CHENXXX 第七章作业

完成情况

本次作业我完成了及格、良好和优秀的要求。同时也基于参考论文，推导并复现了基于名义状态的EKF。

及格要求

补充`UpdateOdomEstimation`

```
// get deltas:
Eigen::Vector3d angluar_delta;
GetAngularDelta(1, 0, angluar_delta, angular_vel_mid);

// update orientation:
Eigen::Matrix3d R_curr, R_prev;
UpdateOrientation(angluar_delta, R_curr, R_prev);

// get velocity delta:
double T = 0.0;
Eigen::Vector3d velocity_delta;
GetVelocityDelta(1, 0, R_curr, R_prev, T, velocity_delta, linear_acc_mid);

// update position:
UpdatePosition(T, velocity_delta);
```

补充`SetProcessEquation`

```
// a. set process equation for delta vel:
F_.block<3,3>(kIndexErrorVel, kIndexErrorOri) = -C_nb * Sophus::SO3d::hat(f_n).matrix();
F_.block<3,3>(kIndexErrorVel, kIndexErrorAccel) = -C_nb;
// b. set process equation for delta ori:
F_.block<3,3>(kIndexErrorOri, kIndexErrorOri) = -Sophus::SO3d::hat(w_b).matrix();
B_.block<3,3>(kIndexErrorVel, kIndexNoiseAccel) = C_nb;
```

补充`UpdateErrorEstimation`

```
UpdateProcessEquation(linear_acc_mid, angular_vel_mid);

MatrixF F = MatrixF::Identity() + F_*T;
MatrixB B = B_*T;
if(correct_bias_){
   B.block<6,6>(kIndexErrorAccel, kIndexNoiseBiasAccel) = Eigen::Matrix<double, 6, 6>::Identity() * sqrt(T);
}

P_ = F*P_*F.transpose()+B*Q_*B.transpose();
```

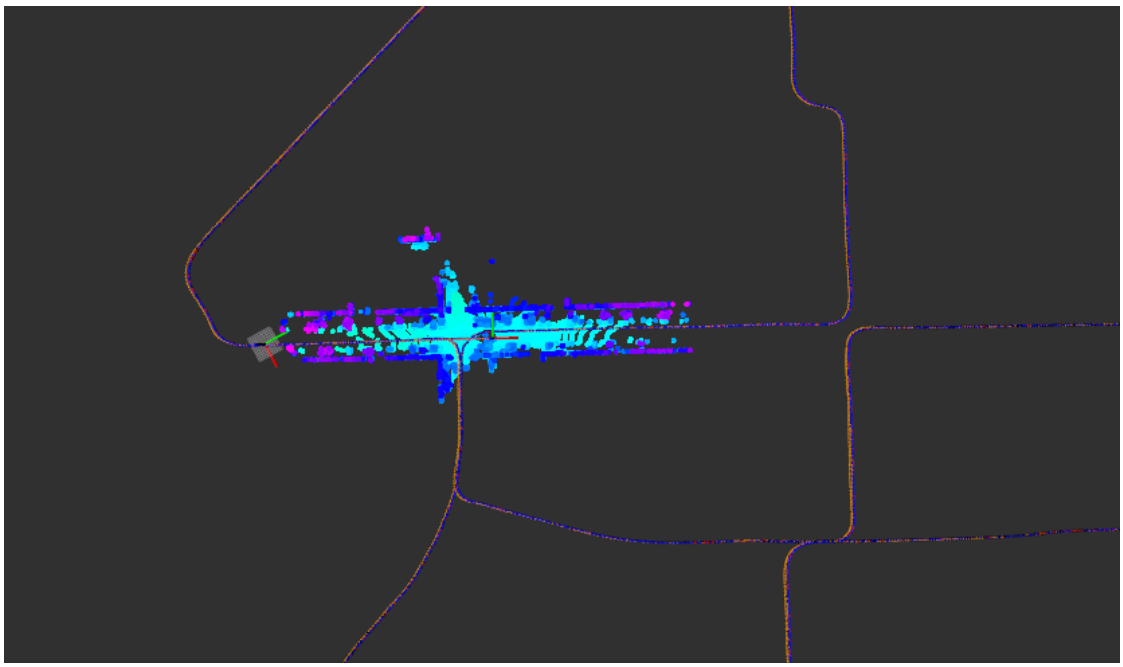
补充`CorrectErrorEstimationPose`

```
Eigen::Vector3d delta_p = pose_.block<3,1>(0,3) - T_nb.block<3,1>(0,3);
Eigen::Matrix3d delta_R = T_nb.block<3,3>(0,0).transpose() * pose_.block<3,3>(0,0);
Eigen::Vector3d delta_theta = Sophus::SO3d::vee(delta_R-Eigen::Matrix3d::Identity());

YPose_.block<3,1>(0,0) = delta_p;
YPose_.block<3,1>(3,0) = delta_theta;
Y = YPose_;
G = GPose_;

// 这里没必要C*R*C^T 因为C是一个单位阵
K = P_*G.transpose()*(G*P_*G.transpose() + RPose_).inverse();
```

补充`EliminateError`

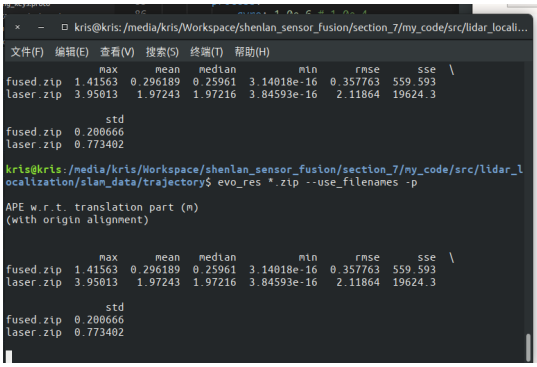
```
// a. position:
pose_.block<3,1>(0,3) = pose_.block<3,1>(0,3) - X_.block<3,1>(kIndexErrorPos,0);
// b. velocity:
vel_ = vel_ - X_.block<3,1>(kIndexErrorVel,0);
// c. orientation:
Eigen::Matrix3d delta_R = Eigen::Matrix3d::Identity() - Sophus::SO3d::hat(X_.block<3,1>
(kIndexErrorOri,0)).matrix();
Eigen::Quaterniond dq = Eigen::Quaterniond(delta_R);
dq = dq.normalized();
pose_.block<3,3>(0,0) = pose_.block<3,3>(0,0) * dq.toRotationMatrix();

gyro_bias_ -= X_.block<3, 1>(kIndexErrorGyro, 0);

// e. accel bias:
accl_bias_ -= X_.block<3, 1>(kIndexErrorAccel, 0);
```

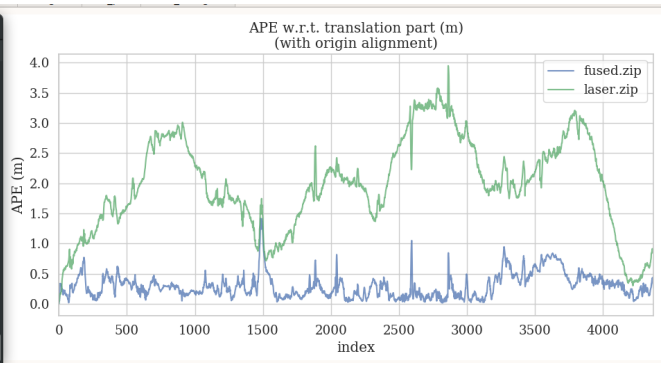
补全代码，程序正常运行，如下图：



良好要求

融合里程计的rmse为0.26，激光里程计的rmse为2.119，同时从ape曲线能看出，滤波后性能比滤波前好





优秀要求

分析Q和R对滤波器的影响

我的个人调试卡尔曼滤波器的方法是按Q/R的大小来调：

- 当Q/R越大，表示Q越大，预测的噪声越大，**系统更相信观测**
- 当Q/R越小，表示R越大，观测的噪声越大，**系统更相信预测**

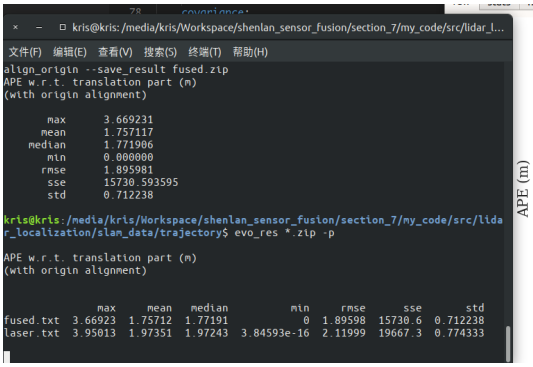
为了使结果更明显，设计如下实验：

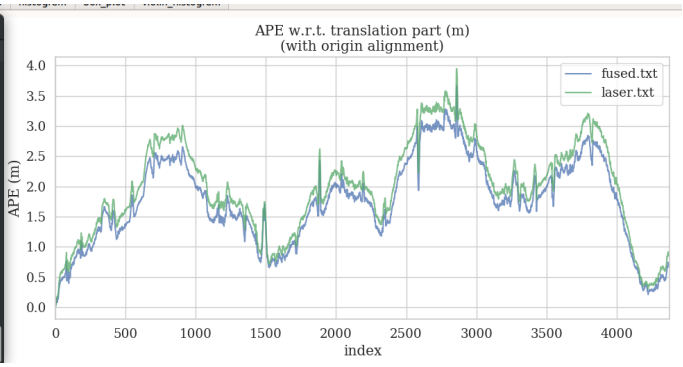
- 当Q减少，R不变时，更相信预测值
- 当Q增大，R减小时，更相信观测值
- 当R减少，Q不变时，更相信观测值
- 当R增大，Q不变时，更相信预测值

（1）减小Q，R增大

```
process:
gyro: 1.0e-2 # 1.0e-4
accel: 2.5e-1 # 2.5e-3
bias_accel: 2.5e-1 # 2.5e-3
bias_gyro: 1.0e-2 # 1.0e-4
measurement:
pose:
pos: 1.0e-6 # 1.0e-4
ori: 1.0e-6 # 1.0e-4
pos: 1.0e-6 # 1.0e-4
vel: 2.5e-5 # 2.5e-3
```

系统更相信观测值，融合ape曲线与激光ape曲线越相似

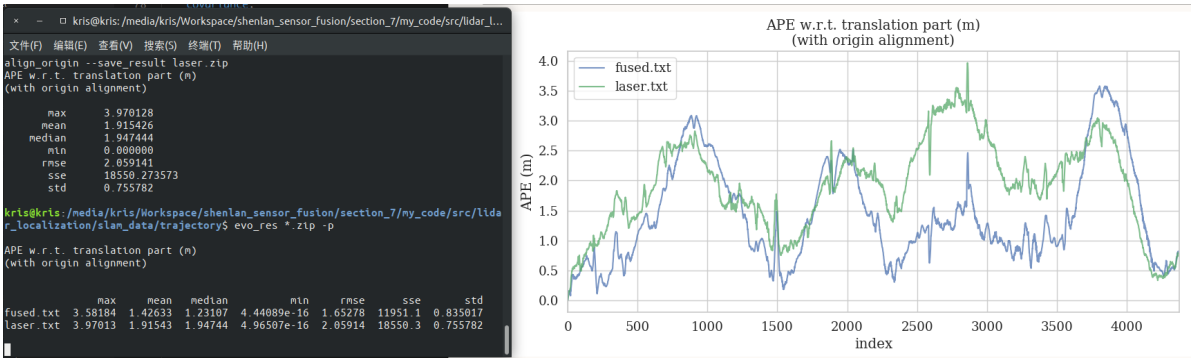




（2）增大Q，R减小

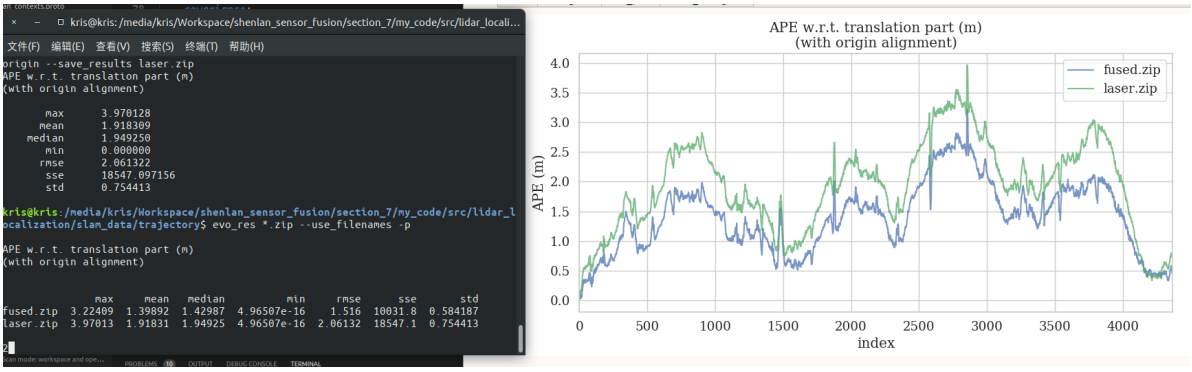
```
process:
gyro: 1.0e-6 # 1.0e-4
accel: 2.5e-5 # 2.5e-3
bias_accel: 2.5e-5 # 2.5e-3
bias_gyro: 1.0e-6 # 1.0e-4
measurement:
pose:
pos: 1.0e-2 # 1.0e-4
ori: 1.0e-2 # 1.0e-4
pos: 1.0e-2 # 1.0e-4
vel: 2.5e-1 # 2.5e-3
```

系统更相信预测值，由于预测有误差，系统精度会下降，同时融合ape曲线与激光ape曲线不相似



（3）减小R，Q不变

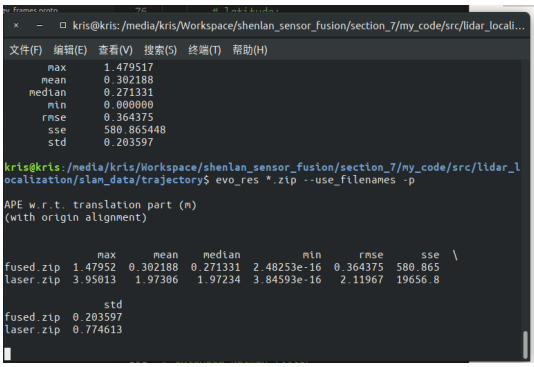
```
process:
gyro: 1.0e-4 # 1.0e-4
accel: 2.5e-3 # 2.5e-3
bias_accel: 2.5e-3 # 2.5e-3
bias_gyro: 1.0e-4 # 1.0e-4
measurement:
pos: 1.0e-6 # 1.0e-4
ori: 1.0e-6 # 1.0e-4
pos: 1.0e-6 # 1.0e-4
vel: 2.5e-5 # 2.5e-3
```

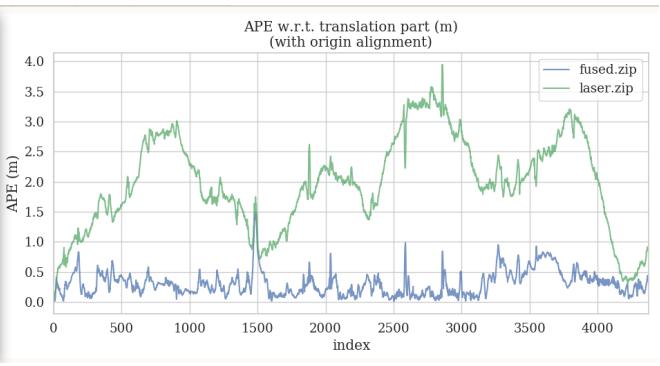


系统更相信观测值，同时融合ape曲线与激光ape曲线相似

（4）增大R，Q不变

```
processprocess:
gyro: 1.0e-4 # 1.0e-4
accel: 2.5e-3 # 2.5e-3
bias_accel: 2.5e-3 # 2.5e-3
bias_gyro: 1.0e-4 # 1.0e-4
measurement:
pos: 1.0e-2 # 1.0e-4
ori: 1.0e-2 # 1.0e-4
pos: 1.0e-2 # 1.0e-4
vel: 2.5e-1 # 2.5e-3
```





系统更相信预测值，这是一组好参数，所以精度很高

实验结果：

下表为第一次实验的数据，作为保留参考。上面的四个实验是第二次实验，效果更明显一点

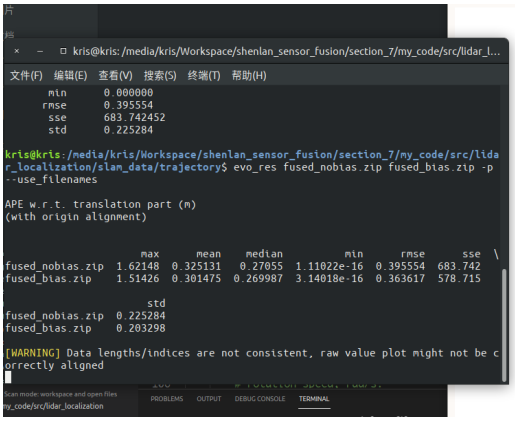
	融合max	融合RMSE	激光max	激光RMSE	现象
原始数据	1.43	0.358	3.947	2.119	-
Q减小，R不变	4.586	2.351	4.362	2.35	更相信预测
Q增大，R不变	1.466	0.421	3.947	2.117	更相信观测
R减小，Q不变	4.156	2.104	4.362	2.349	更相信观测
R增大，Q不变	4.647	2.375	4.362	2.352	更相信预测

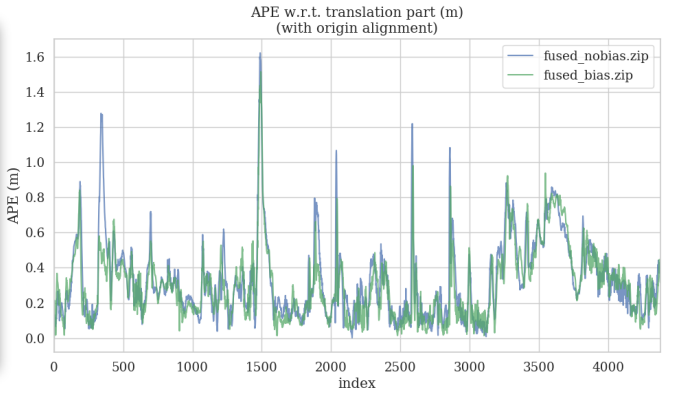
随机游走对滤波器的影响

当不考虑随机游走的噪声，矩阵B变成：

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ R_t & 0 & 0 & 0 \\ 0 & I_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

其余矩阵不变，**注意**$\delta b_a, \delta b_g$**仍有更新量**。实验结果如下图，考虑bias的噪声能提高精度：

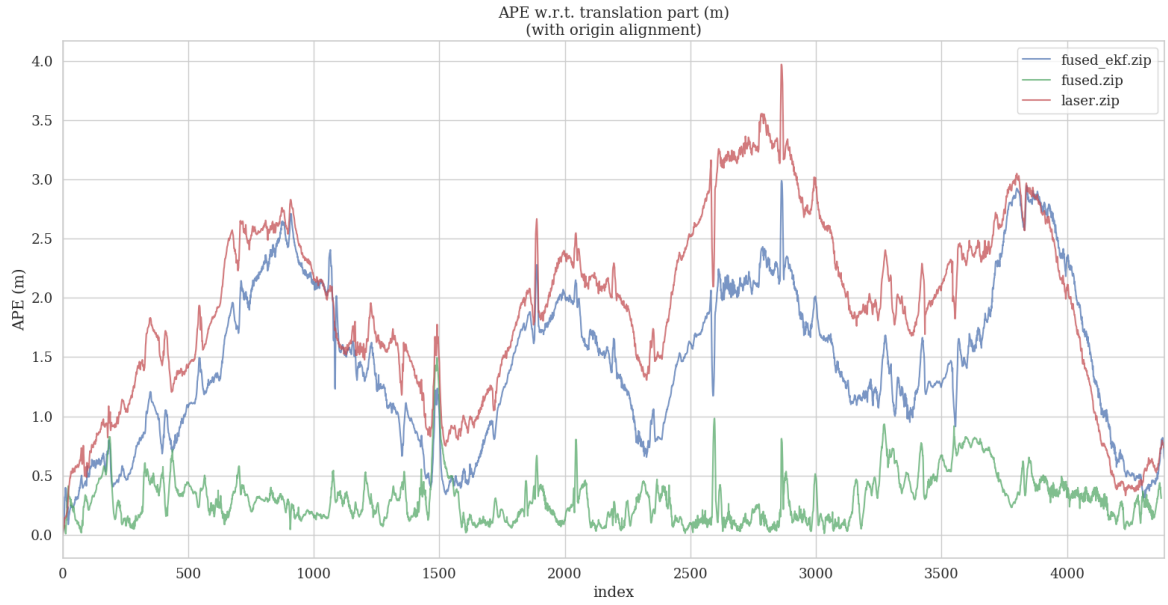




拓展

参考《Quaternion kinematics for the error-state Kalman filter》推导出名义状态的五条公式，并用程序实现。

详细公式推导写在`EKF推导.pdf`中，代码实现在`extened_kalman_filter.cpp`中。下图为实验结果：



基于名义状态的EKF对噪声更敏感，精度比ESKF要差，同时姿态数值也很不稳定，调参调得不好姿态就会飞