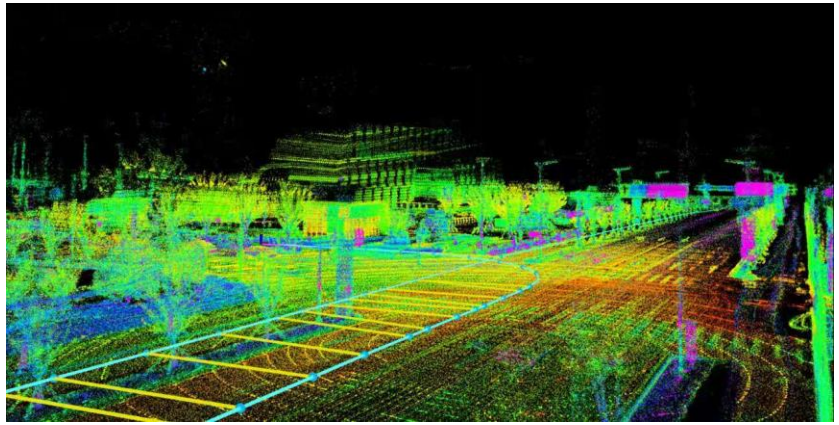
微信扫码加入星球

自动驾驶中实战课之相机与IMU的同步实战

Camera + LiDAR + Radar + IMU



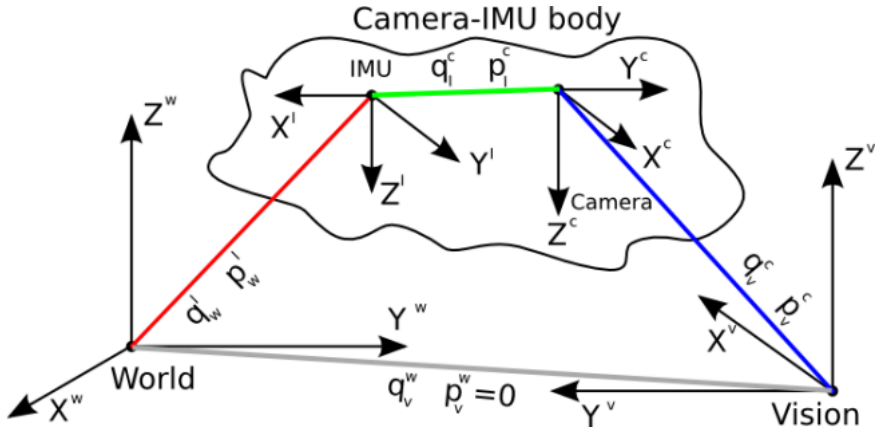
主　讲　人：帅的丑小鸭

公　众　号：3D 视 觉 工 坊

内容

一、相机与IMU为什么要同步？

- IMU所在的位置为车体坐标系的原点(不绝对)
- 相机与IMU具有互补性
  - ➢相机在高速运动、光照改变情况下容易失效，但能够获得丰富的环境信息；
  - ➢IMU能够获得车辆的运动信息，并且不受周围环境的影响，从而弥补相机的不足



一、相机与IMU为什么要同步？

- 相机与IMU之间的相对位姿，即求解相机坐标系与IMU坐标系之间的变换，包括相对旋转角和相对平移量：

$$T_{wb} = T_{wc} \cdot T_{cb}$$

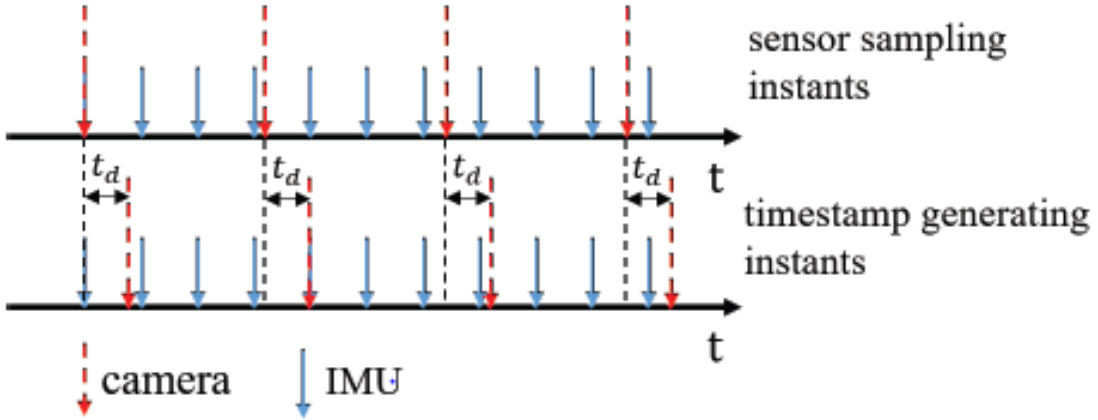
$$\begin{bmatrix} R_{wb} & t_{wb} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{wc} & t_{wc} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_{cb} & t_{cb} \\ 0 & 1 \end{bmatrix}$$

其中wc表示相机坐标系，wb表示IMU坐标系，cb表示相对坐标系。

- 将上式展开可以得到相机坐标系与IMU坐标系之间旋转角和平移量的变换关系：

$$R_{wb} = R_{wc} \cdot R_{cb}$$
$$t_{wb} = R_{wc} \cdot t_{cb} + t_{wc}$$

- 由于触发、传输等延时的存在，相机与IMU采样的时间和时间戳的时间并不匹配，如下图，这会导致相机与IMU之间存在时间差。



那么这里时间差的计算为：

$$t_{IMU} = t_{cam} + t_d$$

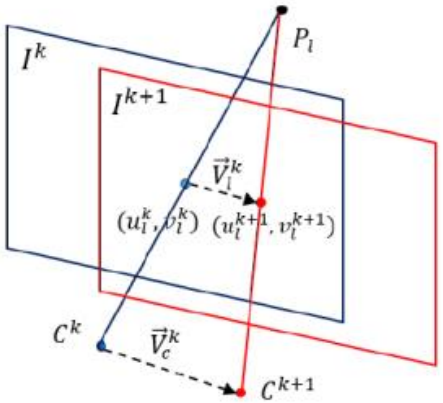
即将相机的时间戳平移td后，相机与IMU之间实现了同步

空间同步的原理如下：

假设在较短时间内相机从C^k匀速运动到C^{k+1}，

特征点在像素平面上的投影也从$[u_l^k, v_l^k]$匀速运动至$[u_l^{k+1}, v_l^{k+1}]$，

像素匀速运动的速度为V_l^k



通过平移像素坐标，将时间差作为优化变量加入投影误差的表达式中，如下式：

$$\mathbf{e}_l^k = \mathbf{z}_l^k(t_d) - \pi(\mathbf{R}_{c_k}^{w^T}(\mathbf{P}_l - \mathbf{p}_{c_k}^w))$$
$$\mathbf{z}_l^k(t_d) = [u_l^k \ v_l^k]^T + t_d \mathbf{V}_l^k.$$

此投影误差项与IMU误差项和初始值误差项相加进行联合非线性优化。每一次优化结束后对时间差进行补偿，最终时间差的标定量逐渐趋于0

环境和代码配置说明

1.配置内容均是在docker 下进行，在docker下/home/下：

```
root@6dc69d3505b5:/home# ls
3rdparty  workspace
root@6dc69d3505b5:/home#
```

```
root@6dc69d3505b5:/home/workspace# ls
ad_sensor_fusion  data
root@6dc69d3505b5:/home/workspace#
```

2. 下载网盘数据和3rdparty配置如下：（如果之前配置过，可以忽略该步骤）

文件夹3rdparty, 这个opencv版本是为了特征点提取专门编译的，同时与其同级放置的还有workspace工作空间，

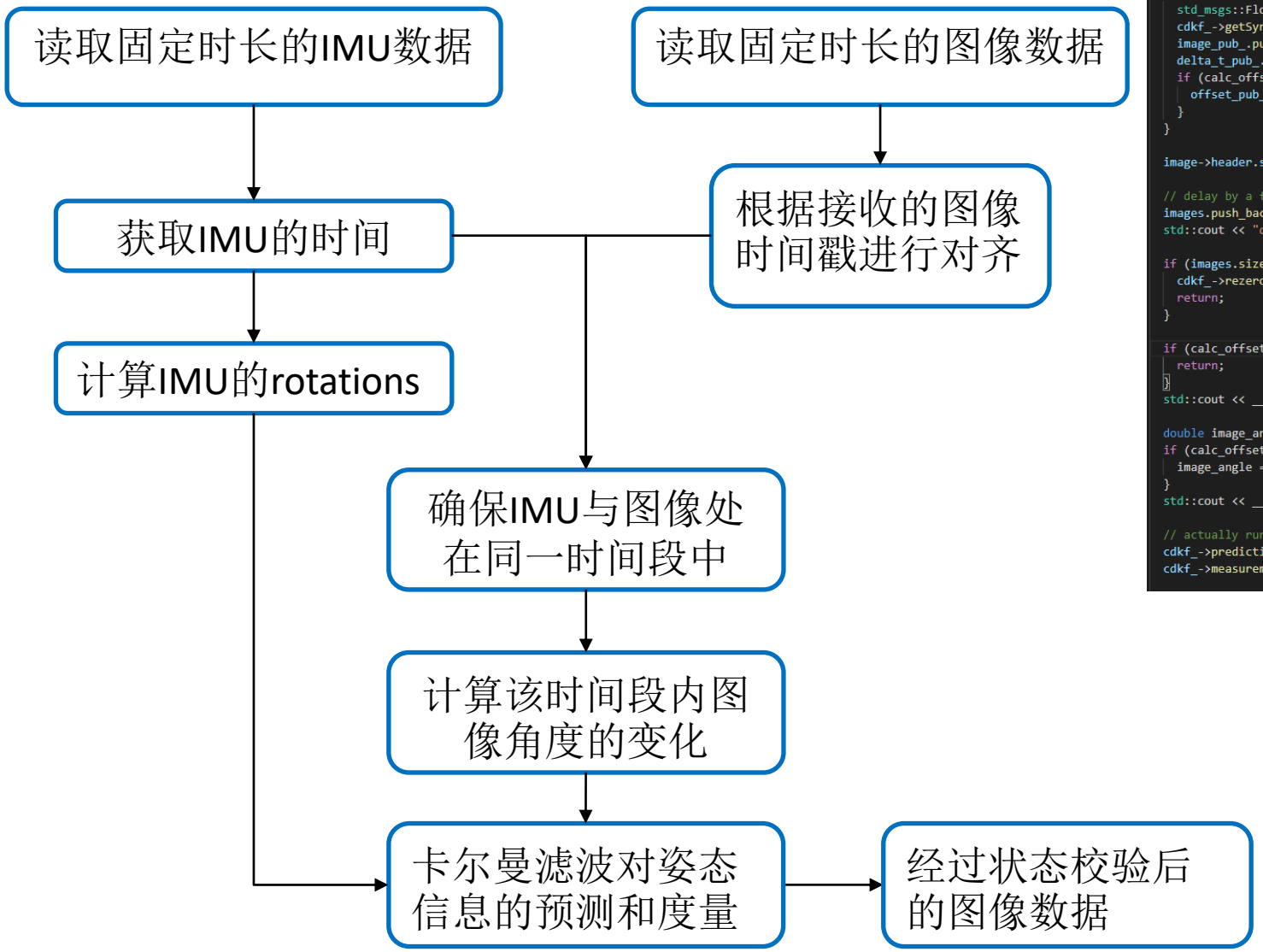
在workspace下放置ad_sensor_fusion和data，后面我们的数据统一放置在data中。

```
path:        0803_0.bag
version:     2.0
duration:    10:00s (600s)
start:       Aug 03 2021 16:45:05.47 (1627980305.47)
end:         Aug 03 2021 16:55:05.52 (1627980905.52)
size:        82.4 GB
messages:    256926
compression: none [18081/18081 chunks]
types:       geometry_msgs/TwistStamped [98d34b0043a2093cf9d9345ab6eef12e]
             nav_msgs/Odometry          [cd5e73d190d741a2f92e81eda573aca7]
             sensor_msgs/Image          [060021388200f6f0f447d0fcd9c64743]
             sensor_msgs/Imu            [6a62c6daae103f4ff57a132d6f95cec2]
             sensor_msgs/NavSatFix      [2d3a8cd499b9b4a0249fb98fd05cfa48]
             sensor_msgs/PointCloud2    [1158d486dd51d683ce2f1be655c3c181]
topics:      /cgi610/imu           59736 msgs    : sensor_msgs/Imu
             /cgi610/nav_fix       59736 msgs    : sensor_msgs/NavSatFix
             /cgi610/twist         59736 msgs    : geometry_msgs/TwistStamped
             /localization/odom    59742 msgs    : nav_msgs/Odometry
             /raw_image            11982 msgs    : sensor_msgs/Image
             /velodyne_points       5994 msgs    : sensor_msgs/PointCloud2
```

3. 重新pull代码，更新到最新版本，查看log，确认更新了camera_imu_sync

```
update camera imu time
```

整体架构




```
// fire the image back out with minimal lag
if (images.size() >= (delay_by_n_frames_ - 1)) {
  std_msgs::Float64 delta_t, offset;
  cdkf_->getSyncedTimestamp(stamp, &(image->header.stamp), &(delta_t.data), &(offset.data));
  image_pub_.publish(image->toImageMsg());
  delta_t_pub_.publish(delta_t);
  if (calc_offset_) {
    offset_pub_.publish(offset);
  }
}

image->header.stamp = stamp;

// delay by a few messages to ensure IMU messages span needed range
images.push_back(*image);
std::cout << "delay by a few messages "<< images.size() << std::endl;

if (images.size() < delay_by_n_frames_) {
  cdkf_->rezeroTimestamps(images.front().header.stamp, true);
  return;
}

if (calc_offset_ && (imu_rotations_.size() < 2)) {
  return;
}
std::cout << __LINE__ << " " << __FILE__ << " " << imu_rotations_.size() << std::endl;

double image_angle = 0.0;
if (calc_offset_) {
  image_angle = calcAngleBetweenImages(images.begin()->image, std::next(images.begin())->image, focal_length_);
}
std::cout << __LINE__ << " " << __FILE__ << " " << imu_rotations_.size() << std::endl;

// actually run filter
cdkf_->predictionUpdate(std::next(images.begin())->header.stamp);
cdkf_->measurementUpdate(images.begin()->header.stamp, std::next(images.begin())->header.stamp, image_angle, imu_rotations_, calc_offset_);
```

代码解释—计算IMU运动状态

```
// integrate imu reading
double half_delta_t = (msg->header.stamp - prev_msg.header.stamp).toSec() / 2.0;

Eigen::Quaterniond delta_angle =
    Eigen::AngleAxisd(half_delta_t * (msg->angular_velocity.x + prev_msg.angular_velocity.x),
                      Eigen::Vector3d::UnitX()) *
    Eigen::AngleAxisd(half_delta_t * (msg->angular_velocity.y + prev_msg.angular_velocity.y),
                      Eigen::Vector3d::UnitY()) *
    Eigen::AngleAxisd(half_delta_t * (msg->angular_velocity.z + prev_msg.angular_velocity.z),
                      Eigen::Vector3d::UnitZ());

imu_rotations_.emplace_back(
    prev_msg.header.stamp + ros::Duration(half_delta_t),
    imu_rotations_.back().second * delta_angle);
imu_rotations_.back().second.normalize();
```

代码解释—计算图像角度变化

```cpp
double calcAngleBetweenImages(const cv::Mat& prev_image,
                              const cv::Mat& image, float focal_length) {
  constexpr int kMaxCorners = 100;
  constexpr double kQualityLevel = 0.01;
  constexpr double kMinDistance = 10;

  std::vector<cv::Point2f> prev_points;
  cv::goodFeaturesToTrack(prev_image, prev_points, kMaxCorners, kQualityLevel, kMinDistance);

  if (prev_points.size() == 0) {
    ROS_ERROR("Tracking has failed cannot calculate angle");
    return 0.0;
  }

  std::vector<cv::Point2f> points;
  std::vector<uint8_t> valid;
  std::vector<float> err;
  cv::calcOpticalFlowPyrLK(prev_image, image, prev_points, points, valid, err);

  std::vector<cv::Point2f> tracked_prev_points, tracked_points;
  for (size_t i = 0; i < prev_points.size(); ++i) {
    if (valid[i]) {
      tracked_prev_points.push_back(prev_points[i]);
      tracked_points.push_back(points[i]);
    }
  }
```

```cpp
  // close enough for most cameras given the low level of accuracy needed
  const cv::Point2f offset(image.cols / 2.0, image.rows / 2.0);

  for (size_t i = 0; i < tracked_points.size(); ++i) {
    tracked_prev_points[i] = (tracked_prev_points[i] - offset) / focal_length;
    tracked_points[i] = (tracked_points[i] - offset) / focal_length;
    //std::cout << tracked_prev_points[i].x << " " << tracked_prev_points[i].y << std::endl;
  }

  constexpr double kMaxEpipoleDistance = 1e-3;
  constexpr double kInlierProbability = 0.99;

  std::vector<uint8_t> inliers;
  cv::Mat cv_F = cv::findFundamentalMat(tracked_prev_points, tracked_points, cv::FM_LMEDS,
                                        kMaxEpipoleDistance, kInlierProbability, inliers);
  Eigen::Matrix3d E, W;

  cv::cv2eigen(cv_F, E);
  Eigen::JacobiSVD<Eigen::MatrixXd> svd(E, Eigen::ComputeThinU | Eigen::ComputeThinV);
  W << 0.0, -1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0;

  Eigen::Matrix3d Ra = svd.matrixU() * W * svd.matrixV().transpose();
  Eigen::Matrix3d Rb = svd.matrixU() * W.transpose() * svd.matrixV().transpose();

  double angle = std::min(Eigen::AngleAxisd(Ra).angle(), Eigen::AngleAxisd(Rb).angle());

  return angle;
```

代码解释—卡尔曼滤波简介

```
void CDKF::predictionUpdate(const ros::Time& received_timestamp) {
```

```
void CDKF::measurementUpdate(const ros::Time& prev_stamp,
                             const ros::Time& current_stamp,
                             const double image_angular_velocity,
                             const IMUList& imu_rotations,
                             const bool calc_offset) {
```

```
void CDKF::stateToMeasurementEstimate(
    const IMUList& imu_rotations, const ros::Time zero_stamp, bool calc_offset,
    const Eigen::VectorXd& input_state, const Eigen::VectorXd& noise,
    Eigen::Ref<Eigen::VectorXd> estimated_measurement) {
```

```
void CDKF::propergateState(const Eigen::VectorXd& noise,
                           Eigen::Ref<Eigen::VectorXd> current_state) {
```

本次课课后作业

1）尝试使用自己的数据集来运行程序

2）尝试优化图像角度计算代码

3）将代码中的结果显式出来

购买该课程请扫描二维码

微信扫码加入星球

感谢聆听

Thanks for Listening