深蓝学院
shenlanxueyuan.com

作业讨论：
基于融合的滤波方法II

主讲人 张峻诚

纲要

深蓝学院
shenlanxueyuan.com

- 第一部分：运动模型实现
- 第二部分：运动模型评估
- 第三部分：编码器融合实现

运动模型实现

深蓝学院
shenlanxueyuan.com

- 为了在body系加入在对Y和Z系的约束, 需要修改代码中的 **ErrorStateKalmanFilter::CorrectErrorEstimationPose**
- 具体公式详见PPT

```
void ErrorStateKalmanFilter::CorrectErrorEstimationPose(
    const Eigen::Matrix4d &T_nb, Eigen::VectorXd &Y, Eigen::MatrixXd &G,
    Eigen::MatrixXd &K) {
  //
  // TODO: set measurement:
  //
  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);

  // Check whether we apply the motion constraint or not based on the angular
  // rate.
  bool apply_motion_constraint =
      GetUnbiasedAngularVel(curr_raw_gyro_, pose_.block<3, 3>(0, 0)).norm() <
      0.15;
  //apply_motion_constraint = false;
  LOG(INFO) << "apply_motion_constraint? " << apply_motion_constraint;

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

  const Eigen::Matrix3d rx_robot_world = pose_.block<3, 3>(0, 0).transpose();
  const Eigen::Vector3d velocity_robot = rx_robot_world * vel_;
  if (apply_motion_constraint) {
    Y = Eigen::MatrixXd::Zero(8, 1);
    Y.block<3, 1>(0, 0) = YPose_.block<3, 1>(0, 0);
    Y.block<2, 1>(3, 0) = velocity_robot.block<2, 1>(1, 0);
    Y.block<3, 1>(5, 0) = YPose_.block<3, 1>(3, 0);
  } else {
    Y = YPose_;
  }
```

运动模型实现

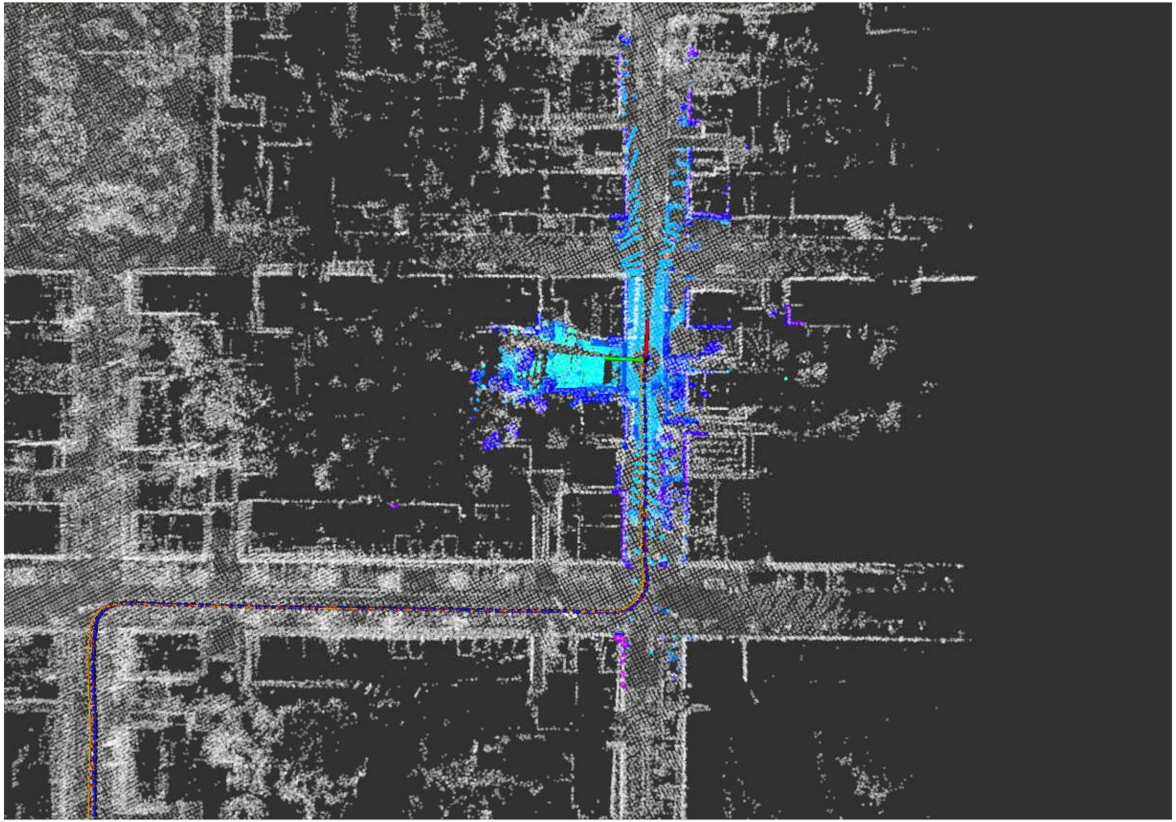
深蓝学院 shenlanxueyuan.com

- 我发现在转弯时，ESKF 观测的 body 系下的速度在 Y 和 Z 轴的分量并不总是接近 0. 我猜测这是因为 ESKF 实际估计的是 imu 的速度，而不是在真正质心的速度，所以在旋转时会 有速度分量．在我上面的实现中，我首先检测了当前的旋转角速度的大小，当其值过大时则不强加在 Y 和 Z 轴上的限制.

```
    // set measurement equation:
    if (apply_motion_constraint) {
        G = Eigen::MatrixXd::Zero(8, 15);
        G.block<3, 3>(0, INDEX_ERROR_POS) = Eigen::Matrix3d::Identity();
        G.block<2, 3>(3, INDEX_ERROR_VEL) = rx_robot_world.block<2, 3>(1, 0);
        G.block<2, 3>(3, INDEX_ERROR_ORI) =
            Sophus::SO3d::hat(velocity_robot).block<2, 3>(1, 0);
        G.block<3, 3>(5, INDEX_ERROR_ORI) = Eigen::Matrix3d::Identity();
    } else {
        G = GPose_;
    }

    Eigen::MatrixXd R;
    if (apply_motion_constraint) {
        R = Eigen::MatrixXd::Zero(8, 8);
        R.block<3, 3>(0, 0) = RPose_.block<3, 3>(0, 0);
        R.block<2, 2>(3, 3) = Eigen::Matrix2d::Identity() * 0.01;
        R.block<3, 3>(5, 5) = RPose_.block<3, 3>(3, 3);
    } else {
        R = CPose_ * RPose_ * CPose_.transpose();
    }

    //
    // TODO: set Kalman gain:
    //
    K = P_ * G.transpose() * (G * P_ * G.transpose() + R).inverse();
}
```

运动模型实现

- 模型的运行截图为:



纲要

深蓝学院
shenlanxueyuan.com

- 第一部分：运动模型实现
- 第二部分：运动模型评估
- 第三部分：编码器融合实现

运动模型评估

- 为了对新模型做出评价，我首先更改代码, 将在 body 系中的速度保存下来. 我定义了一个新的 rosservice, 叫 做 save_velocity.

- 具体实现如右图所示

```cpp
bool KITTIFilteringFlow::SaveVelocity(void) {
  if (trajectory.N == 0) return false;

  std::ofstream robot_velocity_ofs;
  if (!FileManager::CreateFile(
        robot_velocity_ofs,
        WORK_SPACE_PATH + "/slam_data/trajectory/robot_velocity.txt")) {
    return false;
  }

  const double init_time = trajectory.time_.at(0);
  for (size_t i = 0; i < trajectory.N; ++i) {
    const double time = trajectory.time_.at(i) - init_time;
    const Eigen::Matrix4f& fused_pose = trajectory.fused_.at(i);
    const Eigen::Vector3f& fused_vel = trajectory.fused_vel.at(i);
    const Eigen::Vector3f robot_vel =
      fused_pose.block<3, 3>(0, 0).transpose() * fused_vel;
    // Save robot vel into the output.
    robot_velocity_ofs << time << " " << robot_vel.x() << " " << robot_vel.y()
                       << " " << robot_vel.z() << std::endl;
  }
  return true;
}
```

运动模型评估

- 最后为了更好分析在 Y 轴和 Z 轴的速度分量，我写了如右图所示的 python 代码 .
- 该代码会把各个轴的分量画图显示，并计算 Y 轴和 Z 轴速度的统计值 .

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data_path = 'slam_data/trajectory/motion_constraint/robot_velocity7.txt'
robot_velocity = pd.read_csv(
    data_path,
    sep=' ',
    names=['timestamp', 'velocity_x', 'velocity_y', 'velocity_z'])\

plt.subplot(3, 1, 1)
plt.plot(robot_velocity['timestamp'],
         robot_velocity['velocity_x'],
         label='velocity_x')
plt.title('VelocityX vs time')
plt.legend(loc='upper right')

# Calculate min, max, mean, std for velocity y.
vel_y = np.array(robot_velocity['velocity_y'])
print('Velocity Y stats:')
print('min: ', vel_y.min())
print('max: ', vel_y.max())
print('mean: ', np.mean(vel_y))
print('median: ', np.median(vel_y))
print('std: ', np.std(vel_y))
plt.subplot(3, 1, 2)
plt.plot(robot_velocity['timestamp'],
         robot_velocity['velocity_y'],
         label='velocity_y')
plt.title('VelocityY vs time')
plt.legend(loc='upper right')

# Calculate min, max, mean, std for velocity z.
vel_z = np.array(robot_velocity['velocity_z'])
print('Velocity Z stats:')
print('min: ', vel_z.min())
print('max: ', vel_z.max())
print('mean: ', np.mean(vel_z))
print('median: ', np.median(vel_z))
print('std: ', np.std(vel_z))
plt.subplot(3, 1, 3)
plt.plot(robot_velocity['timestamp'],
         robot_velocity['velocity_z'],
         label='velocity_z')
plt.title('VelocityZ vs time')
plt.legend(loc='upper right')

plt.show()
```

运动模型评估

- 在未加入速度约束时，evo_ape 的结果如右图所示:

```
root@7efda655577c:/workspace/assignments/06-filtering-basic/src/lidar_localiza
tion/slam_data/trajectory/motion_constraint# evo_ape kitti ground_truth5.txt f
used5.txt -r full --plot --plot_mode xyz
APE w.r.t. full transformation (unit-less)
(not aligned)

       max      1.097528
      mean      0.253381
    median      0.196732
       min      0.020702
      rmse      0.306149
       sse      411.088207
       std      0.171829
```

- 对其速度分量进行分析结果为:

```
Velocity Y stats:
min:  -0.941034
max:  0.8375360000000001
mean:  0.05523174418552211
median:  0.05680075
std:  0.2102908964584072
Velocity Z stats:
min:  -0.44849700000000003
max:  0.677524
mean:  0.08759630227274282
median:  0.08312605
std:  0.1443409273883334
```

运动模型评估

- 当在 body 系速度的 Y 轴和 Z 轴加入 variance 为 0.01 的观测约束后 . evo_ape 的结果为 :

```
root@7efda655577c:/workspace/assignments/06-filtering-basic/src/lidar_localiza
tion/slam_data/trajectory/motion_constraint# evo_ape kitti ground_truth8.txt f
used8.txt -r full --plot --plot_mode xyz
APE w.r.t. full transformation (unit-less)
(not aligned)

       max      1.090655
      mean      0.252798
    median      0.197146
       min      0.018921
      rmse      0.305777
       sse      410.087980
       std      0.172025
```

- 对其速度分量进行分析结果为:

```
Velocity Y stats:
min:  -0.9441579999999999
max:  0.850617
mean:  0.03716084245414957
median:  0.0326878
std:  0.20031082206904963
Velocity Z stats:
min:  -0.465204
max:  0.51263
mean:  0.05022094725136799
median:  0.0516732
std:  0.11984731032703487
```

运动模型评估

- 我还尝试了加入更强的速度约束，比如将速度分量的方差改为0.0004(std = 0.02).
  此时 evo_ape 的结果为：

```
root@7efda655577c:/workspace/assignments/06-filtering-basic/src/lidar_localiza
tion/slam_data/trajectory/motion_constraint# evo_ape kitti ground_truth7.txt f
used7.txt -r full --plot --plot_mode xyz
APE w.r.t. full transformation (unit-less)
(not aligned)

       max      1.103153
      mean      0.253840
    median      0.197939
       min      0.018235
      rmse      0.306245
       sse      411.814667
       std      0.171323
```

- 对其速度分量进行分析结果为:

```
Velocity Y stats:
min:  -0.957677
max:  0.806966
mean:  0.016980173016057844
median:  0.00459146
std:  0.17989495922193122
Velocity Z stats:
min:  -0.477475
max:  0.44374399999999997
mean:  0.000662094089182418
median:  0.00356599
std:  0.0710874561437351
```

- 和之前两组结果对比，可以看出 evo_ape 结果仍然类似，但并没有更好．而其速度分量则更接近于 0.

纲要

- 第一部分：运动模型实现
- 第二部分：运动模型评估
- 第三部分：编码器融合实现

编码器融合实现

- 为了实现以 gps 位置和编码器速度为观测量的融合方法，首先应该更改 CorrectErrorEstimation, 使得这两种观测量可以被处理 .
- 注意之前我们实现的是 **CorrectErrorEstimationPose** 然而现在因为只有gps position 和 velocity的观测, 我们需要定义 **CorrectErrorEstimation PositionVelocity**

```
void ErrorStateKalmanFilter::CorrectErrorEstimation(
    const MeasurementType &measurement_type, const Measurement &measurement) {
  //
  // TODO: understand ESKF correct workflow
  //
  Eigen::VectorXd Y;
  Eigen::MatrixXd G, K;
  switch (measurement_type) {
    case MeasurementType::POSE:
      CorrectErrorEstimationPose(measurement.T_nb, Y, G, K);
      break;
    case MeasurementType::POSI_VEL:
      CorrectErrorEstimationPositionVelocity(measurement.T_nb, measurement.v_b,
                                             Y, G, K);
      break;
    default:
      return;
  }

  //
  // TODO: perform Kalman correct:
  //
  P_ = (MatrixP::Identity() - K * G) * P_;
  X_ = X_ + K * (Y - G * X_);
  // LOG(INFO) << "Calculated delta_x: " << X_;
}
```

编码器融合实现

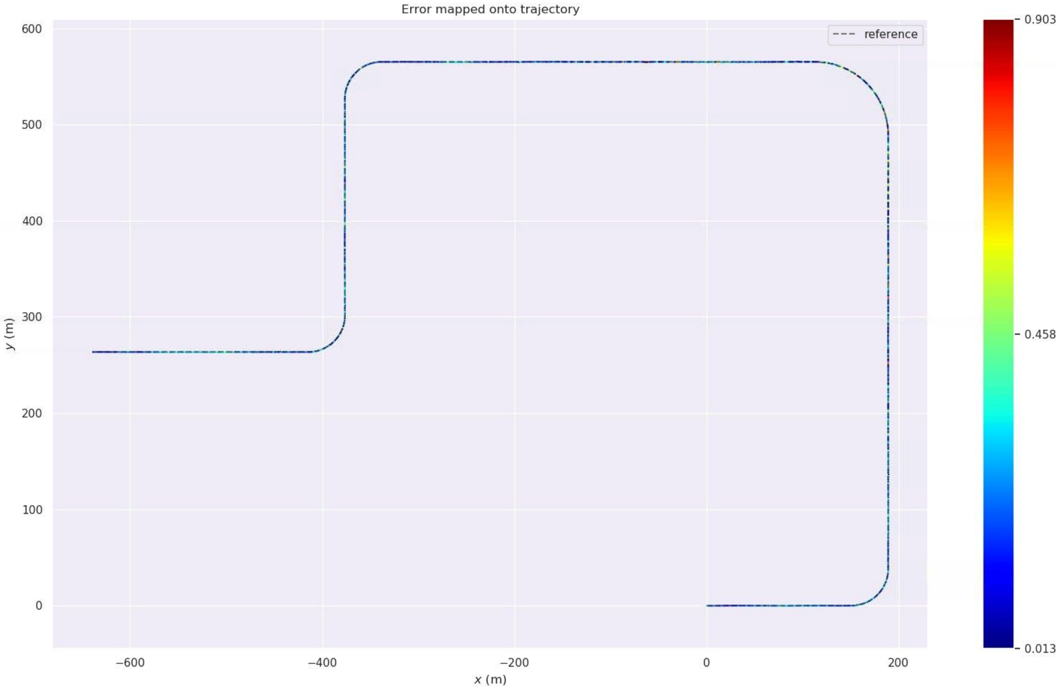
- 真正进行 update 的函数 **CorrectErrorEstimationPositionVelocity** 定义如右图所示：

```cpp
void ErrorStateKalmanFilter::CorrectErrorEstimationPositionVelocity(
    const Eigen::Matrix4d &T_nb, const Eigen::Vector3d &v_b, Eigen::VectorXd &Y,
    Eigen::MatrixXd &G, Eigen::MatrixXd &K) {
  Eigen::Vector3d P_nn_obs = pose_.block<3, 1>(0, 3) - T_nb.block<3, 1>(0, 3);
  const Eigen::Matrix3d rx_robot_world = pose_.block<3, 3>(0, 0).transpose();
  const Eigen::Vector3d velocity_robot = rx_robot_world * vel_;
  Eigen::Vector3d v_b_nn_obs = velocity_robot - v_b;

  Y = Eigen::MatrixXd::Zero(6, 1);
  Y.block<3, 1>(0, 0) = P_nn_obs;
  Y.block<3, 1>(3, 0) = v_b_nn_obs;

  const bool is_turning =
      GetUnbiasedAngularVel(curr_raw_gyro_, pose_.block<3, 3>(0, 0)).norm() >
      0.15;
  G = Eigen::MatrixXd::Zero(6, 15);
  G.block<3, 3>(0, INDEX_ERROR_POS) = Eigen::Matrix3d::Identity();
  G.block<3, 3>(3, INDEX_ERROR_VEL) = rx_robot_world;
  G.block<3, 3>(3, INDEX_ERROR_ORI) = Sophus::SO3d::hat(velocity_robot);

  Eigen::MatrixXd R = RPosiVel_;
  // Set kalman gain.
  K = P_ * G.transpose() * (G * P_ * G.transpose() + R).inverse();
}
```

编码器融合实现

- 利用evo_ape来评估误差结果为:

```
root@7efda655577c:/workspace/assignments/06
tion/slam_data/trajectory# evo_ape kitti gr
--plot --plot_mode xyz
APE w.r.t. full transformation (unit-less)
(not aligned)

       max      0.902714
      mean      0.242863
    median      0.227156
       min      0.013237
      rmse      0.268721
       sse      112.504897
       std      0.115016
```



在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening