深蓝学院
shenlanxueyuan.com

基于融合的滤波方法II

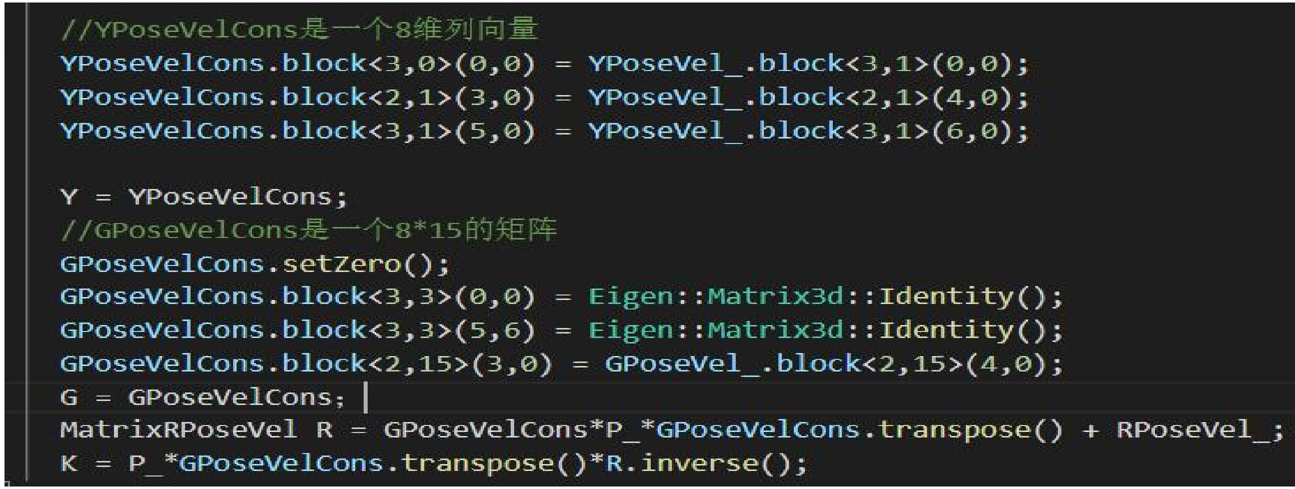
主讲人 雍川

第八章作业思路分享

相对于第七章的作业，第八章的作业相对比较简单。加入运动约束后，车体的yz轴的速度观测值为零，x轴的观测值未知（也可理解为没有x轴的观测值），因此需要去除Gt,Ct矩阵关于速度误差状态的第一行。融合运动约束的方法比较简单，就不详细阐述。

$$G_t = \begin{bmatrix} I_3 & 0 & 0 & 0 & 0 \\ 0 & [R_{bw}]_{yz} & [[v^b]_\times]_{yz} & 0 & 0 \\ 0 & 0 & I_3 & 0 & 0 \end{bmatrix}$$

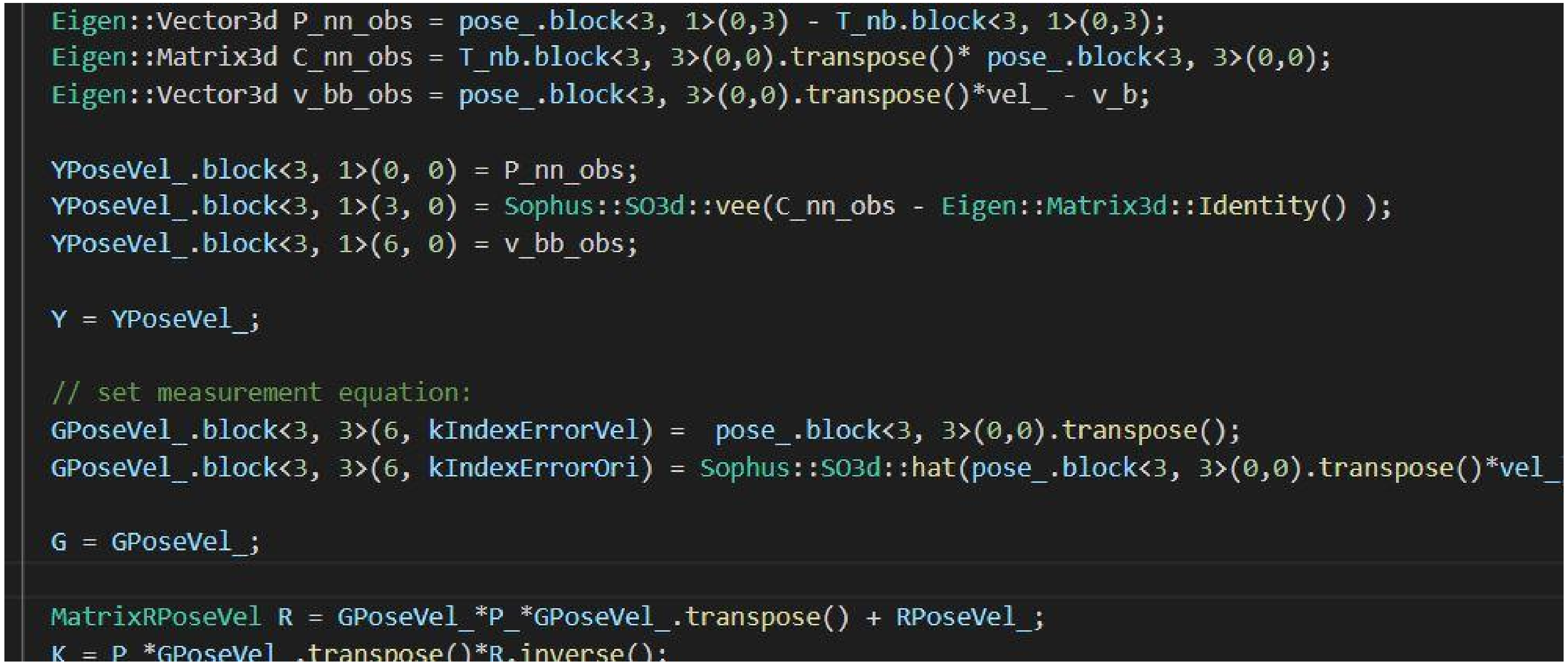
$$C_t = \begin{bmatrix} I_3 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix}$$



```
//YPoseVelCons是一个8维列向量
YPoseVelCons.block<3,0>(0,0) = YPoseVel_.block<3,1>(0,0);
YPoseVelCons.block<2,1>(3,0) = YPoseVel_.block<2,1>(4,0);
YPoseVelCons.block<3,1>(5,0) = YPoseVel_.block<3,1>(6,0);

Y = YPoseVelCons;
//GPoseVelCons是一个8*15的矩阵
GPoseVelCons.setZero();
GPoseVelCons.block<3,3>(0,0) = Eigen::Matrix3d::Identity();
GPoseVelCons.block<3,3>(5,6) = Eigen::Matrix3d::Identity();
GPoseVelCons.block<2,15>(3,0) = GPoseVel_.block<2,15>(4,0);
G = GPoseVelCons;
MatrixRPoseVel R = GPoseVelCons*P_*GPoseVelCons.transpose() + RPoseVel_;
K = P_*GPoseVelCons.transpose()*R.inverse();
```

第八章作业思路分享

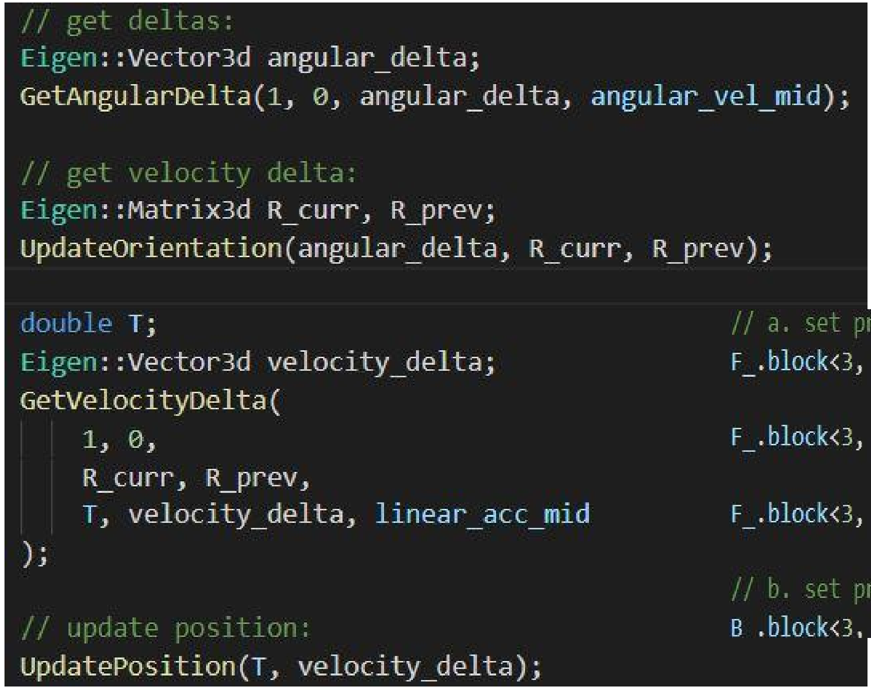
接下来就是融合编码器，编码器的融合主要是增加了车身坐标系下的速度观测值。观测方程的Gt、Ct推导大家可以查看课程PPT，主要实现如下：



```
Eigen::Vector3d P_nn_obs = pose_.block<3, 1>(0,3) - T_nb.block<3, 1>(0,3);
Eigen::Matrix3d C_nn_obs = T_nb.block<3, 3>(0,0).transpose()* pose_.block<3, 3>(0,0);
Eigen::Vector3d v_bb_obs = pose_.block<3, 3>(0,0).transpose()*vel_ - v_b;

YPoseVel_.block<3, 1>(0, 0) = P_nn_obs;
YPoseVel_.block<3, 1>(3, 0) = Sophus::SO3d::vee(C_nn_obs - Eigen::Matrix3d::Identity() );
YPoseVel_.block<3, 1>(6, 0) = v_bb_obs;

Y = YPoseVel_;

// set measurement equation:
GPoseVel_.block<3, 3>(6, kIndexErrorVel) =  pose_.block<3, 3>(0,0).transpose();
GPoseVel_.block<3, 3>(6, kIndexErrorOri) = Sophus::SO3d::hat(pose_.block<3, 3>(0,0).transpose()*vel_

G = GPoseVel_;

MatrixRPoseVel R = GPoseVel_*P_*GPoseVel_.transpose() + RPoseVel_;
K = P_*GPoseVel_.transpose()*R.inverse();
```

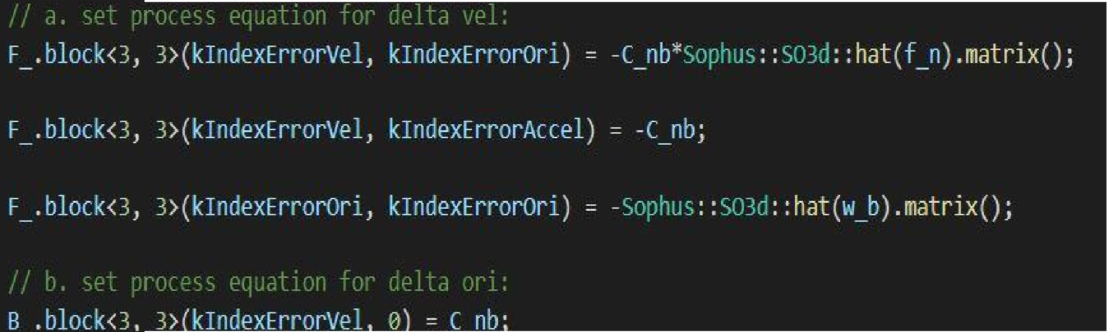
第八章作业思路分享

框架中里程计的更新函数实现如下：



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

// get velocity delta:
Eigen::Matrix3d R_curr, R_prev;
UpdateOrientation(angular_delta, R_curr, R_prev);

double T;
Eigen::Vector3d velocity_delta;
GetVelocityDelta(
    1, 0,
    R_curr, R_prev,
    T, velocity_delta, linear_acc_mid
);

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

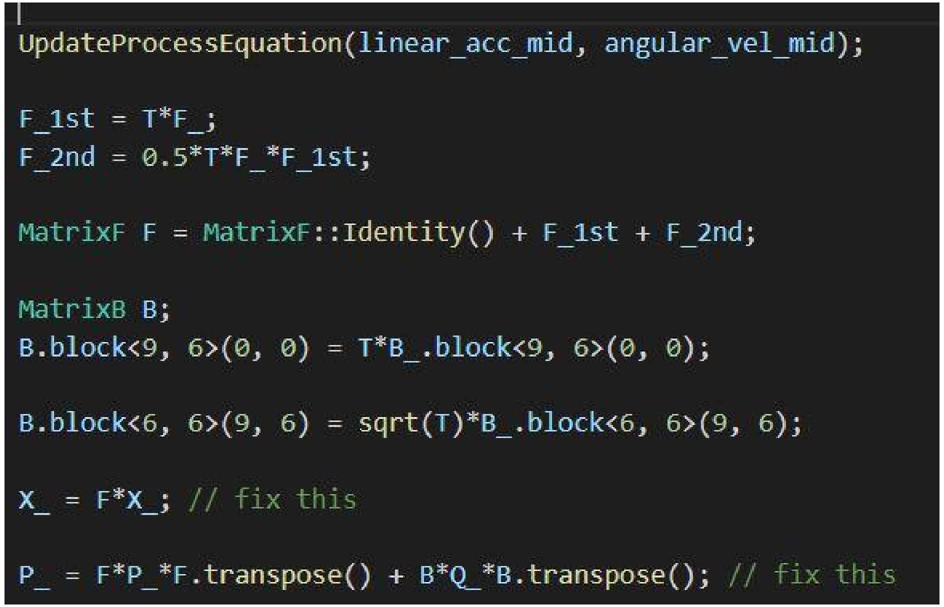
框架中函数SetProcessEquation实现如下：



```
// 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;

F_.block<3, 3>(kIndexErrorOri, kIndexErrorOri) = -Sophus::SO3d::hat(w_b).matrix();

// b. set process equation for delta ori:
B_.block<3, 3>(kIndexErrorVel, 0) = C_nb;
```

第八章作业思路分享

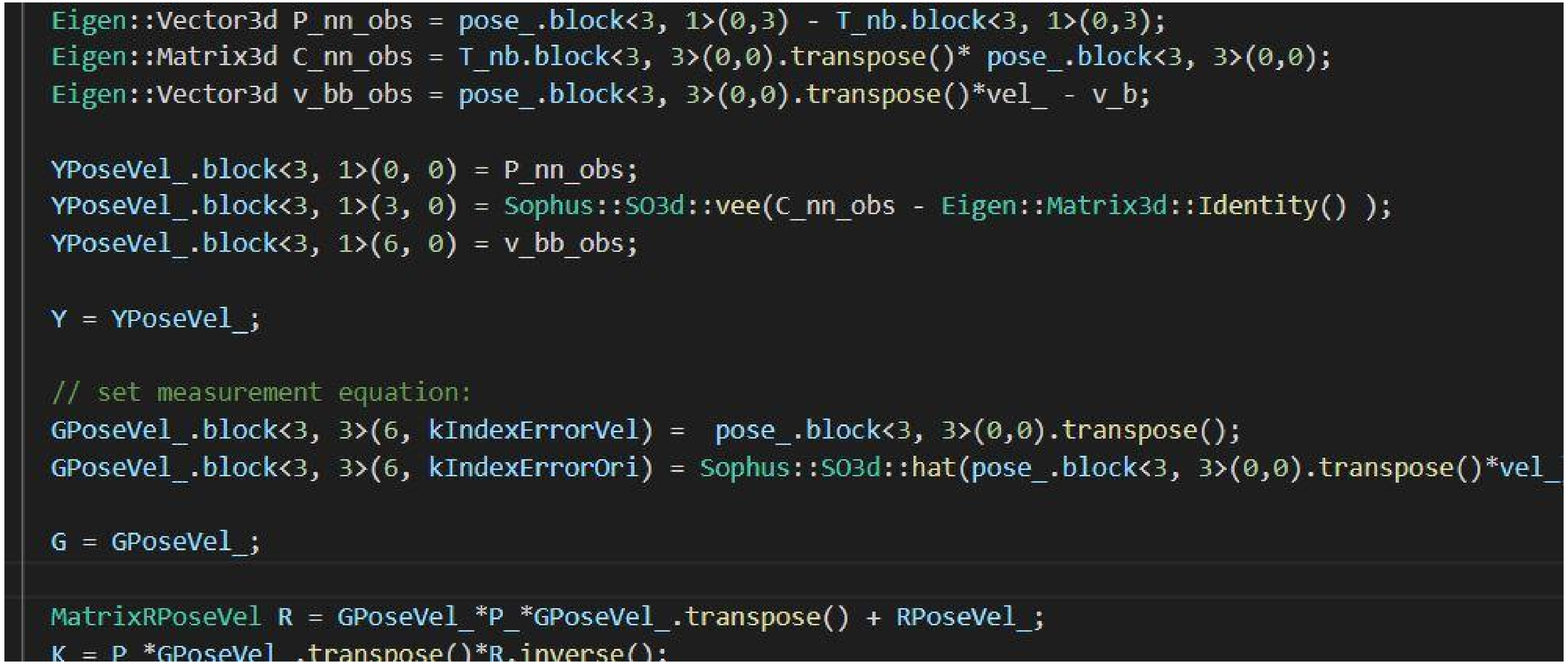
框架中UpdateErrorEstimation函数实现如下：
实现如下：



```
UpdateProcessEquation(linear_acc_mid, angular_vel_mid);

F_1st = T*F_;
F_2nd = 0.5*T*F_*F_1st;

MatrixF F = MatrixF::Identity() + F_1st + F_2nd;

MatrixB B;
B.block<9, 6>(0, 0) = T*B_.block<9, 6>(0, 0);

B.block<6, 6>(9, 6) = sqrt(T)*B_.block<6, 6>(9, 6);

X_ = F*X_; // fix this

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

第八章作业思路分享

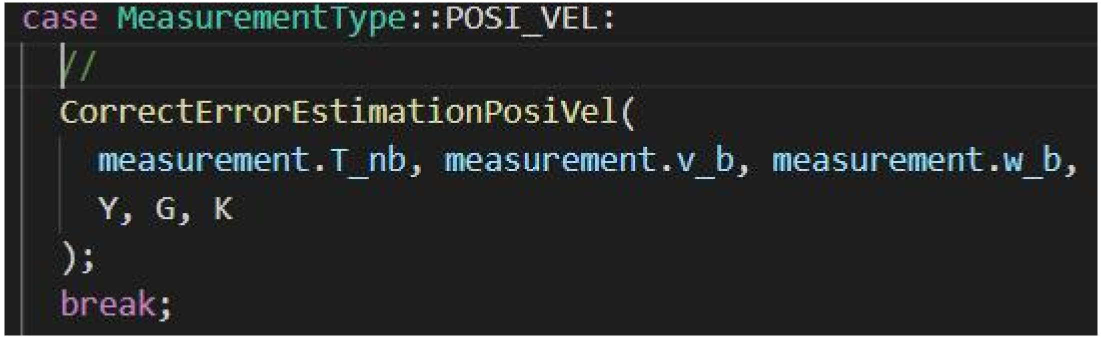
框架中CorrectErrorEstimationPose函数实现如下：



```
Eigen::Vector3d P_nn_obs = pose_.block<3, 1>(0,3) - T_nb.block<3, 1>(0,3);
Eigen::Matrix3d C_nn_obs = T_nb.block<3, 3>(0,0).transpose()* pose_.block<3, 3>(0,0);
Eigen::Vector3d v_bb_obs = pose_.block<3, 3>(0,0).transpose()*vel_ - v_b;

YPoseVel_.block<3, 1>(0, 0) = P_nn_obs;
YPoseVel_.block<3, 1>(3, 0) = Sophus::SO3d::vee(C_nn_obs - Eigen::Matrix3d::Identity() );
YPoseVel_.block<3, 1>(6, 0) = v_bb_obs;

Y = YPoseVel_;

// set measurement equation:
GPoseVel_.block<3, 3>(6, kIndexErrorVel) =  pose_.block<3, 3>(0,0).transpose();
GPoseVel_.block<3, 3>(6, kIndexErrorOri) = Sophus::SO3d::hat(pose_.block<3, 3>(0,0).transpose()*vel_

G = GPoseVel_;

MatrixRPoseVel R = GPoseVel_*P_*GPoseVel_.transpose() + RPoseVel_;
K = P_*GPoseVel_.transpose()*R.inverse();
```

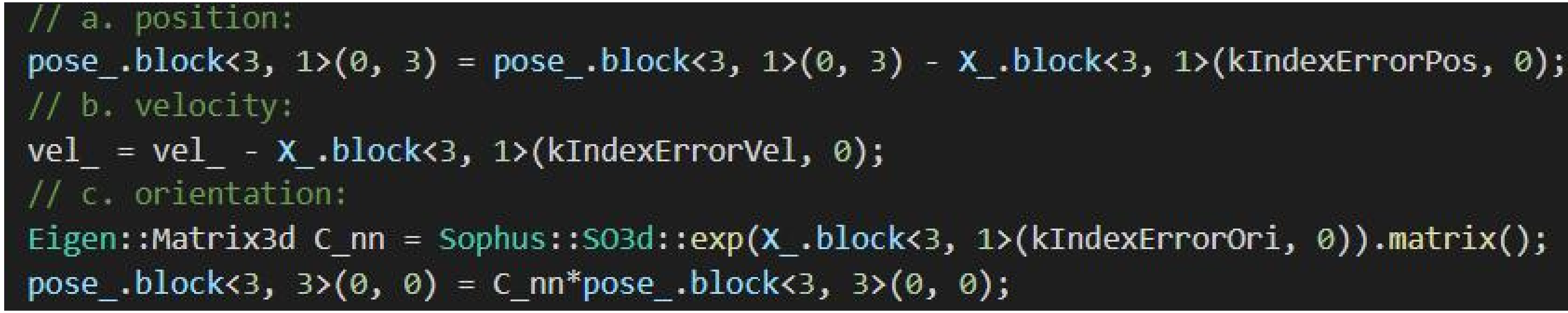
第八章作业思路分享

框架中CorrectErrorEstimation函数实现如下：



```
case MeasurementType::POSI_VEL:
    //
    CorrectErrorEstimationPosiVel(
        measurement.T_nb, measurement.v_b, measurement.w_b,
        Y, G, K
    );
    break;
```

最后，误差消除函数EliminateError实现如下：



```
// 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 C_nn = Sophus::SO3d::exp(X_.block<3, 1>(kIndexErrorOri, 0)).matrix();
pose_.block<3, 3>(0, 0) = C_nn*pose_.block<3, 3>(0, 0);
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听!

Thanks for Listening