DEBUG("EIGEN_DONT_PARALLELIZE");
#endif
    ROS_WARN("waiting for image and imu...");

    registerPub(n);

    ros::Subscriber sub_imu = n.subscribe(IMU_TOPIC, 2000, imu_callback, ros::TransportHints().tcpNoDelay());
    ros::Subscriber sub_image = n.subscribe("/feature_tracker/feature", 2000, feature_callback);
    ros::Subscriber sub_restart = n.subscribe("/feature_tracker/restart", 2000, restart_callback);
    ros::Subscriber sub_relo_points = n.subscribe("/pose_graph/match_points", 2000, relocalization_callback);

    std::thread measurement_process{process};
    ros::spin();

    return 0;
}
```

接收imu消息

接收feature消息

后端线程

代码流程图：从main函数开始说起（estimator_node.cpp）




```cpp
void feature_callback(const sensor_m
{
    if (!init_feature)
    {
        //skip the first detected fe
        init_feature = 1;
        return;
    }
    m_buf.lock();
    feature_buf.push(feature_msg);
    m_buf.unlock();
    con.notify_one();
}
```

```cpp
// thread: visual-inertial odometry
void process()
{
    while (true) {
        std::vector<std::pair<std::vector<sensor_msgs::ImuConstPtr>,
            sensor_msgs::PointCloudConstPtr>> measurements;
        std::unique_lock<std::mutex> lk(m_buf);
        con.wait(lk, [&] {
            return (measurements = getMeasurements()).size() != 0;
        });
        lk.unlock();
```

一个C++知识点：线程互斥锁

http://www.cplusplus.com/

class

std::mutex C++11

<mutex>

```
class mutex;
```

Mutex class

A *mutex* is a *lockable object* that is designed to signal when critical sections of code need exclusive access, preventing other threads with the same protection from executing concurrently and access the same memory locations.

`mutex` objects provide *exclusive ownership* and do not support recursivity (i.e., a thread shall not lock a `mutex` it already owns) -- see `recursive_mutex` for an alternative class that does.

It is guaranteed to be a standard-layout class.

Member types

member type	description
`native_handle_type`	Type returned by `native_handle` (only defined if library implementation supports it)

Member functions

(constructor)	Construct mutex (public member function)
lock	Lock mutex (public member function)
try_lock	Lock mutex if not locked (public member function)
unlock	Unlock mutex (public member function)
native_handle	Get native handle (public member function)

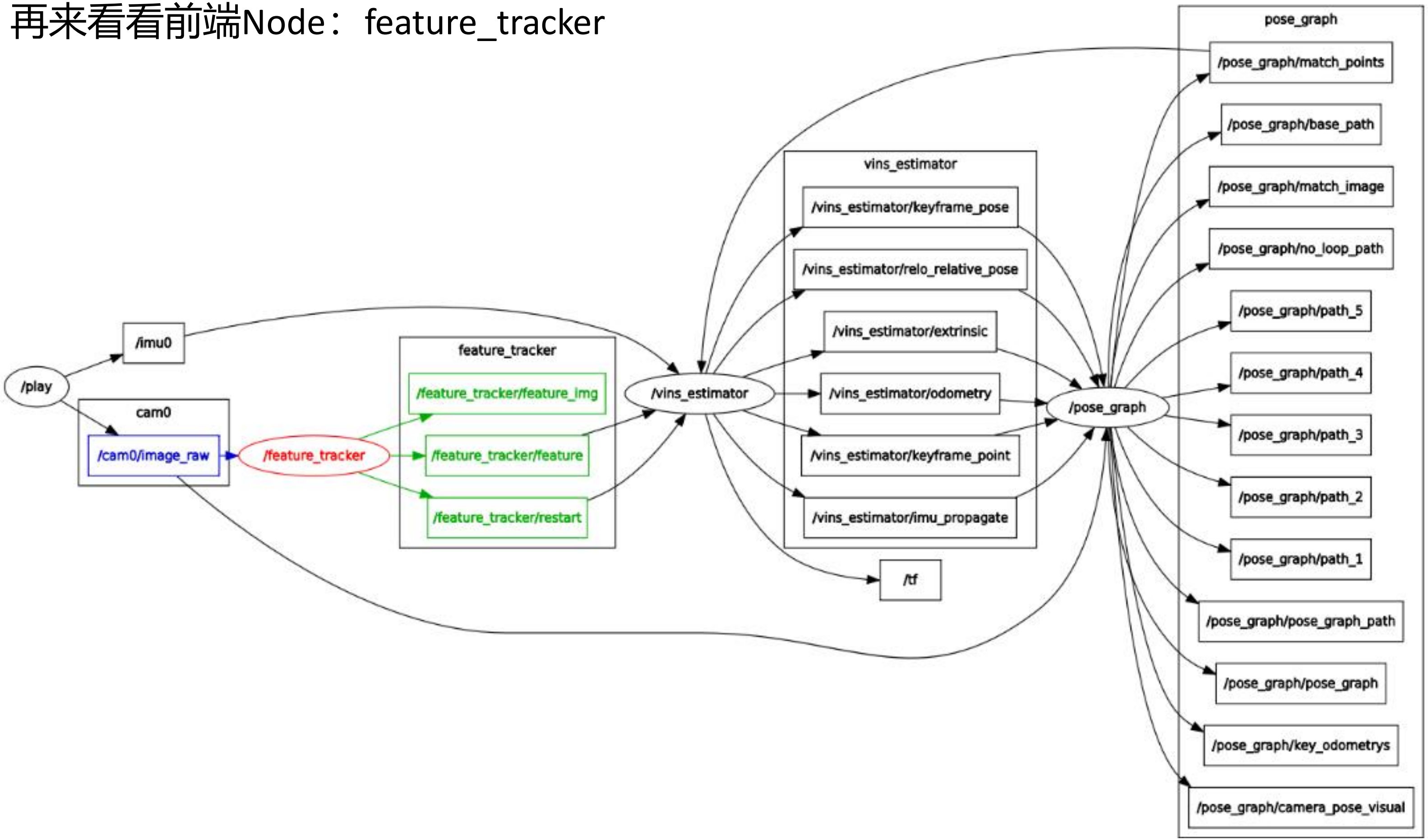
一个C++知识点：线程互斥锁

mutex 多线程互斥锁 m_buf
condition_variable 条件变量 con

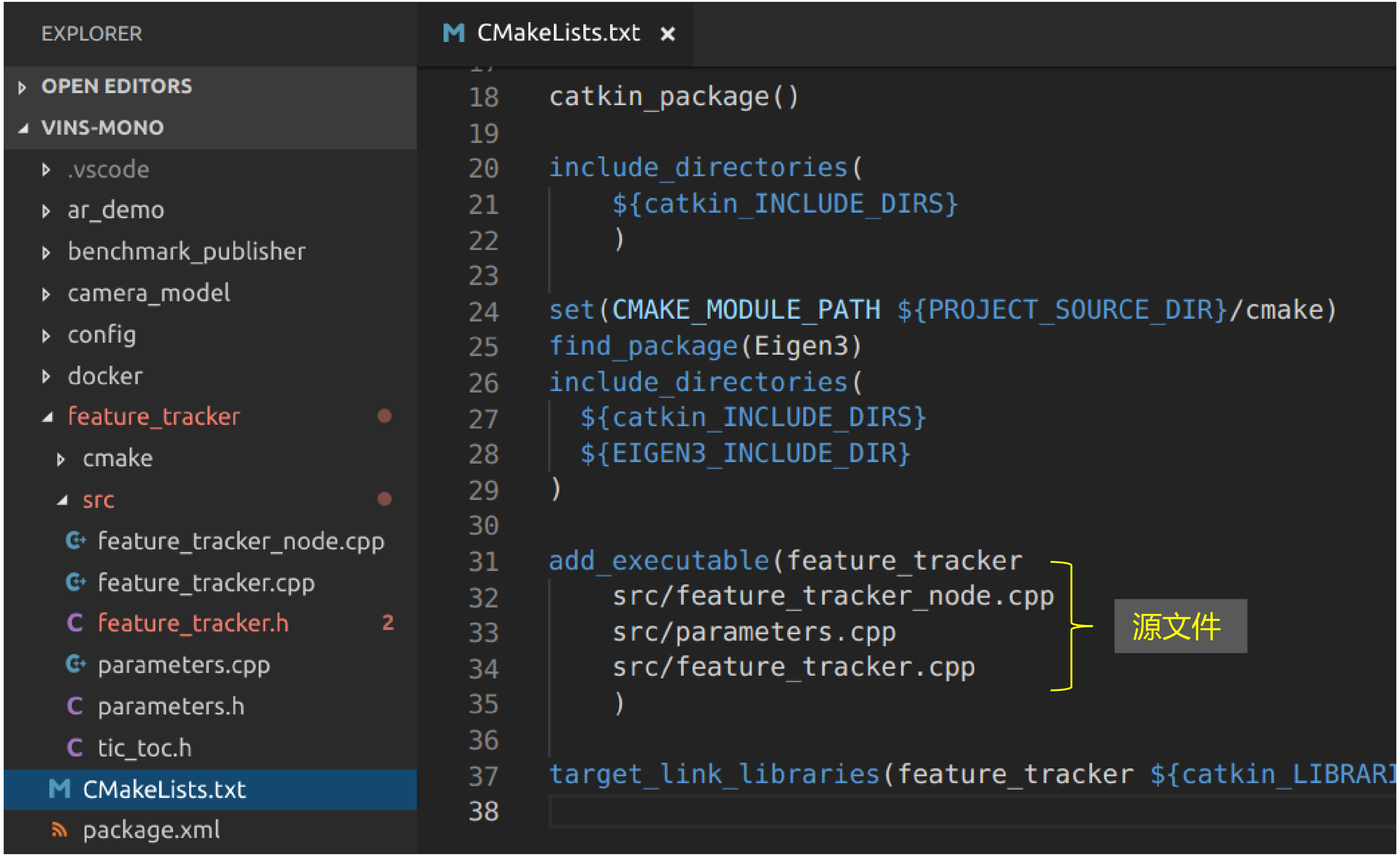
```
// thread: visual-inertial odometry
void process()
{
    while (true)
    {
        std::vector<std::pair<std::vector<sensor_msgs::ImuConstPtr>,
        measurements;
        std::unique_lock<std::mutex> lk(m_buf);
        con.wait(lk, [&]{
            return (measurements = getMeasurements()).size() != 0;
            });
        lk.unlock();
```

- IMU、图像、后端三个线程共用一个m_buf
- 后端process()线程调用wait()，自动调用m_buf.lock()来加锁；
- 若getMeasurements未得到图像和两帧之间的IMU，则返回false，线程被阻塞，此时，wait()会自动调用m_buf.unlock()来释放锁，使得imu和feature的回调线程继续push；
- imu和feature的回调线程，每拿到数据，都会调用con.notify_one唤醒process()线程，使其继续尝试getMeasurements。

再来看看前端Node：feature_tracker



再来看看前端Node：feature_tracker



前端main()：启动前端回调 img_callback()

feature_tracker_node.cpp

```cpp

int main(int argc, char **argv)
{
    ros::init(argc, argv, "feature_tracker");
    ros::NodeHandle n("~");
    ros::console::set_logger_level(ROSCONSOLE_DEFAULT_NAME, ros::console::levels::Info);
    readParameters(n);

    for (int i = 0; i < NUM_OF_CAM; i++)
        trackerData[i].readIntrinsicParameter(CAM_NAMES[i]);

    if(FISHEYE)
    {⋯
    }

    ros::Subscriber sub_img = n.subscribe(IMAGE_TOPIC, 100, img_callback);

    pub_img = n.advertise<sensor_msgs::PointCloud>("feature", 1000);
    pub_match = n.advertise<sensor_msgs::Image>("feature_img",1000);
    pub_restart = n.advertise<std_msgs::Bool>("restart",1000);
    /*
    if (SHOW_TRACK)
        cv::namedWindow("vis", cv::WINDOW_NORMAL);
    */
    ros::spin();
    return 0;
}
```

前端回调

readImage()

createCLAHE
自适应局部直方图均衡化

↓

calcOpticalFlowPyrLK
光流跟踪

↓

rejectWithF
基础矩阵剔除异常点

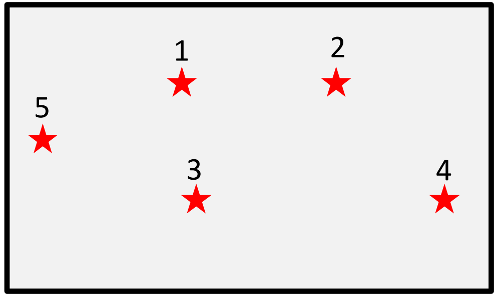
↓

setMask
不在已有点附近提取

↓

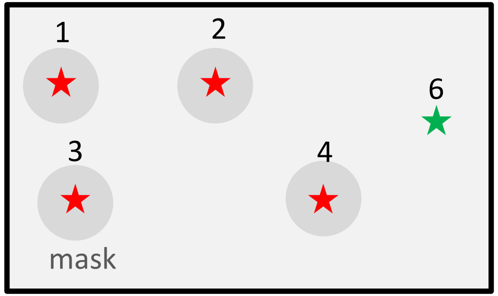
goodFeaturesToTrack
提取Shi-Tomas角点

cur_pts



cur_img

forw_pts



forw_img：5跟丢，
新提特征点6

cur_img　上一帧图像
forw_img 当前帧图像
cur_pts　上一帧的点坐标
forw_pts　当前帧的点坐标
ids　每个点的id号
track_cnt 每个点被跟踪的次数
cur_un_pts 最新帧的归一化相机系坐标

readImage()[假设每帧总点数为5]

第1帧	track_cnt:	1	1	1	1	1
		0	1	2	3	4
第2帧		2	2	2	2	1
		0	1	2	4	5
第3帧		3	3	3	2	1
		1	2	4	5	6
第4帧		4	4	3	2	1
		2	4	5	6	7
第5帧		5	4	3	1	1
		4	5	6	8	9

自适应局部直方图均衡化

直方图均衡化可增强图像对比度，主要思想是将一副图像的直方图分布变成近似均匀分布。

CLAHE

```
if (EQUALIZE)
{
    cv::Ptr<cv::CLAHE> clahe = cv::createCLAHE(3.0, cv::Size(8, 8));
    TicToc t_c;
    clahe->apply(_img, img);
    ROS_DEBUG("CLAHE costs: %fms", t_c.toc());
}
```

calcOpticalFlowPyrLK() 光流跟踪

```
vector<uchar> status;
vector<float> err;
cv::calcOpticalFlowPyrLK(cur_img, forw_img, cur_pts, forw_pts, status, err, cv::Size(21, 21), 3);
```

void **calcOpticalFlowPyrLK(**prevImg, nextImg, prevPts, nextPts, status, err, winSize, maxLevel, TermCriteria criteria, flags, minEigThreshold)

说明：

maxLevel=3 金字塔层数

flags=OPTFLOW_USE_INITIAL_FLOW 设置下一帧特征点的初始位置，若forw_pts为空，则表示与cur_pts相同

基础矩阵——异常点剔除

```
vector<uchar> status;
cv::findFundamentalMat(un_cur_pts, un_forw_pts, cv::FM_RANSAC, F_THRESHOLD, 0.99, status);
```

Mat **findFundamentalMat**(points1, points2, method, param1=3., param2=0.99, mask)

作用：根据两帧匹配点，计算基础矩阵，用于异常点剔除

说明：
method:
- CV_FM_7POINT for a 7-point algorithm.
- CV_FM_8POINT for an 8-point algorithm.
- **CV_FM_RANSAC** for the RANSAC algorithm.
- CV_FM_LMEDS for the LMedS algorithm.

goodFeaturesToTrack 特征点提取

```
cv::goodFeaturesToTrack(forw_img, n_pts, MAX_CNT - forw_pts.size(), 0.01, MIN_DIST, mask);
```

void **goodFeaturesToTrack**(image, corners, maxCorners, qualityLevel, minDistance, mask, blockSize, useHarrisDetector, k)

作用：提取Shi-Tomas角点

说明:
maxCorners 最大点数
minDistance 两点之间允许最小距离
mask 避免在老点附近提取

前端发布出去的消息

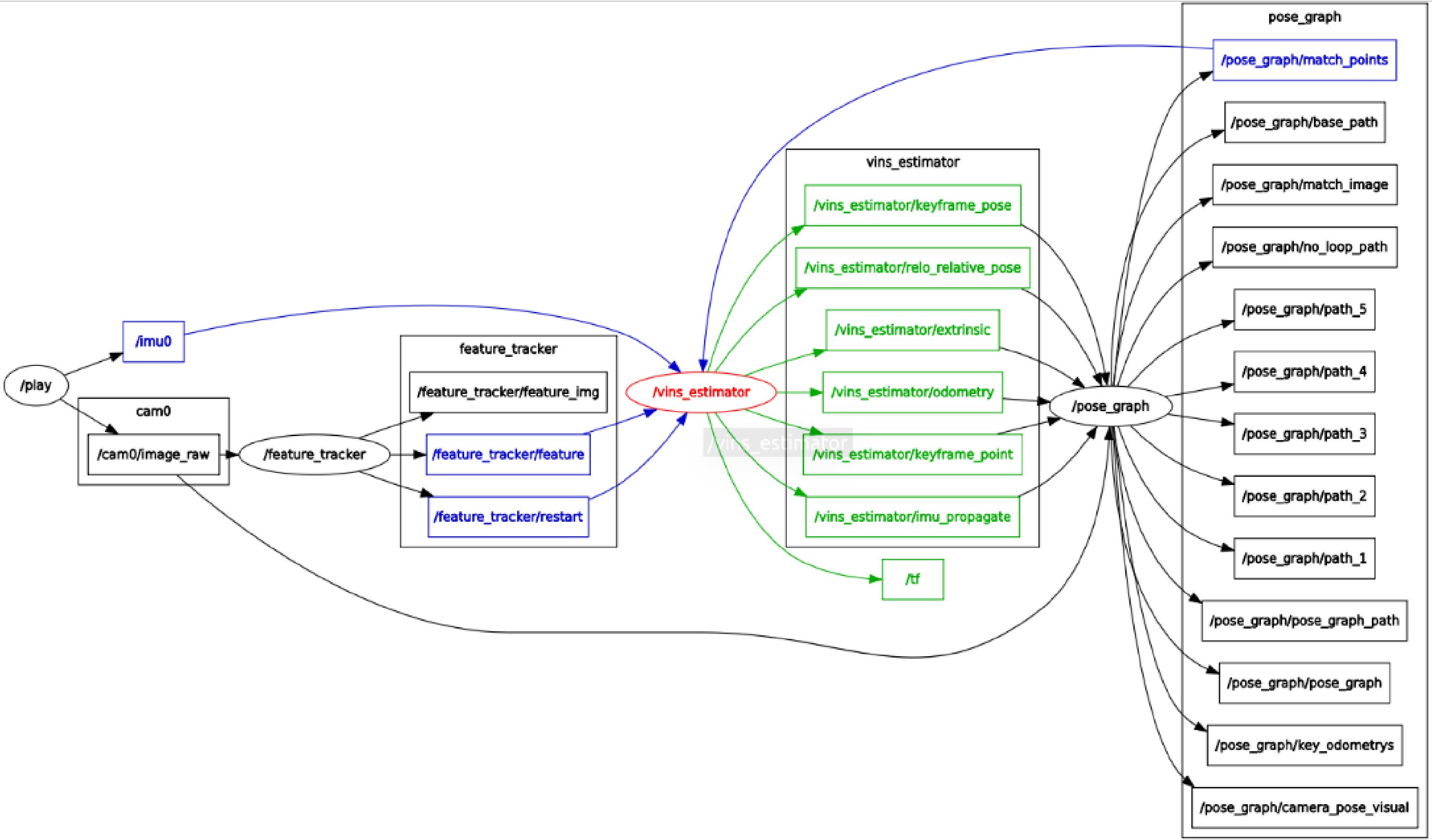
```
ros::Subscriber sub_img = n.subscribe(IMAGE_TOPIC, 100, img_callback);

pub_img = n.advertise<sensor_msgs::PointCloud>("feature", 1000);
pub_match = n.advertise<sensor_msgs::Image>("feature_img", 1000);
pub_restart = n.advertise<std_msgs::Bool>("restart", 1000);
```

```
stevencui@stevencui: ~
stevencui@stevencui:~$ rosmsg show PointCloud
[sensor_msgs/PointCloud]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Point32[] points
  float32 x
  float32 y
  float32 z
sensor_msgs/ChannelFloat32[] channels
  string name
  float32[] values
```

[sensor_msgs/**PointCloud**]:

- header 时间戳
  - -seq
  - -stamp
  - -frame_id
- points 归一化相机系坐标
  - -x、y、z
- channels (id、u、v、v_x、v_y)
  - -name
  - -values

前端发布特征点跟踪信息给到后端



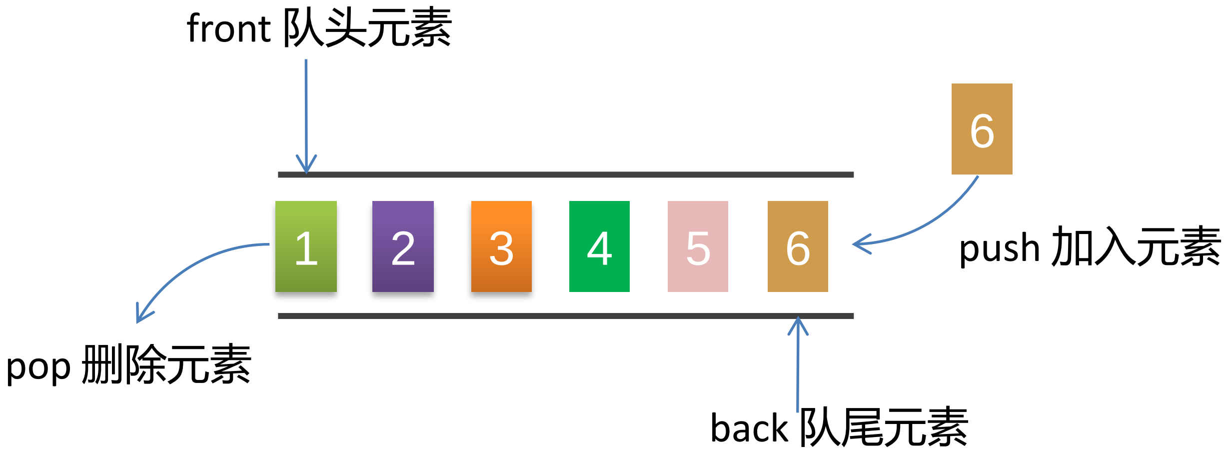
getMeasurements干了啥?

```cpp
// thread: visual-inertial odometry
void process()
{
    while (true) {
        std::vector<std::pair<std::vector<sensor_msgs::ImuConstPtr>,
            sensor_msgs::PointCloudConstPtr>> measurements;
        std::unique_lock<std::mutex> lk(m_buf);
        con.wait(lk, [&] {
            return (measurements = getMeasurements()).size() != 0;
        });
        lk.unlock();
```

根据时间戳，挑选当前帧和上一帧图像之间的imu数据，用于后续的IMU积分

```cpp
std::vector<std::pair<std::vector<sensor_msgs::ImuConstPtr>, sensor_msgs::PointCloudConstPtr>>
getMeasurements()
{
    std::vector<std::pair<std::vector<sensor_msgs::ImuConstPtr>, sensor_msgs::PointCloudConstPtr>> measurements;

    while (true) {
        if (imu_buf.empty() || feature_buf.empty())
            return measurements;

        if (!(imu_buf.back()->header.stamp.toSec() > feature_buf.front()->header.stamp.toSec() + estimator.td)) {
            //ROS_WARN("wait for imu, only should happen at the beginning");
            sum_of_wait++;
            return measurements;
        }

        if (!(imu_buf.front()->header.stamp.toSec() < feature_buf.front()->header.stamp.toSec() + estimator.td)) {
            ROS_WARN("throw img, only should happen at the beginning");
            feature_buf.pop();
            continue;
        }
        sensor_msgs::PointCloudConstPtr img_msg = feature_buf.front();
        feature_buf.pop();

        std::vector<sensor_msgs::ImuConstPtr> IMUs;
        while (imu_buf.front()->header.stamp.toSec() < img_msg->header.stamp.toSec() + estimator.td) {
            IMUs.emplace_back(imu_buf.front());
            imu_buf.pop();
        }
        IMUs.emplace_back(imu_buf.front());
        if (IMUs.empty())
            ROS_WARN("no imu between two image");
        measurements.emplace_back(IMUs, img_msg);
    }
```

一个C++知识点：queue



queue
“先进先出”的单向队列，只能从“后面”压进(Push)元素，从“前面”提取(Pop)元素

fx **Member functions**

(constructor)	Construct queue (public member function)
empty	Test whether container is empty (public member function)
size	Return size (public member function)
front	Access next element (public member function)
back	Access last element (public member function)
push	Insert element (public member function)
emplace C++11	Construct and insert element (public member function)
pop	Remove next element (public member function)
swap C++11	Swap contents (public member function)

```cpp
std::vector<std::pair<std::vector<sensor_msgs::ImuConstPtr>, sensor_msgs::PointCloudConstPtr>>
getMeasurements()
{
    std::vector<std::pair<std::vector<sensor_msgs::ImuConstPtr>, sensor_msgs::PointCloudConstPtr>> measurements;

    while (true) {
        if (imu_buf.empty() || feature_buf.empty())
            return measurements;

        if (!(imu_buf.back()->header.stamp.toSec() > feature_buf.front()->header.stamp.toSec() + estimator.td)) {
            //ROS_WARN("wait for imu, only should happen at the beginning");
            sum_of_wait++;
            return measurements;
        }

        if (!(imu_buf.front()->header.stamp.toSec() < feature_buf.front()->header.stamp.toSec() + estimator.td)) {
            ROS_WARN("throw img, only should happen at the beginning");
            feature_buf.pop();
            continue;
        }
        sensor_msgs::PointCloudConstPtr img_msg = feature_buf.front();
        feature_buf.pop();

        std::vector<sensor_msgs::ImuConstPtr> IMUs;
        while (imu_buf.front()->header.stamp.toSec() < img_msg->header.stamp.toSec() + estimator.td) {
            IMUs.emplace_back(imu_buf.front());
            imu_buf.pop();
        }
        IMUs.emplace_back(imu_buf.front());
        if (IMUs.empty())
            ROS_WARN("no imu between two image");
        measurements.emplace_back(IMUs, img_msg);
    }
```

imu或图像为空

imu太慢，等等imu

图像太老，扔掉图像

挑选两帧之间的imu

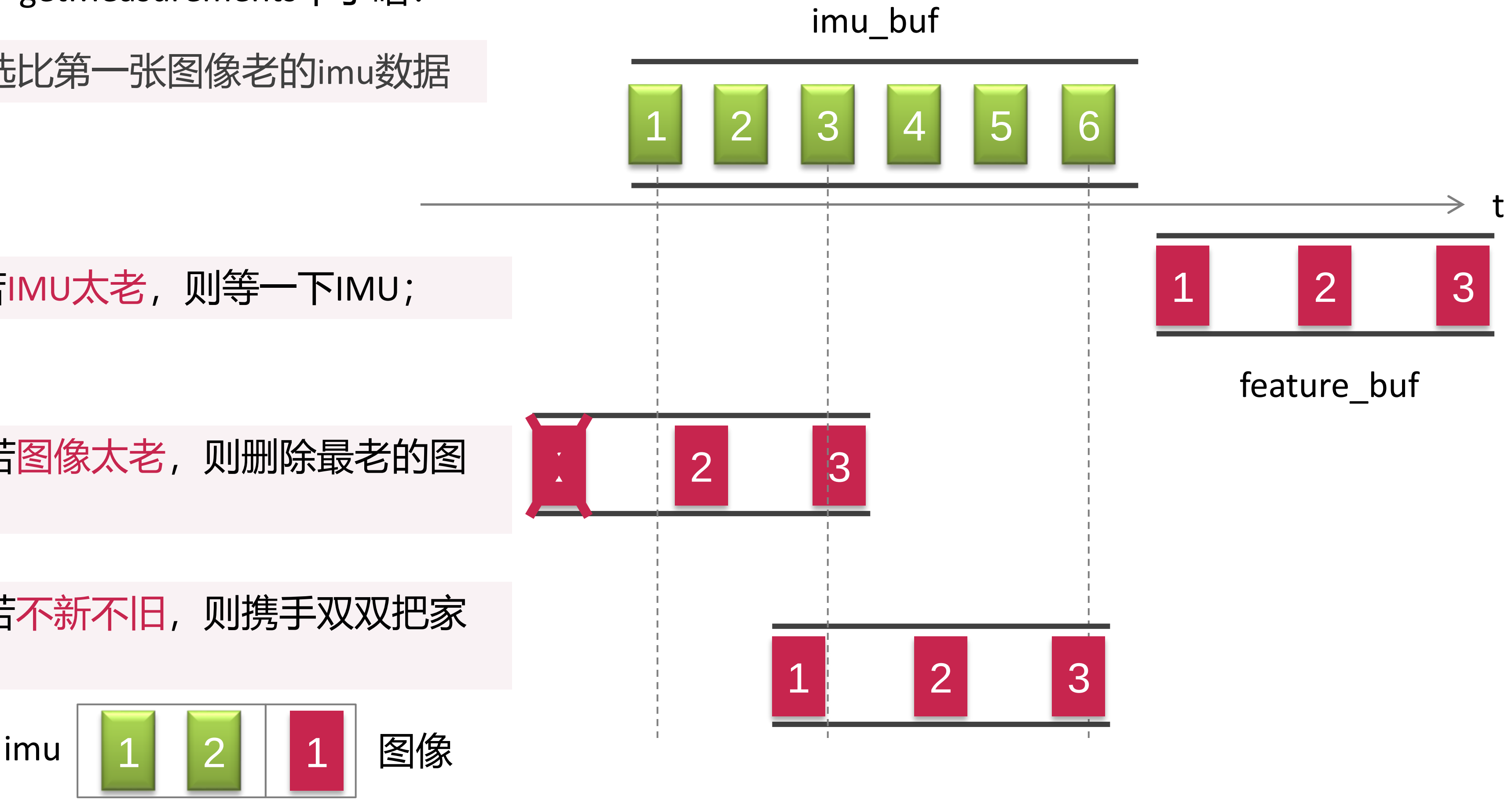
getMeasurements干了啥?

挑选比第一张图像老的imu数据

1 若IMU太老，则等一下IMU；

2. 若图像太老，则删除最老的图像；

3. 若不新不旧，则携手双双把家还。

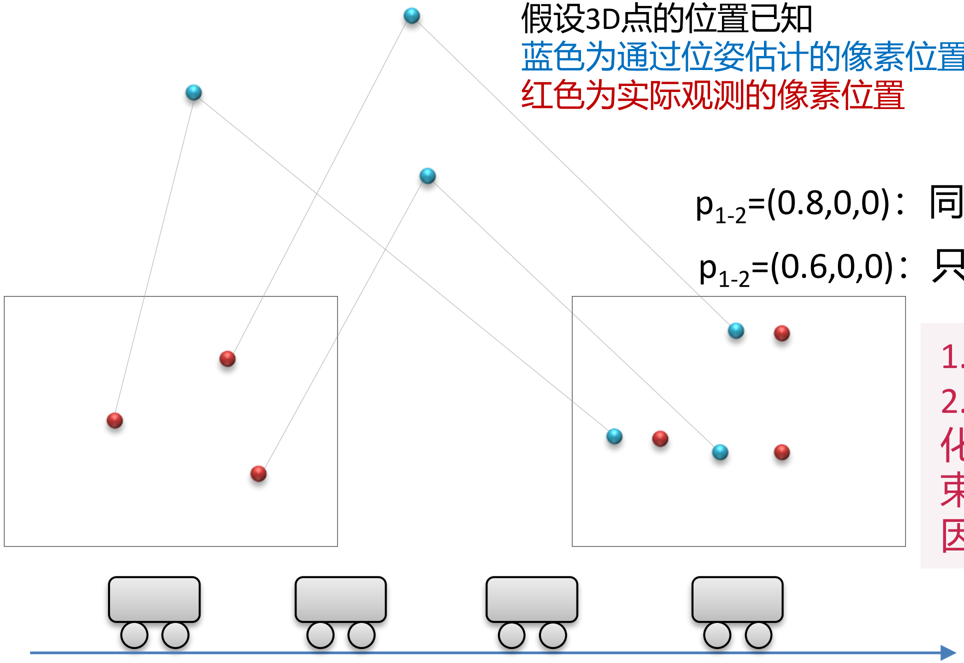


视觉和IMU的联合优化

假设3D点的位置已知
蓝色为通过位姿估计的像素位置
红色为实际观测的像素位置

$p_{1\text{-}2}=(0.8,0,0)$：同时信视觉和IMU

$p_{1\text{-}2}=(0.6,0,0)$：只信视觉



1. 如果只进行重投影约束，可得到$p_{1\text{-}2}=(0.6,0,0)$
2. 若联合优化重投影约束和IMU的运动约束，则优化到$p_{1\text{-}2}=(0.8,0,0)$，相信谁多一些，取决于两种约束的协方差。

因此，需要计算两种约束的观测和协方差。

令t1时刻的速度$v_1=0$，假设小车以$a=(2,0,0)$加速向右运动

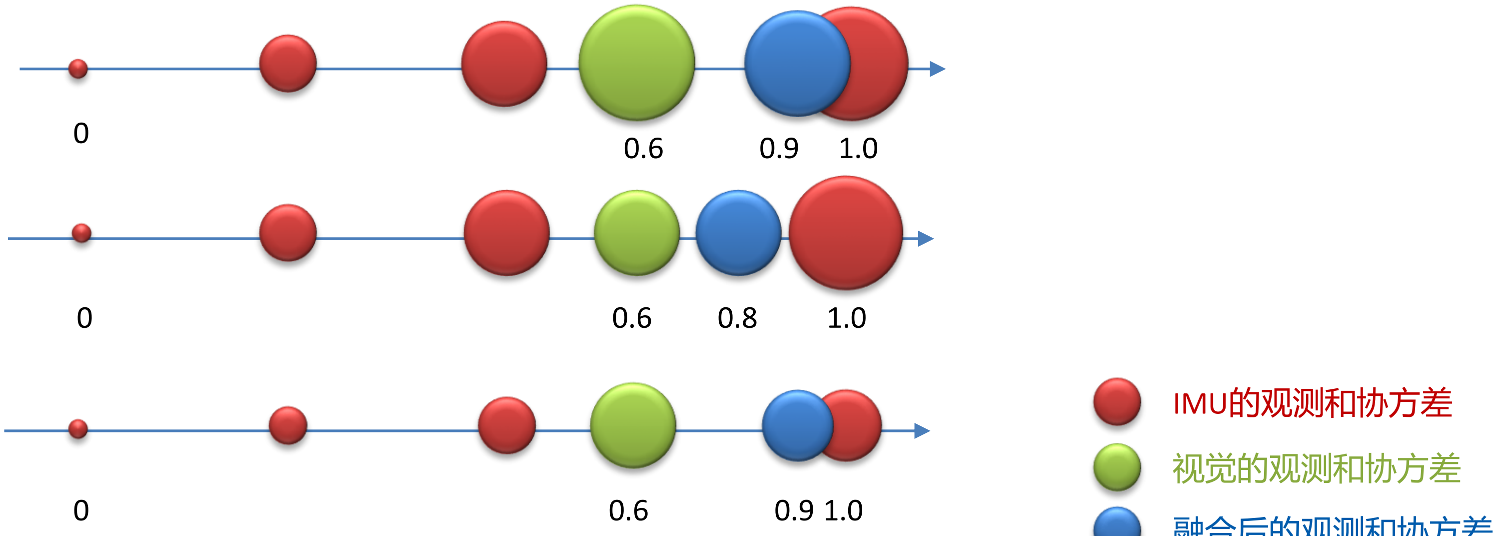
1s后，通过IMU从t1积分到t2，得到$p_{1\text{-}2}=(1,0,0)$

IMU预积分作用：计算IMU观测的误差、协方差、Jacobian

协方差的根源是因为imu有噪声，因此随着预测的增加，观测的不确定度会越来越大，即协方差越來越大；
当来了视觉的观测后，系统会根据两者的协方差大小，来权衡更相信谁。

视觉噪声大时，更相信IMU，所以误差大小是相对的

IMU噪声小时，更相信IMU



processIMU：IMU预积分

```
IntegrationBase* pre_integrations[(WINDOW_SIZE + 1)];
```

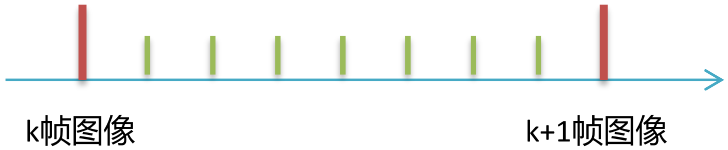
```
void Estimator::processIMU(double dt, const Vector3d& linear_acceleration, const Vector3d& angular_velocity)
{
    if (!first_imu) {
        first_imu = true;
        acc_0 = linear_acceleration;
        gy
    }

    if (!pr
        pre_integrations[frame_count] = new IntegrationBase{ acc_0, gyr_0, Bas[frame_count], Bgs[frame_count] };
    }
    if (frame_count != 0) {
        pre_integrations[frame_count]->push_back(dt, linear_acceleration, angular_velocity);
        //if(solver_flag != NON_LINEAR)
        tmp_pre_integration->push_back(dt, linear_acceleration, angular_velocity);

        dt_buf[frame_count].push_back(dt);
        linear_acceleration_buf[frame_count].push_back(linear_acceleration);
        angular_velocity_buf[frame_count].push_back(angular_velocity);

        int j = frame_count;
        Vector3d un_acc_0 = Rs[j] * (acc_0 - Bas[j]) - g;
        Vector3d un_gyr = 0.5 * (gyr_0 + angular_velocity) - Bgs[j];
        Rs[j] *= Utility::deltaQ(un_gyr * dt).toRotationMatrix();
        Vector3d un_acc_1 = Rs[j] * (linear_acceleration - Bas[j]) - g;
        Vector3d un_acc = 0.5 * (un_acc_0 + un_acc_1);
        Ps[j] += dt * Vs[j] + 0.5 * dt * dt * un_acc;
        Vs[j] += dt * un_acc;
    }
    acc_0 = linear_acceleration;
    gyr_0 = angular_velocity;
}
```

push_back进行预积分，根据两帧之间的IMU信息，计算两帧之间的位置、旋转、速度的变化量，作为IMU约束的误差项，并根据IMU的噪声大小，计算IMU约束的协方差（即不确定度）

中值积分预测最新帧的位姿，作为视觉的初始位姿

$$R_{w \leftarrow b} = R_s$$

IntegrationBase::push_back() IMU预积分



push_back(dt, acc, gyr)

↓

propagate

↓

midPointIntegration
更新IMU约束的误差项、协方差、Jacobian

```cpp
void push_back(double dt, const Eigen::Vector3d& acc, const Eigen::Vector3d& gyr)
{
    dt_buf.push_back(dt);
    acc_buf.push_back(acc);
    gyr_buf.push_back(gyr);
    propagate(dt, acc, gyr);
}
```

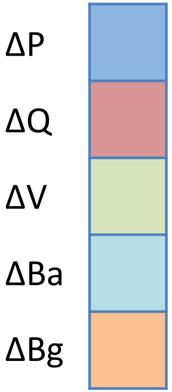
```cpp
void propagate(double _dt, const Eigen::Vector3d& _acc_1, const Eigen::Vector3d& _gyr_1)
{
    dt = _dt;
    acc_1 = _acc_1;
    gyr_1 = _gyr_1;
    Vector3d result_delta_p;
    Quaterniond result_delta_q;
    Vector3d result_delta_v;
    Vector3d result_linearized_ba;
    Vector3d result_linearized_bg;

    midPointIntegration(_dt, acc_0, gyr_0, _acc_1, _gyr_1, delta_p, delta_q, delta_v,
        linearized_ba, linearized_bg,
        result_delta_p, result_delta_q, result_delta_v,
        result_linearized_ba, result_linearized_bg, 1);
```

IMU预积分：
midPointIntegration更新IMU约束的误差项

```
void midPointIntegration(double _dt,
    const Eigen::Vector3d& _acc_0, const Eigen::Vector3d& _gyr_0,
    const Eigen::Vector3d& _acc_1, const Eigen::Vector3d& _gyr_1,
    const Eigen::Vector3d& delta_p, const Eigen::Quaterniond& delta_q, const Eigen::Vector3d& delta_v,
    const Eigen::Vector3d& linearized_ba, const Eigen::Vector3d& linearized_bg,
    Eigen::Vector3d& result_delta_p, Eigen::Quaterniond& result_delta_q, Eigen::Vector3d&
    result_delta_v,
    Eigen::Vector3d& result_linearized_ba, Eigen::Vector3d& result_linearized_bg, bool update_jacobian)
{
    //ROS_INFO("midpoint integration");
    Vector3d un_acc_0 = delta_q * (_acc_0 - linearized_ba);
    Vector3d un_gyr = 0.5 * (_gyr_0 + _gyr_1) - linearized_bg;
    result_delta_q = delta_q * Quaterniond(1, un_gyr(0) * _dt / 2, un_gyr(1) * _dt / 2, un_gyr(2) *
    _dt / 2);
    Vector3d un_acc_1 = result_delta_q * (_acc_1 - linearized_ba);
    Vector3d un_acc = 0.5 * (un_acc_0 + un_acc_1);
    result_delta_p = delta_p + delta_v * _dt + 0.5 * un_acc * _dt * _dt;
    result_delta_v = delta_v + un_acc * _dt;
```

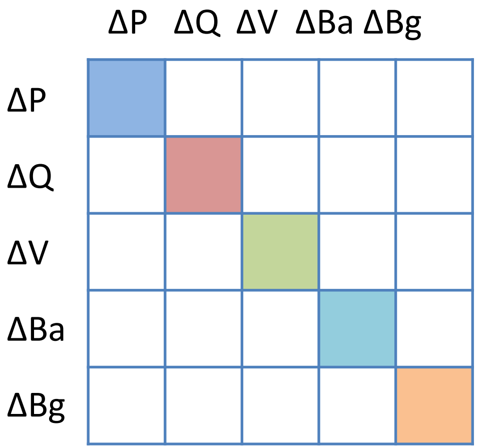
```
Eigen::Matrix<double, 15, 1> evaluate(const Eigen::Vector3d& Pi, const Eigen::Quaterniond& Qi,
    const Eigen::Vector3d& Vi, const Eigen::Vector3d& Bai, const Eigen::Vector3d& Bgi,
    const Eigen::Vector3d& Pj, const Eigen::Quaterniond& Qj, const Eigen::Vector3d& Vj,
    const Eigen::Vector3d& Baj, const Eigen::Vector3d& Bgj)
{
    Eigen::Matrix<double, 15, 1> residuals;

    residuals.block<3, 1>(O_P, 0) = Qi.inverse() * (0.5 * G * sum_dt * sum_dt + Pj - Pi - Vi * sum_dt)
    - corrected_delta_p;
    residuals.block<3, 1>(O_R, 0) = 2 * (corrected_delta_q.inverse() * (Qi.inverse() * Qj)).vec();
    residuals.block<3, 1>(O_V, 0) = Qi.inverse() * (G * sum_dt + Vj - Vi) - corrected_delta_v;
    residuals.block<3, 1>(O_BA, 0) = Baj - Bai;
    residuals.block<3, 1>(O_BG, 0) = Bgj - Bgi;

    return residuals;
}
```

误差项

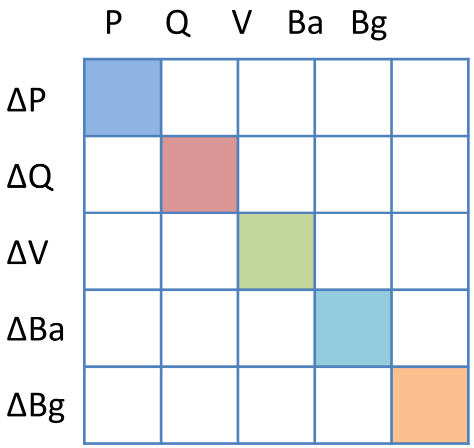


IMU预积分：
midPointIntegration更新IMU约束的协方差、Jacobian

```
jacobian = F * jacobian;
covariance = F * covariance * F.transpose() + V * noise * V.transpose();
```



covariance协方差：
表示IMU误差的不确定度



IMU误差关于优化变量的Jacobian

IntegrationBase： IMU预积分

协方差的更新公式：
确实很复杂，推导一遍，弄懂后
作为工具拿来主义即可

```
MatrixXd F = MatrixXd::Zero(15, 15);
F.block<3, 3>(0, 0) = Matrix3d::Identity();
F.block<3, 3>(0, 3) = -0.25 * delta_q.toRotationMatrix() * R_a_0_x * _dt * _dt + -0.25 *
result_delta_q.toRotationMatrix() * R_a_1_x * (Matrix3d::Identity() - R_w_x * _dt) * _dt * _dt;
F.block<3, 3>(0, 6) = MatrixXd::Identity(3, 3) * _dt;
F.block<3, 3>(0, 9) = -0.25 * (delta_q.toRotationMatrix() + result_delta_q.toRotationMatrix())
* _dt * _dt;
F.block<3, 3>(0, 12) = -0.25 * result_delta_q.toRotationMatrix() * R_a_1_x * _dt * _dt * -_dt;
F.block<3, 3>(3, 3) = Matrix3d::Identity() - R_w_x * _dt;
F.block<3, 3>(3, 12) = -1.0 * MatrixXd::Identity(3, 3) * _dt;
F.block<3, 3>(6, 3) = -0.5 * delta_q.toRotationMatrix() * R_a_0_x * _dt + -0.5 *
result_delta_q.toRotationMatrix() * R_a_1_x * (Matrix3d::Identity() - R_w_x * _dt) * _dt;
F.block<3, 3>(6, 6) = Matrix3d::Identity();
F.block<3, 3>(6, 9) = -0.5 * (delta_q.toRotationMatrix() + result_delta_q.toRotationMatrix())
* _dt;
F.block<3, 3>(6, 12) = -0.5 * result_delta_q.toRotationMatrix() * R_a_1_x * _dt * -_dt;
F.block<3, 3>(9, 9) = Matrix3d::Identity();
F.block<3, 3>(12, 12) = Matrix3d::Identity();
//cout<<"A"<<endl<<A<<endl;

MatrixXd V = MatrixXd::Zero(15, 18);
V.block<3, 3>(0, 0) = 0.25 * delta_q.toRotationMatrix() * _dt * _dt;
V.block<3, 3>(0, 3) = 0.25 * -result_delta_q.toRotationMatrix() * R_a_1_x * _dt * _dt * 0.5 *
_dt;
V.block<3, 3>(0, 6) = 0.25 * result_delta_q.toRotationMatrix() * _dt * _dt;
V.block<3, 3>(0, 9) = V.block<3, 3>(0, 3);
V.block<3, 3>(3, 3) = 0.5 * MatrixXd::Identity(3, 3) * _dt;
V.block<3, 3>(3, 9) = 0.5 * MatrixXd::Identity(3, 3) * _dt;
V.block<3, 3>(6, 0) = 0.5 * delta_q.toRotationMatrix() * _dt;
V.block<3, 3>(6, 3) = 0.5 * -result_delta_q.toRotationMatrix() * R_a_1_x * _dt * 0.5 * _dt;
V.block<3, 3>(6, 6) = 0.5 * result_delta_q.toRotationMatrix() * _dt;
V.block<3, 3>(6, 9) = V.block<3, 3>(6, 3);
V.block<3, 3>(9, 12) = MatrixXd::Identity(3, 3) * _dt;
V.block<3, 3>(12, 15) = MatrixXd::Identity(3, 3) * _dt;
```

IMU协方差矩阵长什么样子?

```
jacobian = F * jacobian;
covariance = F * covariance * F.transpose() + V * noise * V.transpose();
```

初始值为0，表示完全信任IMU约束，因为这时候还没有来IMU数据

covariance_0

15x15 double

		ΔP			ΔQ			ΔV			ΔBa			ΔBg		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔP	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔQ	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔV	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔBa	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔBg	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMU协方差矩阵长什么样子？

```
jacobian = F * jacobian;
covariance = F * covariance * F.transpose() + V * noise * V.transpose();
```

第一帧IMU来了后，IMU约束的不确定度增加

covariance_1

15x15 double

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		ΔP			ΔQ			ΔV			ΔBa			ΔBg		
	1	5.0005e-13	5.8103e-20	2.2320e-19	-1.2685e-18	-4.4913e-15	1.1691e-15	2.0001e-10	2.3241e-17	8.9278e-17	0	0	0	0	0	0
ΔP	2	5.8103e-20	5.0005e-13	-2.6254e-20	4.4932e-15	-1.6601e-18	9.9389e-15	2.3241e-17	2.0001e-10	-1.0501e-17	0	0	0	0	0	0
	3	2.2320e-19	-2.6254e-20	5.0005e-13	-1.1725e-15	-9.9394e-15	6.3160e-20	8.9278e-17	-1.0501e-17	2.0001e-10	0	0	0	0	0	0
	4	-1.2685e-18	4.4932e-15	-1.1725e-15	2.0001e-10	0	0	-5.0738e-16	1.7972e-12	-4.6898e-13	0	0	0	0	0	0
ΔQ	5	-4.4913e-15	-1.6601e-18	-9.9394e-15	0	2.0001e-10	0	-1.7965e-12	-6.6404e-16	-3.9757e-12	0	0	0	0	0	0
	6	1.1691e-15	9.9389e-15	6.3160e-20	0	0	2.0001e-10	4.6762e-13	3.9755e-12	2.5263e-17	0	0	0	0	0	0
	7	2.0001e-10	2.3241e-17	8.9278e-17	-5.0738e-16	-1.7965e-12	4.6762e-13	8.0004e-08	9.2961e-15	3.5710e-14	0	0	0	0	0	0
ΔV	8	2.3241e-17	2.0001e-10	-1.0501e-17	1.7972e-12	-6.6404e-16	3.9755e-12	9.2961e-15	8.0004e-08	-4.2004e-15	0	0	0	0	0	0
	9	8.9278e-17	-1.0501e-17	2.0001e-10	-4.6898e-13	-3.9757e-12	2.5263e-17	3.5710e-14	-4.2004e-15	8.0004e-08	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	4.0002e-14	0	0	0	0	0
ΔBa	11	0	0	0	0	0	0	0	0	0	0	4.0002e-14	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	4.0002e-14	0	0	0
ΔBg	13	0	0	0	0	0	0	0	0	0	0	0	0	1.0001e-16	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0001e-16	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0001e-16

IMU协方差矩阵长什么样子？

```
jacobian = F * jacobian;
covariance = F * covariance * F.transpose() + V * noise * V.transpose();
```

第20帧IMU来了后，IMU约束的不确定度最大，ΔBa与ΔQ之间没有相互关系故为0，ΔBa和ΔBg对角线递增

covariance_20

15x15 double

		ΔP			ΔQ			ΔV			ΔBa			ΔBg		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔP	1	5.3302e-09	2.2246e-15	5.6751e-13	-8.5375e-14	-2.1893e-11	7.8286e-14	8.0005e-08	7.1496e-14	1.4155e-11	-1.2350e-15	2.1847e-18	1.7488e-19	3.4814e-22	2.4632e-19	-1.8045e-21
	2	2.2246e-15	5.3317e-09	-8.7698e-16	2.1794e-11	-2.1390e-13	5.7700e-11	1.8672e-13	8.0042e-08	-7.1288e-14	-2.1846e-18	-1.2350e-15	-1.3735e-18	-2.4655e-19	9.6677e-22	-6.5145e-19
	3	5.6751e-13	-8.7698e-16	5.3315e-09	-3.5156e-13	-5.7662e-11	-1.2922e-13	1.3950e-11	-2.7054e-14	8.0037e-08	-1.7189e-19	1.3730e-18	-1.2350e-15	2.9672e-21	6.5154e-19	6.1489e-22
ΔQ	4	-8.5375e-14	2.1794e-11	-3.5156e-13	4.0000e-09	5.5654e-16	5.2523e-16	-2.5105e-12	6.3749e-10	-1.8019e-11	0	0	0	-9.5000e-17	-1.1692e-19	1.1241e-19
	5	-2.1893e-11	-2.1390e-13	-5.7662e-11	5.5654e-16	4.0000e-09	-1.8046e-15	-6.4046e-10	-6.3371e-12	-1.7225e-09	0	0	0	1.1703e-19	-9.5000e-17	6.6465e-20
	6	7.8286e-14	5.7700e-11	-1.2922e-13	5.2523e-16	-1.8046e-15	4.0000e-09	9.8860e-12	1.7237e-09	-3.8729e-12	0	0	0	-1.1228e-19	-6.6670e-20	-9.5000e-17
ΔV	7	8.0005e-08	1.8672e-13	1.3950e-11	-2.5105e-12	-6.4046e-10	9.8860e-12	1.6001e-06	5.4185e-12	3.7008e-10	-3.8000e-14	9.7226e-17	-1.6493e-17	1.8076e-20	9.7541e-18	-2.2576e-19
	8	7.1496e-14	8.0042e-08	-2.7054e-14	6.3749e-10	-6.3371e-12	1.7237e-09	5.4185e-12	1.6011e-06	-2.0210e-12	-9.7187e-17	-3.8000e-14	-5.6016e-17	-9.7559e-18	4.9080e-20	-2.6549e-17
	9	1.4155e-11	-7.1288e-14	8.0037e-08	-1.8019e-11	-1.7225e-09	-3.8729e-12	3.7008e-10	-2.0210e-12	1.6010e-06	1.6657e-17	5.5936e-17	-3.8000e-14	2.8667e-19	2.6549e-17	3.1194e-20
ΔBa	10	-1.2350e-15	-2.1846e-18	-1.7189e-19	0	0	0	-3.8000e-14	-9.7187e-17	1.6657e-17	8.0000e-13	0	0	0	0	0
	11	2.1847e-18	-1.2350e-15	1.3730e-18	0	0	0	9.7226e-17	-3.8000e-14	5.5936e-17	0	8.0000e-13	0	0	0	0
	12	1.7488e-19	-1.3735e-18	-1.2350e-15	0	0	0	-1.6493e-17	-5.6016e-17	-3.8000e-14	0	0	8.0000e-13	0	0	0
ΔBg	13	3.4814e-22	-2.4655e-19	2.9672e-21	-9.5000e-17	1.1703e-19	-1.1228e-19	1.8076e-20	-9.7559e-18	2.8667e-19	0	0	0	2.0000e-15	0	0
	14	2.4632e-19	9.6677e-22	6.5154e-19	-1.1692e-19	-9.5000e-17	-6.6670e-20	9.7541e-18	4.9080e-20	2.6549e-17	0	0	0	0	2.0000e-15	0
	15	-1.8045e-21	-6.5145e-19	6.1489e-22	1.1241e-19	6.6465e-20	-9.5000e-17	-2.2576e-19	-2.6549e-17	3.1194e-20	0	0	0	0	0	2.0000e-15

Jacobian长什么样子?

```
jacobian = F * jacobian;
covariance = F * covariance * F.transpose() + V * noise * V.transpose();
```

初始值为单位阵

jacobian_0

15x15 double

		P			Q			V			Ba			Bg		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔP	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
ΔQ	4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ΔV	7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
ΔBa	10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
ΔBg	13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Jacobian长什么样子？

```
jacobian = F * jacobian;
covariance = F * covariance * F.transpose() + V * noise * V.transpose();
```

第20帧IMU来了后，主要用PQV的误差关于Ba、Bg的Jacobian

jacobian_20

15x15 double

			P			Q			V			Ba			Bg	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	1	0	0	3.1455e-11	-0.0166	1.7279e-06	0.1000	0	0	-0.0050	5.7737e-06	1.6661e-06	5.2003e-07	5.4734e-04	-3.6422e-06
ΔP	2	0	1	0	0.0166	1.4219e-09	0.0429	0	0.1000	0	-5.7741e-06	-0.0050	-3.2771e-06	-5.4798e-04	1.2361e-06	-0.0014
	3	0	0	1	-1.7284e-06	-0.0429	-1.4572e-09	0	0	0.1000	-1.6601e-06	3.2775e-06	-0.0050	5.2931e-06	0.0014	7.0945e-07
	4	0	0	0	1.0000	0.0039	-0.0017	0	0	0	0	0	0	-0.1000	-1.9556e-04	1.6205e-04
ΔQ	5	0	0	0	-0.0039	1.0000	-0.0022	0	0	0	0	0	0	1.9578e-04	-0.1000	1.1213e-04
	6	0	0	0	0.0017	0.0022	1.0000	0	0	0	0	0	0	-1.6177e-04	-1.1250e-04	-0.1000
	7	0	0	0	-2.6081e-09	-0.3264	0.0030	1	0	0	-0.1000	1.8386e-04	-4.0377e-06	1.9658e-05	0.0160	-2.9366e-04
ΔV	8	0	0	0	0.3264	-2.1411e-08	0.8593	0	1	0	-1.8382e-04	-0.1000	-1.0543e-04	-0.0160	5.0152e-05	-0.0431
	9	0	0	0	-0.0030	-0.8593	2.3887e-08	0	0	1	4.3095e-06	1.0533e-04	-0.1000	3.5842e-04	0.0431	3.0417e-05
	10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
ΔBa	11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
ΔBg	14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

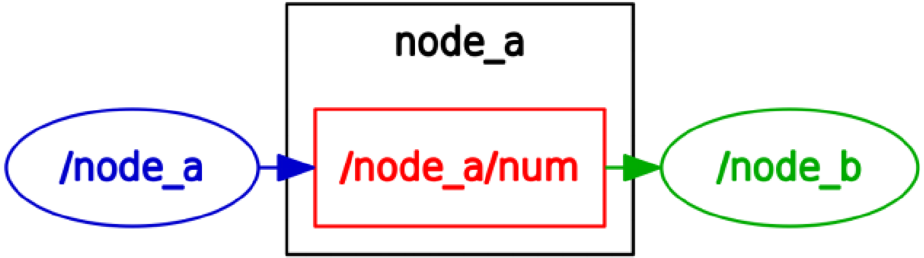
作业

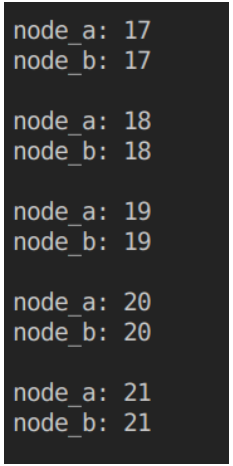
一、练习ROS基本功（1h）

新建一个ROS工程，有两个Package：node_a和node_b，如图1。其中，node_a负责每隔一定时间发送一个数字，node_b负责接收数字并显示，通过roslaunch运行这两个node。

要求：手写package.xml、CMakeLists.txt、run.launch

提交形式：ROS工程、运行结果截图





结果示例

作业

二、画vins_mono中的feature_tracker和vins_estimator流程图（可用dia、starUML等软件）(1h)

要求：
画关键流程
尽量简洁、明了，主要为了缕清数据走向。

提交形式：图片

作业

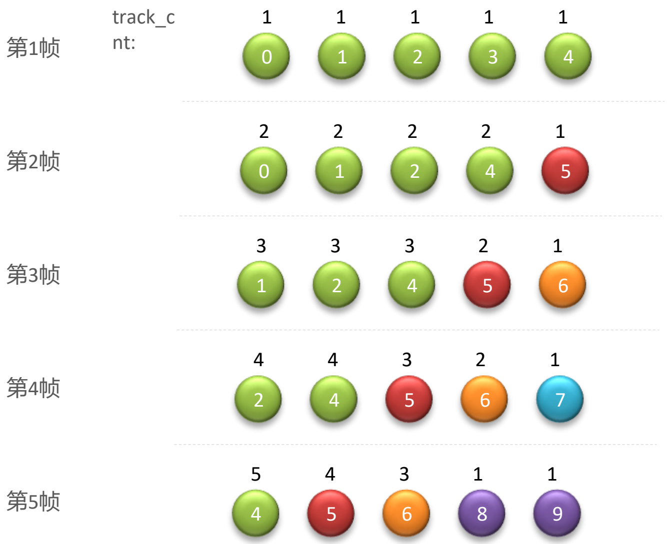
三、运行MH05，打印相邻两帧图像的特征点跟踪情况 (0.5 h)

具体包括：

1. 上一帧所有特征点的id及其被跟踪的次数
2. 当前帧所有特征点的id及其被跟踪的次数

分析特征点的跟踪情况是否符合你预期？

提交形式：图片



作业

四、打印相邻两帧图像间的IMU预积分信息 (0.5 h)

具体包括：
1. 协方差
2. Jacobian

为了对协方差、Jacobian有个直观的印象。

提交形式：图片

ΔP ΔQ ΔV ΔBa ΔBg

covariance_20

15x15 double

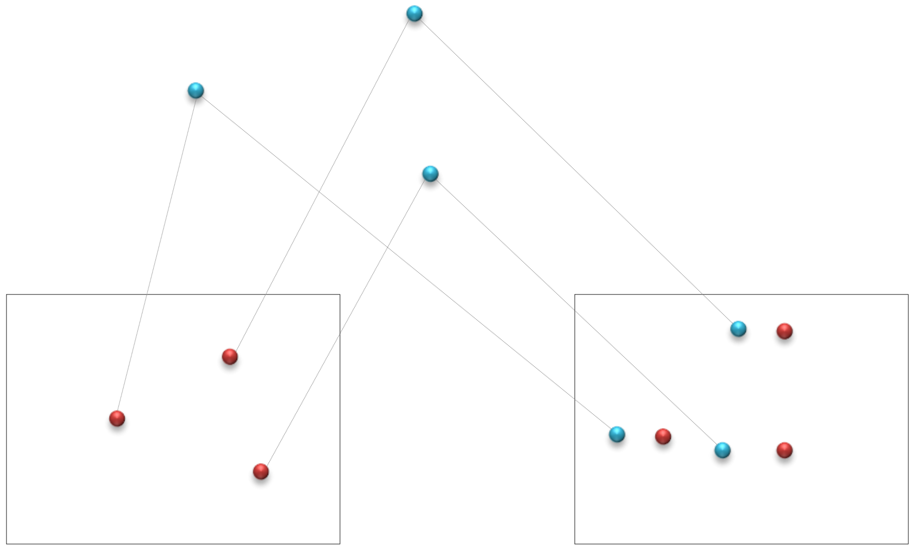
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	5.3302e-09	2.2246e-15	5.6751e-13	-8.5375e-14	-2.1893e-11	7.8286e-14	8.0005e-08	7.1496e-14	1.4155e-11	-1.2350e-15	2.1847e-18	1.7488e-19	3.4814e-22	2.4632e-19	-1.8045e-21
ΔP	2	2.2246e-15	5.3317e-09	-8.7698e-16	2.1794e-11	-2.1390e-13	5.7700e-11	1.8672e-13	8.0042e-08	-7.1288e-14	-2.1846e-18	-1.2350e-15	-1.3735e-18	-2.4655e-19	9.6677e-22	-6.5145e-19
	3	5.6751e-13	-8.7698e-16	5.3315e-09	-3.5156e-13	-5.7662e-11	-1.2922e-13	1.3950e-11	-2.7054e-14	8.0037e-08	-1.7189e-19	1.3730e-18	-1.2350e-15	2.9672e-21	6.5154e-19	6.1489e-22
	4	-8.5375e-14	2.1794e-11	-3.5156e-13	4.0000e-09	5.5654e-16	5.2523e-16	-2.5105e-12	6.3749e-10	-1.8019e-11	0	0	0	-9.5000e-17	-1.1692e-19	1.1241e-19
ΔQ	5	-2.1893e-11	-2.1390e-13	-5.7662e-11	5.5654e-16	4.0000e-09	-1.8046e-15	-6.4046e-10	-6.3371e-12	-1.7225e-09	0	0	0	1.1703e-19	-9.5000e-17	6.6465e-20
	6	7.8286e-14	5.7700e-11	-1.2922e-13	5.2523e-16	-1.8046e-15	4.0000e-09	9.8860e-12	1.7237e-09	-3.8729e-12	0	0	0	-1.1228e-19	-6.6670e-20	-9.5000e-17
	7	8.0005e-08	1.8672e-13	1.3950e-11	-2.5105e-12	-6.4046e-10	9.8860e-12	1.6001e-06	5.4185e-12	3.7008e-10	-3.8000e-14	9.7226e-17	-1.6493e-17	1.8076e-20	9.7541e-18	-2.2576e-19
ΔV	8	7.1496e-14	8.0042e-08	-2.7054e-14	6.3749e-10	-6.3371e-12	1.7237e-09	5.4185e-12	1.6011e-06	-2.0210e-12	-9.7187e-17	-3.8000e-14	-5.6016e-17	-9.7559e-18	4.9080e-20	-2.6549e-17
	9	1.4155e-11	-7.1288e-14	8.0037e-08	-1.8019e-11	-1.7225e-09	-3.8729e-12	3.7008e-10	-2.0210e-12	1.6010e-06	1.6657e-17	5.5936e-17	-3.8000e-14	2.8667e-19	2.6549e-17	3.1194e-20
	10	-1.2350e-15	-2.1846e-18	-1.7189e-19	0	0	0	-3.8000e-14	-9.7187e-17	1.6657e-17	8.0000e-13	0	0	0	0	0
ΔBa	11	2.1847e-18	-1.2350e-15	1.3730e-18	0	0	0	9.7226e-17	-3.8000e-14	5.5936e-17	0	8.0000e-13	0	0	0	0
	12	1.7488e-19	-1.3735e-18	-1.2350e-15	0	0	0	-1.6493e-17	-5.6016e-17	-3.8000e-14	0	0	8.0000e-13	0	0	0
ΔBg	13	3.4814e-22	-2.4655e-19	2.9672e-21	-9.5000e-17	1.1703e-19	-1.1228e-19	1.8076e-20	-9.7559e-18	2.8667e-19	0	0	0	2.0000e-15	0	0
	14	2.4632e-19	9.6677e-22	6.5154e-19	-1.1692e-19	-9.5000e-17	-6.6670e-20	9.7541e-18	4.9080e-20	2.6549e-17	0	0	0	0	2.0000e-15	0
	15	-1.8045e-21	-6.5145e-19	6.1489e-22	1.1241e-19	6.6465e-20	-9.5000e-17	-2.2576e-19	-2.6549e-17	3.1194e-20	0	0	0	0	0	2.0000e-15

作业

五、根据IMU积分出的当前帧位姿作为初始值，将上一帧的路标点投影到当前帧的像素坐标系下，看看与LK跟踪的像素位置偏差大小 (0.5 h)

为了理解IMU与图像的相互关系

提交形式：图片



感谢各位聆听!

Thanks for Listening