



## 多传感器融合第七章作业讲评



主讲人 王志勇



在提供的工程框架中,补全代码,实现基于地图的融合定位,并与不加滤波时的定位结果做比较。

(注:由于kitti数据有部分问题,虽然课程提供了修复后的数据,但是仍会对结果有影响,因此在某些路段滤波效果会较差,可以忽略这部分路段。)

# 1) 及格

- 1) 及格: 补全代码, 且滤波功能正常。

lidar\_localization/src/models/kalman\_filter/error\_state\_kalman\_filter

```
602 602 //
603 603 // TODO: perform Kalman prediction
604 604 //
605 605 - X_ = F * X_; // fix this
606 606 - P_ = F * P_ * F.transpose() + B * Q_ * B.transpose(); // fix this
605 605 + X_ = X_; // fix this
606 606 + P_ = P_; // fix this
607 607 }
608 608
609 609 /**
```

# 1) 及格

```
617 617 //
618 618 // TODO: set measurement:
619 619 //
620 - Eigen::Vector3d P_nn_obs =
621 -     T_nb.block<3, 1>(0, 3) - pose_.block<3, 1>(0, 3); // fix this
622 - Eigen::Vector3d R_nn_obs = Sophus::SO3d::vee(
623 -     pose_.block<3, 3>(0, 0).transpose() * T_nb.block<3, 3>(0, 0) -
624 -     Eigen::Matrix3d::Identity()); // fix this
620 + Eigen::Vector3d P_nn_obs = Eigen::Vector3d::Zero(); // fix this
621 + Eigen::Vector3d R_nn_obs = Eigen::Matrix3d::Identity(); // fix this
625 622
626 623 YPose_.block<3, 1>(0, 0) = P_nn_obs;
627 624 YPose_.block<3, 1>(3, 0) = R_nn_obs;
```

# 1) 及格

```
635 632    // TODO: set Kalman gain:
636 633    //
637 634    MatrixRPose R = RPose_; // fix this
638    - // K = K;                // fix this
639    - K = P_ * G.transpose() *
640    -   (G * P_ * G.transpose() + CPose_ * R * CPose_.transpose()).inverse();
635    + K = K;                // fix this
636    +
641 637 }
642 638
643 639 /**
```

# 1) 及格

```
665 661 // TODO: perform Kalman correct:
666 662 //
667 663
668 - P_ = (MatrixP::Identity() - K * G) * P_; // fix this
669 - X_ = X_ + K * (Y - G * X_); // fix this
664 + P_ = P_; // fix this
665 + X_ = X_; // fix this
670 666 }
671 667
672 668 /**
```

# 1) 及格

```
680 676 //
681 677 // a. position:
682 678 pose_.block<3, 1>(0, 3) =
683 - pose_.block<3, 1>(0, 3) + X_.block<3, 1>(INDEX_ERROR_POS, 0); // fix this
679 + pose_.block<3, 1>(0, 3); // fix this
684 680 // b. velocity:
685 - vel_ = vel_ + X_.block<3, 1>(INDEX_ERROR_VEL, 0); // fix this
681 + vel_ = vel_; // fix this
686 682 // c. orientation:
687 683 Eigen::Matrix3d C_nn =
688 684 Sophus::SO3d::exp(X_.block<3, 1>(INDEX_ERROR_ORI, 0)).matrix();
689 - pose_.block<3, 3>(0, 0) = pose_.block<3, 3>(0, 0) * C_nn; // fix this
685 + pose_.block<3, 3>(0, 0) = pose_.block<3, 3>(0, 0); // fix this
690 686
691 687 // d. gyro bias:
692 688 if (IsCovStable(INDEX_ERROR_GYRO)) {
```

## 2)良好

2)良好:补全代码,功能正常,且经过调试参数,滤波后性能比滤波前好。(这个没有定量标准,各位把详细的误差对比结果提供在作业中,供助教有足够依据评阅)

```
error_state_kalman_filter:
earth:
  # gravity can be calculated from https://www.sensorsone.com/local-gravity-calculator/ using latitude and height:
  gravity_magnitude: 9.80943
  # rotation speed, rad/s:
  rotation_speed: 7.292115e-5
  # latitude:
  latitude: 48.9827703173
covariance:
prior:
  pos: 1.0e-6
  vel: 1.0e-6
  ori: 1.0e-6
  epsilon: 1.0e-6
  delta: 1.0e-6
process:
  gyro: 1.0e-6
  accel: 1.0e-4 #2.5e-3
  bias_accel: 1.0e-4 #2.5e-3
  bias_gyro: 1.0e-6
measurement:
  pose:
    pos: 1.0e-3
    ori: 1.0e-3
  pos: 1.0e-3
  vel: 1.0e-3
motion_constraint:
  activated: true
  w_b_thresh: 0.13
```



### 3)优秀

**3)优秀：**在前面的模型推导中，考虑了器件误差中的随机游走，请给出不考虑随机游走模型(即  $\delta \dot{\mathbf{b}}_a = 0$  ,  $\delta \dot{\mathbf{b}}_\omega = 0$  )时的推导过程，并在工程框架中实现。对比这两种方法的性能差异(最好给出原因分析)。另外，kalman滤波的性能对噪声的设置较为敏感，请在提供结果的同时，给出不同噪声设置情况下的结果对比(至少5组参数)。

$$\boldsymbol{\omega} = \begin{bmatrix} \mathbf{n}_a \\ \mathbf{n}_\omega \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{B}_t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \mathbf{R}_t & 0 & 0 & 0 \\ 0 & \mathbf{I}_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B}_{k-1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \mathbf{R}_{k-1}T & 0 & 0 & 0 \\ 0 & \mathbf{I}_3T & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# 参考

- 
- south\_west 的作业



深蓝学院  
shenlanxueyuan.com

感谢各位聆听 !  
Thanks for Listening

