第8章作业

一. 运动约束模型

编码器可以额外提供一个轮速里程计，在编码器不是很高频的情况下，可以做为一个约束边，通过提供线速度(b系) 到上一章节的观测方程中，进行kalman融合。

注：

1. 编码器提供的线速度是比较准确的，但是角速度不太准确(转弯存在打滑现象)，角速度不宜用作观测边。
2. 编码器参与的融合，还有另外一种融合方式，即编码器不当做观测使用，而是和IMU一起进行状态预测，然后再与其他传感器提供的观测进行滤波融合。具体思路为IMU提供角速度，编码器提供线速度，假设二者频率相同、时间戳已对齐,且外参已标定，那么它们可以直接被认为是一个通过解算后得到姿态、位置的新传感器。

车子坐标系(前左上)，在没有编码器硬件的基础上，可以使用四轮底盘本身的运动属性进行约束，正常情况下，车子只有前向x的速度，观测上，y 和 z 向的观测都为0。所以可在观测中添加 Vy 、 Vz 两个约束边。

代码编写

FILE：lidar_localization/src/models/kalman_filter/error_state_kalman_filter.cpp

添加运动约束

FUNCTION: ErrorStateKalmanFilter::CorrectErrorEstimationPosiVel

```
void ErrorStateKalmanFilter::CorrectErrorEstimationPosiVel(       //  position + velocity
    const Eigen::Matrix4d &T_nb, const Eigen::Vector3d &v_b, const Eigen::Vector3d &w_b,
    Eigen::VectorXd &Y, Eigen::MatrixXd &G, Eigen::MatrixXd &K
) {
    //
    // TODO: set measurement:        计算观测 delta pos  、 delta  velocity
    //
    Eigen::Vector3d  v_b_  =  {v_b[0],  0,  0};            //  measurment velocity  (body  系) ，  伪观测 (vy 、vz  = 0)

    Eigen::Vector3d  dp  =  pose_.block<3,  1>(0,  3)  -  T_nb.block<3,  1>(0, 3);
    Eigen::Vector3d  dv  =   pose_.block<3,  3>(0, 0).transpose() *vel_  -  v_b ;                   //  delta v  ,  v_x 来自轮速里程计
    // TODO: set measurement equation:
    YPosiVel_.block<3, 1>(0, 0)  =  dp;          //   delta  position
    YPosiVel_.block<3, 1>(3, 0)  =  dv;           //   delta   velocity
    Y = YPosiVel_;
    //   set measurement  G
    GPosiVel_.setZero();
    GPosiVel_.block<3, 3>(0, kIndexErrorPos)  =  Eigen::Matrix3d::Identity();
    GPosiVel_.block<3, 3>(3, kIndexErrorVel)   =   pose_.block<3,  3>(0, 0).transpose();
```

```
    GPosiVel_.block<3, 3>(3, kIndexErrorOri)  =  Sophus::SO3d::hat(
pose_.block<3,  3>(0, 0).transpose() *vel_  ) ;
    G  =  GPosiVel_;
    //  set measurement  C
    CPosiVel_.setIdentity();
    Eigen::MatrixXd  C  =  CPosiVel_;
    // TODO: set Kalman gain:
    Eigen::MatrixXd R = RPosiVel_;   //  观测噪声
    K =  P_  *  G.transpose() * ( G  *  P_  *  G.transpose( )  +  C * R*
 C.transpose() ).inverse() ;
}
```

添加惯导速度观测

FUNCTION: ErrorStateKalmanFilter::CorrectErrorEstimationPoseVel

```
void ErrorStateKalmanFilter::CorrectErrorEstimationPoseVel(        //  计算  Y
, G  , K
    const Eigen::Matrix4d &T_nb, const Eigen::Vector3d &v_b, const
Eigen::Vector3d &w_b,
    Eigen::VectorXd &Y, Eigen::MatrixXd &G, Eigen::MatrixXd &K
) {
    //
    // TODO: set measurement:     计算观测 delta pos  、 delta  ori
    //
    // Eigen::Vector3d  v_b_  =  {v_b[0],  0,  0};          //  measurment
velocity  (body  系) ,  伪观测 (vy 、vz  = 0)
    Eigen::Vector3d  v_b_  =  v_b;          //  measurment  velocity  (body
系) ,  融入速度 (vx 取自 惯导)

    Eigen::Vector3d  dp  =  pose_.block<3,  1>(0,  3)  -  T_nb.block<3,  1>(0,
 3);
    Eigen::Matrix3d  dR  =  T_nb.block<3,  3>(0, 0).transpose() *
 pose_.block<3, 3>(0, 0) ;
    Eigen::Vector3d  dv  =  pose_.block<3,  3>(0, 0).transpose() *vel_  -  v_b_
;              //  delta v      严格意义上来说，这里的观测是，惯导给的vx
    // TODO: set measurement equation:
    Eigen::Vector3d  dtheta  =  Sophus::SO3d::vee(dR  -
 Eigen::Matrix3d::Identity() );
    YPoseVel_.block<3, 1>(0, 0)  =  dp;        //   delta  position
    YPoseVel_.block<3, 1>(3, 0)  =  dv;         //  delta   velocity
    YPoseVel_.block<3, 1>(6, 0)  =  dtheta;        //   失准角
    Y = YPoseVel_;
    //  set measurement  G
    GPoseVel_.setZero();
    GPoseVel_.block<3, 3>(0, kIndexErrorPos)  =  Eigen::Matrix3d::Identity();
    GPoseVel_.block<3, 3>(3, kIndexErrorVel)  =  pose_.block<3,  3>(0,
0).transpose();
    GPoseVel_.block<3, 3>(3, kIndexErrorOri)  =  Sophus::SO3d::hat(
pose_.block<3,  3>(0, 0).transpose() *vel_  ) ;
    GPoseVel_.block<3 ,3>(6, kIndexErrorOri)  =  Eigen::Matrix3d::Identity();

    G  =  GPoseVel_;
    //  set measurement  C
    CPoseVel_.setIdentity();
    Eigen::MatrixXd  C  =  CPoseVel_;
    // TODO: set Kalman gain:
```

```
    Eigen::MatrixXd R = RPoseVel_;    //  观测噪声
    K =  P_  *  G.transpose() * ( G  *  P_  *  G.transpose( )  +  C * RPoseVel_*
 C.transpose() ).inverse() ;
}
```

调用

FUNCTION: ErrorStateKalmanFilter::CorrectErrorEstimation

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

  //TODO: register new correction logic here:

    CorrectErrorEstimationPosiVel(
        measurement.T_nb,
        measurement.v_b, measurement.w_b,
         Y, G, K
    );
    break;
  default:
    break;
  }

  //
  // TODO: perform Kalman correct:
  P_ = (MatrixP::Identity() -  K*G)  *  P_ ;          //  后验方差
  X_ =  X_ +  K * (Y - G*X_);
   //  更新后的状态量
}
```

运行

代码运行命令:

```
roslaunch lidar_localization kitti_localization.launch
```

播放数据集命令:

```
rosbag  play kitti_lidar_only_2011_10_03_drive_0027_synced.bag
```

保存里程计:

```
rosservice call /save_odometry
```

数据位于:

```
src/lidar_localization/slam_data/trajectory
```

evo工具运行命令:

```
# a. fused 没有输入运动模型  输出评估结果,并以zip的格式存储:
evo_ape kitti ground_truth.txt fused.txt -r full --plot --plot_mode xy  --
save_results ./fused.zip
# b. fused_vel 速度观测  输出评估结果,并以zip的格式存储:
evo_ape kitti ground_truth.txt fused.txt -r full --plot --plot_mode xy  --
save_results ./fused_vel.zip
# c. fused_cons 运动约束伪观测  输出评估结果,并以zip的格式存储:
evo_ape kitti ground_truth.txt fused.txt -r full --plot --plot_mode xy  --
save_results ./fused_cons.zip
# d. 比较 laser  fused  一并比较评估
evo_res  *.zip -p
```

注:

三个不同的模型需要修改如下:

FILE: lidar_localization/src/filtering/kitti_filtering.cpp

FUNCTION: KITTIFiltering::Correct

```
  // Kalman correction:
  if (kalman_filter_ptr_->Correct(imu_data,
KalmanFilter::MeasurementType::POSE_VEL,
                                  current_measurement_)) {
    kalman_filter_ptr_->GetOdometry(current_pose_, current_vel_);
```

三个参数如下:

KalmanFilter::MeasurementType::POSE   没有输入运动模型

KalmanFilter::MeasurementType::POSE_VEL 速度观测

KalmanFilter::MeasurementType::POSI_VEL 运动约束模型

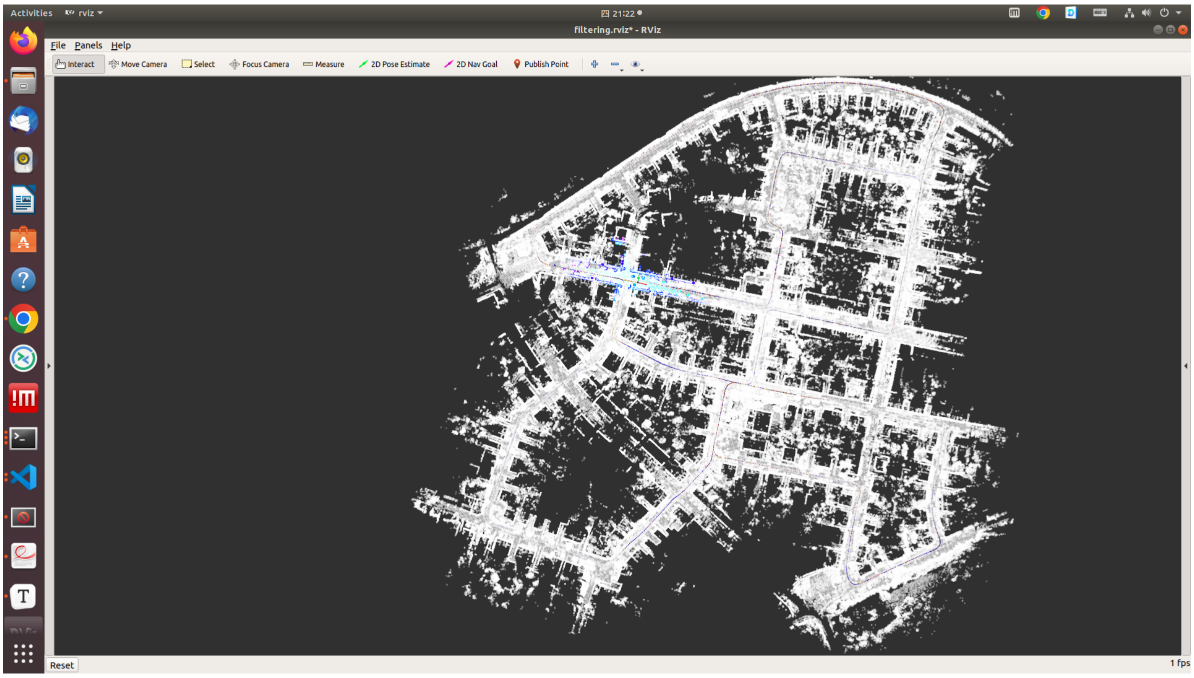
参数

config/filtering/kitti_filtering.yaml

```yaml
# 全局地图
map_path:
/home/qjs/code/ROS_Localization/shenlan/04/global_localization_chapter4_ws/src/slam_data/map/filtered_map.pcd
global_map_filter: voxel_filter # 选择滑窗地图点云滤波方法，目前支持：voxel_filter、no_filter


# 局部地图
# 局部地图从全局地图切割得到，此处box_filter_size是切割区间
# 参数顺序是min_x, max_x, min_y, max_y, min_z, max_z
box_filter_size: [-150.0, 150.0, -150.0, 150.0, -150.0, 150.0]
local_map_filter: voxel_filter # 选择滑窗地图点云滤波方法，目前支持：voxel_filter、no_filter

# 当前帧
# no_filter指不对点云滤波，在匹配中，理论上点云越稠密，精度越高，但是速度也越慢
# 所以提供这种不滤波的模式做为对比，以方便使用者去体会精度和效率随稠密度的变化关系
current_scan_filter: voxel_filter # 选择当前帧点云滤波方法，目前支持：voxel_filter、no_filter

# loop closure for localization initialization/re-initialization:
loop_closure_method: scan_context # 选择回环检测方法, 目前支持scan_context
scan_context_path:
/home/kaho/shenlan_ws/src/lidar_localization/slam_data/scan_context

# 匹配
registration_method: NDT   # 选择点云匹配方法，目前支持：NDT

# select fusion method for IMU-GNSS-Odo-Mag, available methods are:
#     1. error_state_kalman_filter
fusion_method: error_state_kalman_filter
# select fusion strategy for IMU-GNSS-Odo-Mag, available methods are:
#     1. pose_velocity   2.pose
fusion_strategy: pose_velocity

# 各配置选项对应参数
## a. point cloud filtering:
voxel_filter:
    global_map:
        leaf_size: [0.9, 0.9, 0.9]
    local_map:
        leaf_size: [0.5, 0.5, 0.5]
    current_scan:
        leaf_size: [1.5, 1.5, 1.5]
## b. scan context:
scan_context:
    # a. ROI definition:
    max_radius: 80.0
    max_theta: 360.0
    # b. resolution:
    num_rings: 20
    num_sectors: 60
```

```
    # c. ring key indexing interval:
    indexing_interval: 1
    # d. min. key frame sequence distance:
    min_key_frame_seq_distance: 100
    # e. num. of nearest-neighbor candidates to check:
    num_candidates: 5
    # f. sector key fast alignment search ratio:
    #   avoid brute-force match using sector key
    fast_alignment_search_ratio: 0.1
    # g. scan context distance threshold for proposal generation:
    #   0.4-0.6 is good choice for using with robust kernel (e.g., Cauchy, DCS) +
icp fitness threshold
    #   if not, recommend 0.1-0.15
    scan_context_distance_thresh: 0.15
## c. frontend matching
NDT:
    res : 1.0
    step_size : 0.1
    trans_eps : 0.01
    max_iter : 30
## d. Kalman filter for IMU-lidar-GNSS fusion:
## d.1. Error-State Kalman filter for IMU-GNSS-Odo fusion:
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-4
            accel: 2.5e-3
            bias_accel: 2.5e-3
            bias_gyro: 1.0e-4
        measurement:
            pose:
                pos: 1.0e-4
                ori: 1.0e-4
            pos: 1.0e-4
            vel: 2.5e-3
    motion_constraint:
        activated: true
        w_b_thresh: 0.13
```

RVIZ效果



EVO评估

没有输入运动模型

```
       max	1.053713
      mean	0.248408
    median	0.193839
       min	0.018872
      rmse	0.299533
       sse	391.357809
       std	0.167372
```

速度观测

```
       max	0.997094
      mean	0.250137
    median	0.198706
       min	0.017371
      rmse	0.300098
       sse	392.655437
       std	0.165801
```

运动约束模型

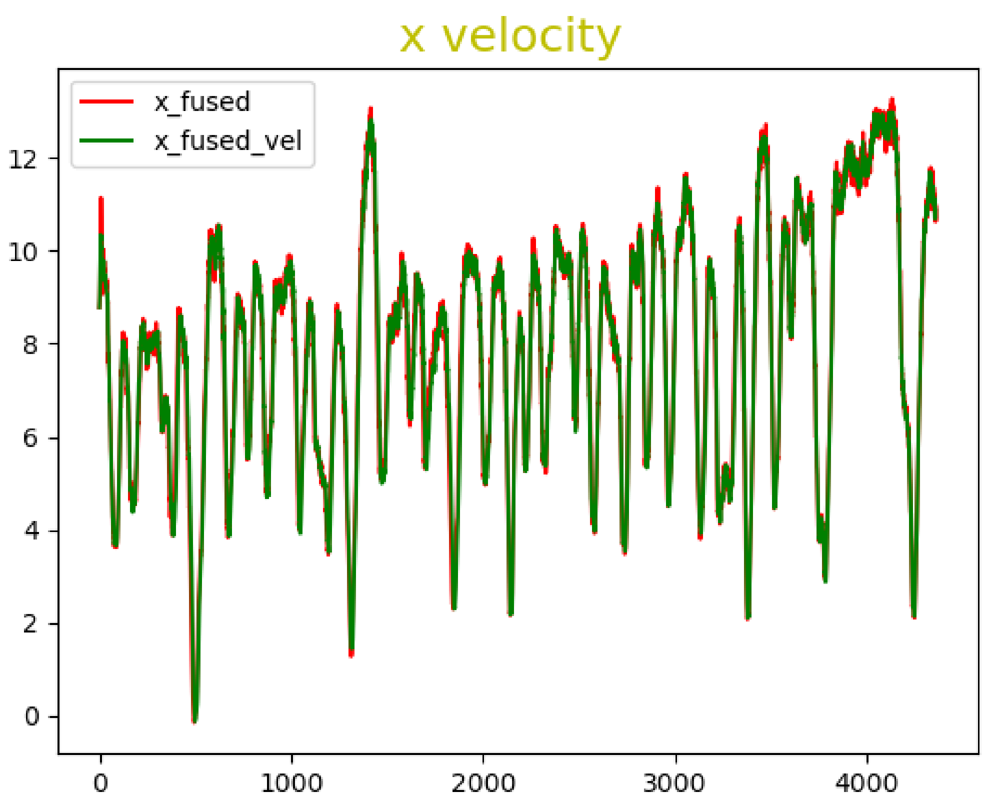
```
   max	1.084376
  mean	0.253829
median	0.200637
   min	0.023816
  rmse	0.302779
   sse	401.078977
   std	0.165063
```

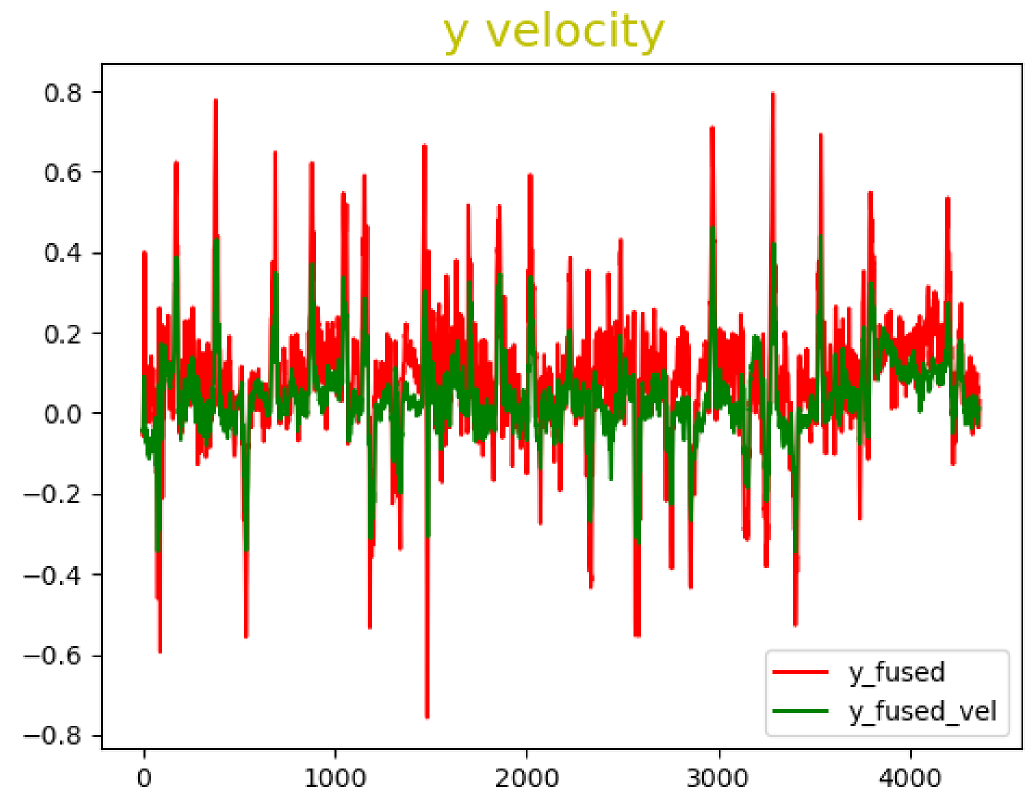
对比：

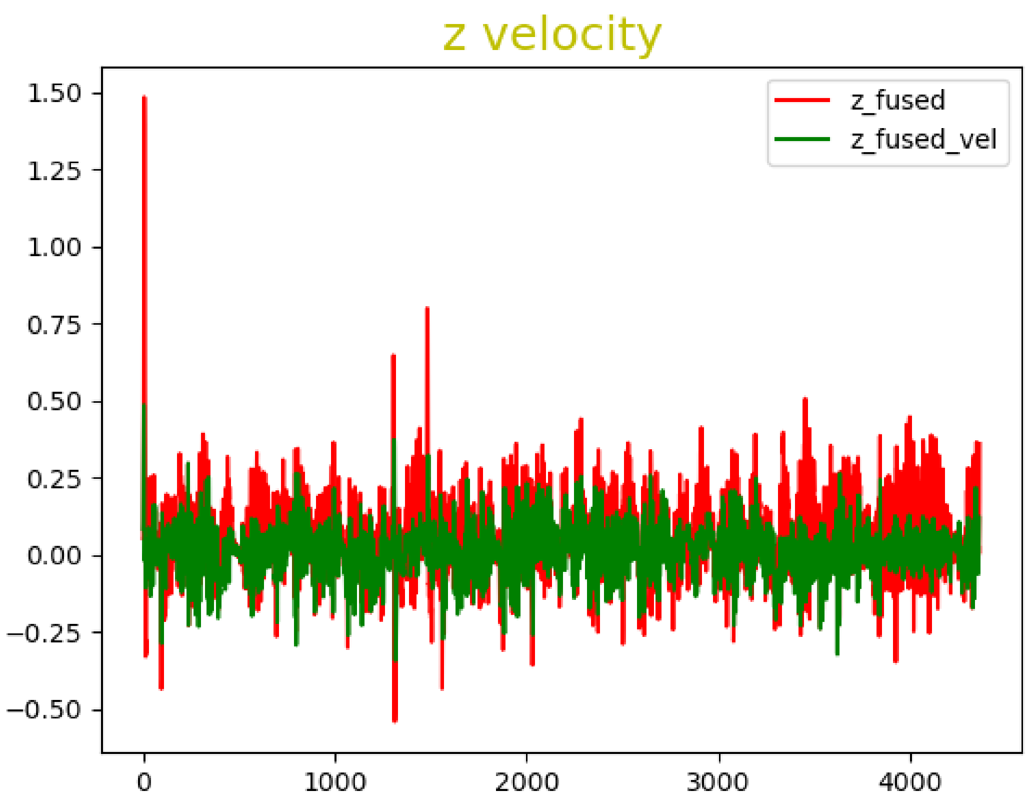
```
                     rmse      mean    median       std       min       max  \
fused_cons.zip   0.302779  0.253829  0.200637  0.165063  0.023816  1.084376
fused_vel.zip    0.300098  0.250137  0.198706  0.165801  0.017371  0.997094
fused.zip        0.299533  0.248408  0.193839  0.167372  0.018872  1.053713

                        sse
fused_cons.zip   401.078977
fused_vel.zip    392.655437
fused.zip        391.357809
```

matplotlib 可视化数据

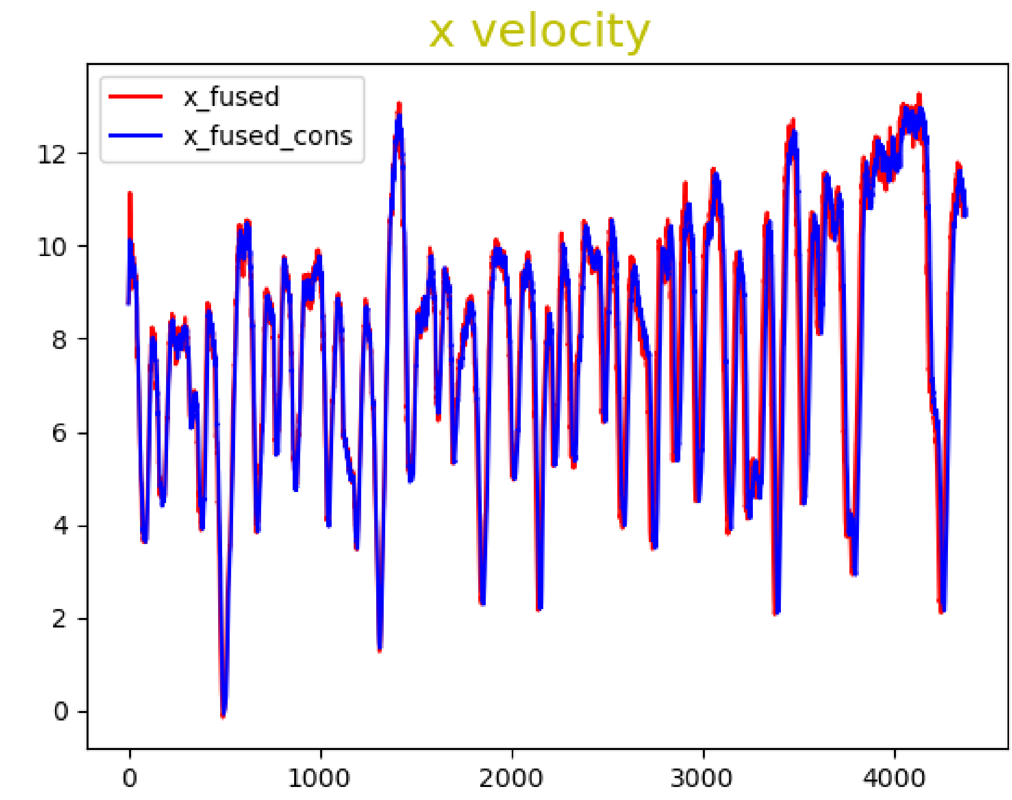
与加了速度观测的模型进行比较：

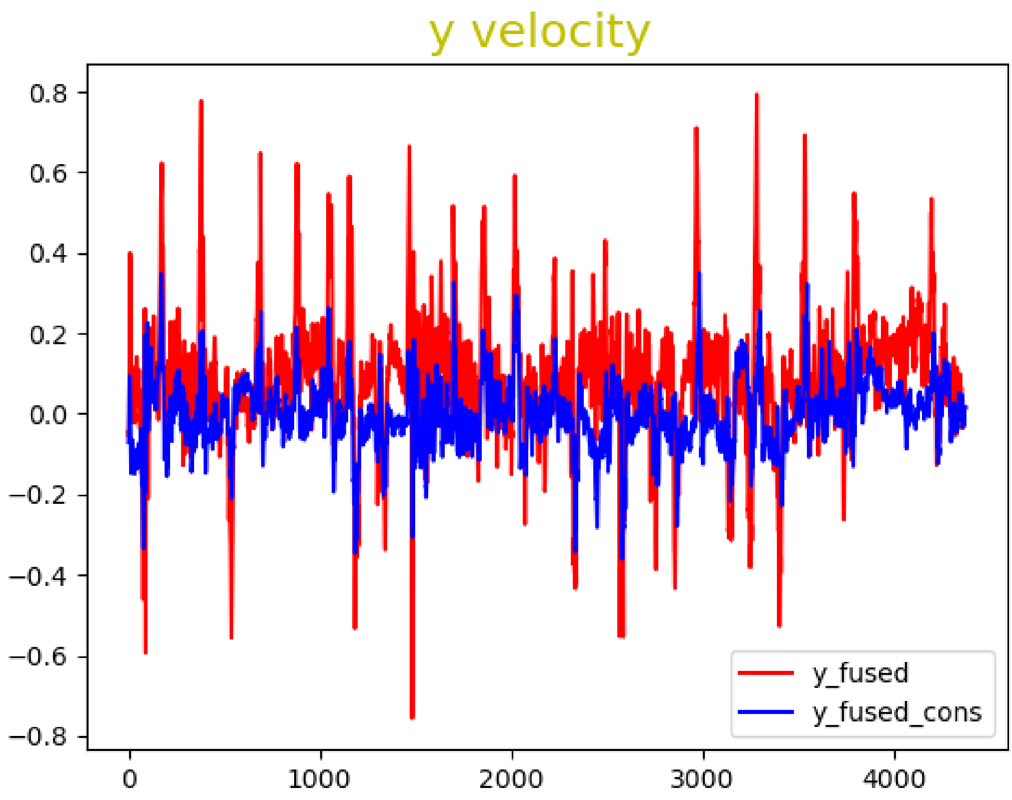


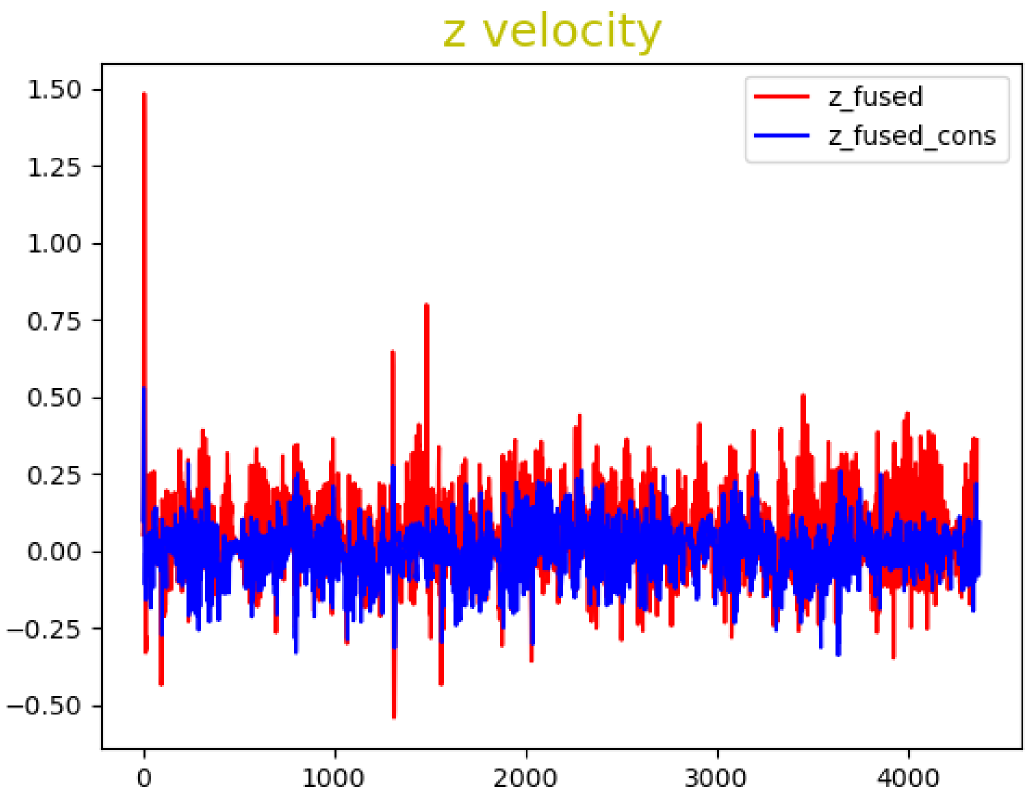




与加了运动约束的模型进行比较：


x velocity
x_fused
x_fused_cons
12
10
8
6
4
2
0
0
1000
2000
3000
4000


y velocity
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
y_fused
y_fused_cons
0
1000
2000
3000
4000



结论

从evo指标来看，添加运动约束和速度观测的精度反而略微差了一些；但是从matplot可视化数据来看，Y轴和Z轴的数据波动小了很多。

二.仿真编码器融合

代码

```
void ErrorStateKalmanFilter::CorrectErrorEstimationPosiVel(      //  position +
velocity
    const Eigen::Matrix4d &T_nb, const Eigen::Vector3d &v_b, const
Eigen::Vector3d &w_b,
    Eigen::VectorXd &Y, Eigen::MatrixXd &G, Eigen::MatrixXd &K
) {
    //
    // TODO: set measurement:        计算观测 delta pos 、 delta  velocity
    //
    Eigen::Vector3d  v_b_  =  {v_b[0],  0,  0};          //  measurment
velocity  (body  系) ,  伪观测 (vy 、vz  = 0)

    Eigen::Vector3d  dp  =  pose_.block<3,  1>(0,  3)  -   T_nb.block<3,  1>(0,
 3);
    Eigen::Vector3d  dv  =   pose_.block<3,  3>(0, 0).transpose() *vel_  -  v_b
;                //  delta v  ,  v_x 来自轮速里程计
    // TODO: set measurement equation:
    YPosiVel_.block<3, 1>(0, 0)  =  dp;          //    delta  position
    YPosiVel_.block<3, 1>(3, 0)  =  dv;           //   delta   velocity
    Y = YPosiVel_;
    //   set measurement  G
    GPosiVel_.setZero();
```

```
    GPosiVel_.block<3, 3>(0, kIndexErrorPos)  =  Eigen::Matrix3d::Identity();
    GPosiVel_.block<3, 3>(3, kIndexErrorVel)   =   pose_.block<3,  3>(0,
0).transpose();
    GPosiVel_.block<3, 3>(3, kIndexErrorOri)   =   Sophus::SO3d::hat(
pose_.block<3,  3>(0, 0).transpose() *vel_  ) ;
    G  =   GPosiVel_;
    //   set measurement  C
    CPosiVel_.setIdentity();
    Eigen::MatrixXd  C  =   CPosiVel_;
    // TODO: set Kalman gain:
    Eigen::MatrixXd R = RPosiVel_;    //  观测噪声
    K =  P_  *  G.transpose() * ( G  *  P_  *  G.transpose( )  +  C * R*
 C.transpose() ).inverse() ;
}
```

参数

config/filtering/gnss_ins_sim_filtering.yaml

```
# select fusion method for IMU-GNSS-Odo-Mag, available methods are:
#     1. error_state_kalman_filter
fusion_method: error_state_kalman_filter
# select fusion strategy for IMU-GNSS-Odo-Mag, available methods are:
#     1. position_velocity
fusion_strategy: position_velocity

## 1. Error-State Kalman filter for IMU-GNSS-Odo fusion:
error_state_kalman_filter:
    earth:
        # gravity can be calculated from https://www.sensorsone.com/local-
gravity-calculator/ using latitude and height:
        gravity_magnitude: -9.794216
        # 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-4
            accel: 2.5e-3
            bias_accel: 2.5e-3
            bias_gyro: 1.0e-4
        measurement:
            pose:
                pos: 1.0e-4
                ori: 1.0e-4
            pos: 2.5e-1 # 1.0-4
            vel: 2.5e-3
    motion_constraint:
        activated: true
        w_b_thresh: 0.13
```

运行

代码运行命令：

```
roslaunch lidar_localization gnss_ins_sim_localization.launch
```

播放数据集命令：

```
rosbag play virtual_proving_ground.bag
```

保存里程计：

```
rosservice call /save_odometry
```

数据位于:

```
src/lidar_localization/slam_data/trajectory
```

evo工具运行命令：

```
# a. fused 没有输入运动模型  输出评估结果，并以zip的格式存储：
evo_ape kitti ground_truth.txt fused.txt -r full --plot --plot_mode xyz  --
save_results ./fused.zip
# b. fused_vel 速度观测  输出评估结果，并以zip的格式存储：
evo_ape kitti ground_truth.txt gnss.txt -r full --plot --plot_mode xyz  --
save_results ./gnss.zip
#e. 比较 laser  fused  一并比较评估
evo_res  *.zip --use_filenames -p
```

注：

两个不同的模型需要修改如下：

FILE：lidar_localization/src/filtering/gnss_ins_sim_filtering.cpp

FUNCTION: GNSSINSSimFiltering::InitFusion

```
 CONFIG.FUSION_STRATEGY_ID["position_velocity"] =
KalmanFilter::MeasurementType::POSI_VEL;
```

两个参数如下：

KalmanFilter::MeasurementType::POSE　没有输入运动模型

KalmanFilter::MeasurementType::POSE_VEL　GNSS观测

EVO评估

没有输入运动模型：

```
   max   14.284080
  mean   2.663262
median   2.147246
   min   0.000118
  rmse   3.518631
   sse   19301.608785
   std   2.299522
```

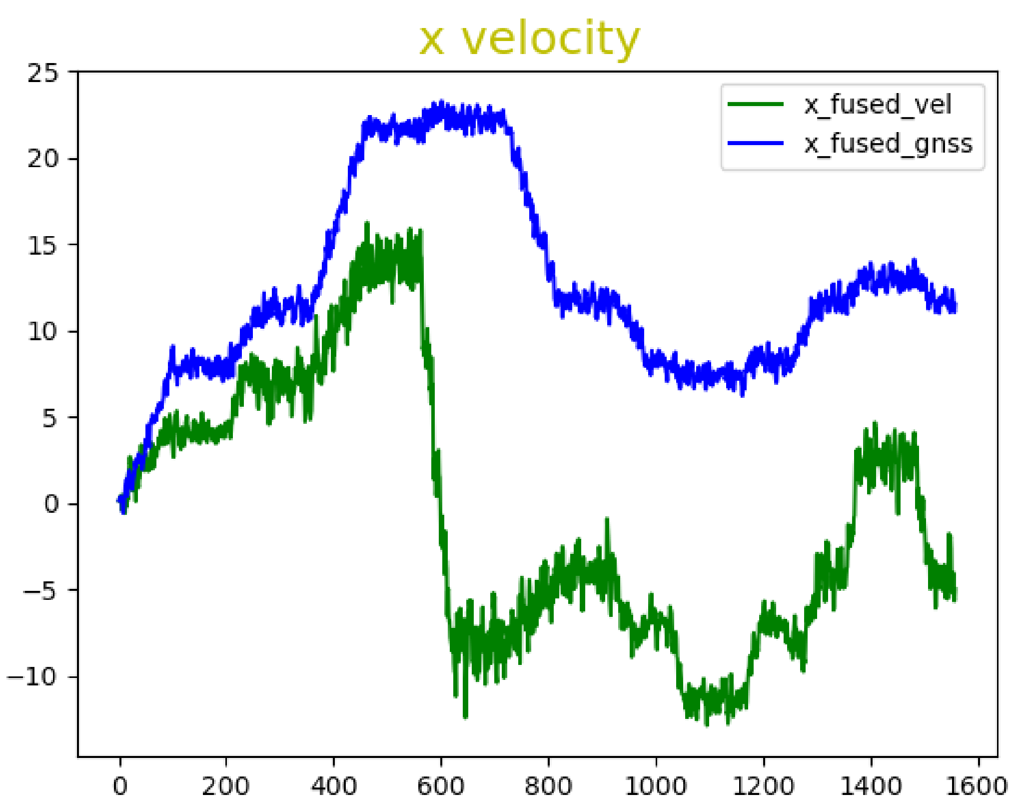
GNSS观测

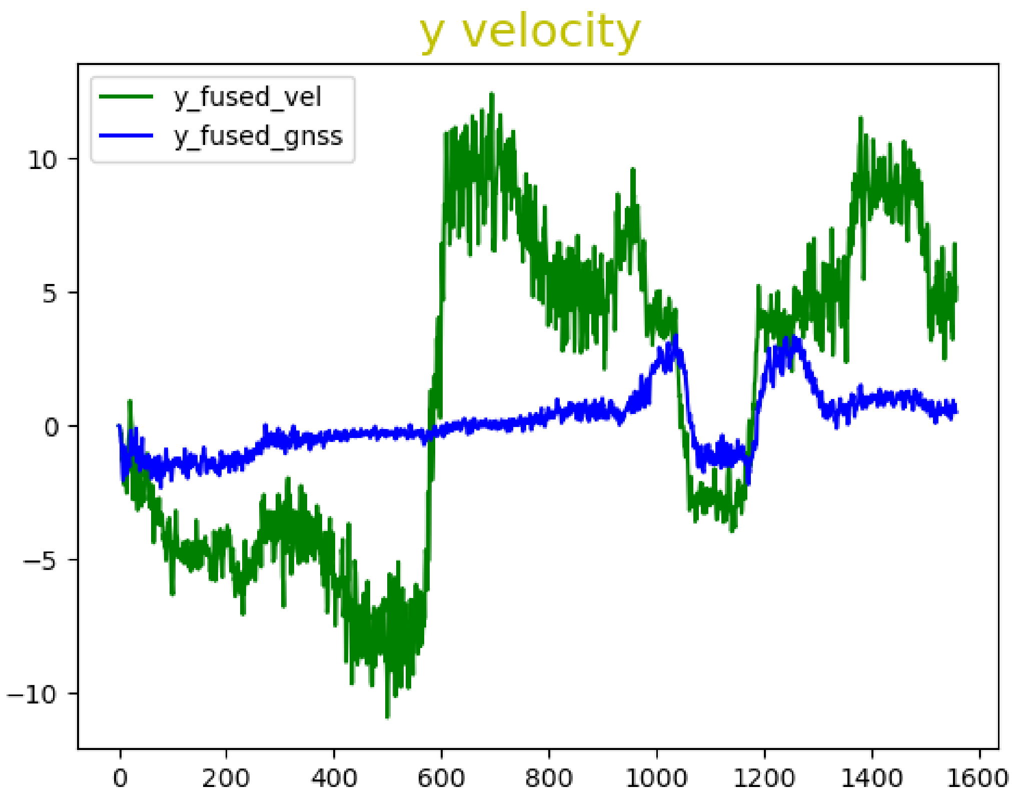
```
   max   14.422205
  mean   2.929031
median   2.366558
   min   0.000000
  rmse   3.992459
   sse   24850.039145
   std   2.713026
```

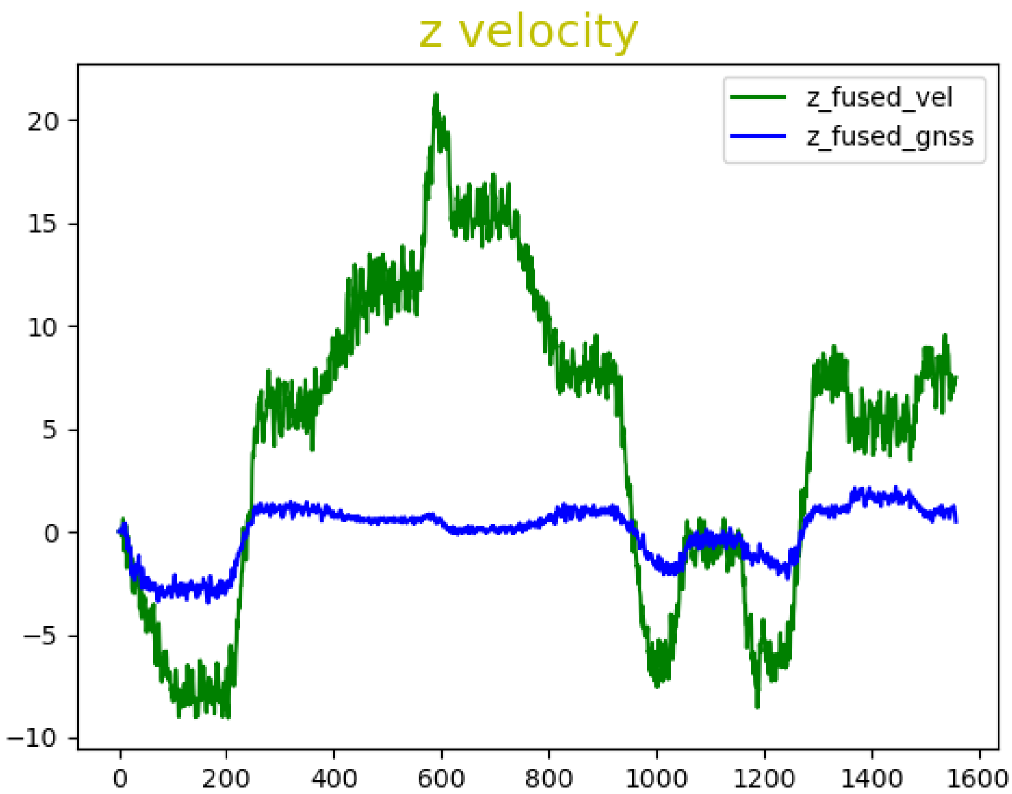
对比:

```
                rmse       mean     median        std        min         max  \
fused.zip  3.518631  2.663262  2.147246  2.299522  0.000118    14.28408
gnss.zip   3.992459  2.929031  2.366558  2.713026        0.0  14.422205

                    sse
fused.zip  19301.608785
gnss.zip   24850.039145
```

matplotlib 可视化数据







结论

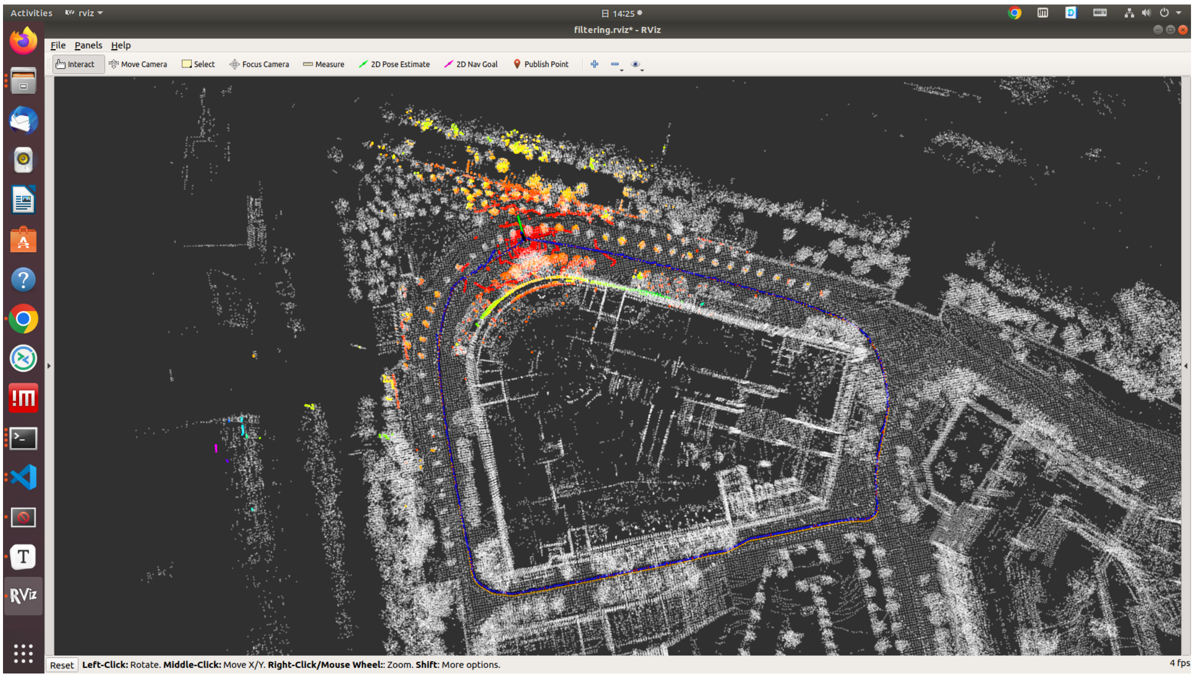
能够成功减少Y轴和Z轴的数据波动。

三.实车部署

实车硬件如下：

1. 松灵Scout2，车速为1.5m/s
2. 速腾16线雷达
3. SBG-ellipse-N 九轴惯导+单天线RTK

rviz效果



evo评估

没有输入运动模型

```
       max	1.094957
      mean	0.610520
    median	0.583310
       min	0.013621
      rmse	0.666098
       sse	297.270148
       std	0.266369
```

速度观测

```
       max	1.236915
      mean	0.655296
    median	0.671979
       min	0.043410
      rmse	0.709470
       sse	326.169547
       std	0.271911
```

运动约束模型

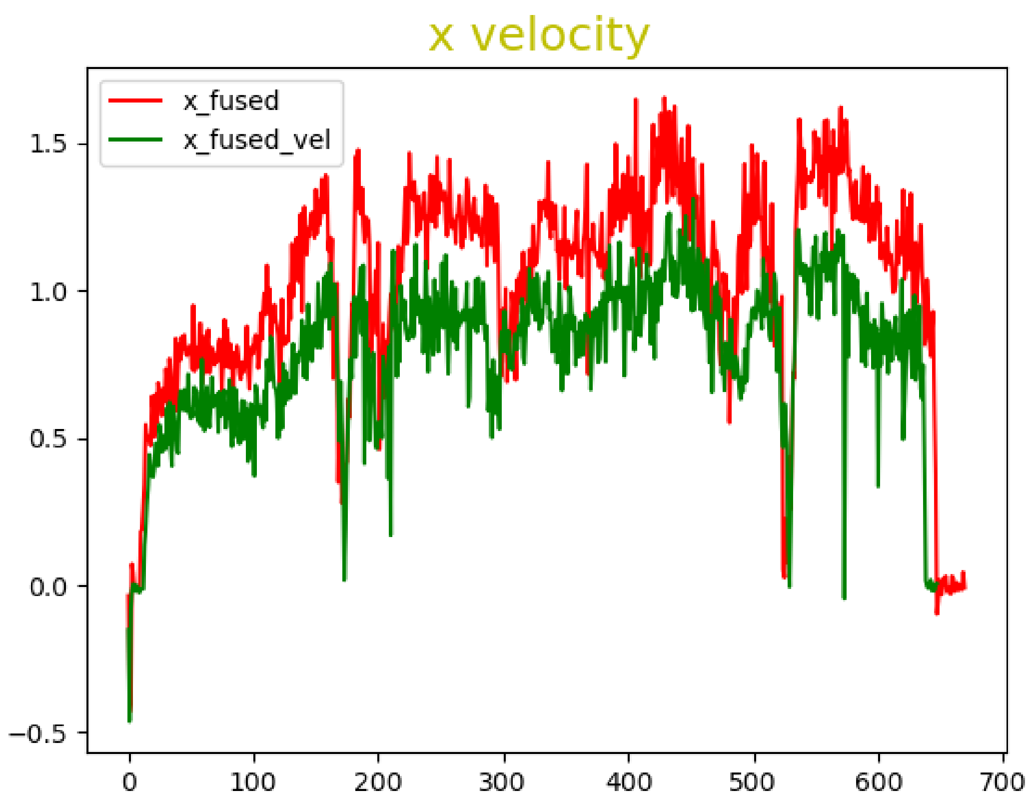
```
      max   2.325925
     mean   2.057419
   median   2.091060
      min   0.120037
     rmse   2.075690
      sse   2766.049187
      std   0.274796
```

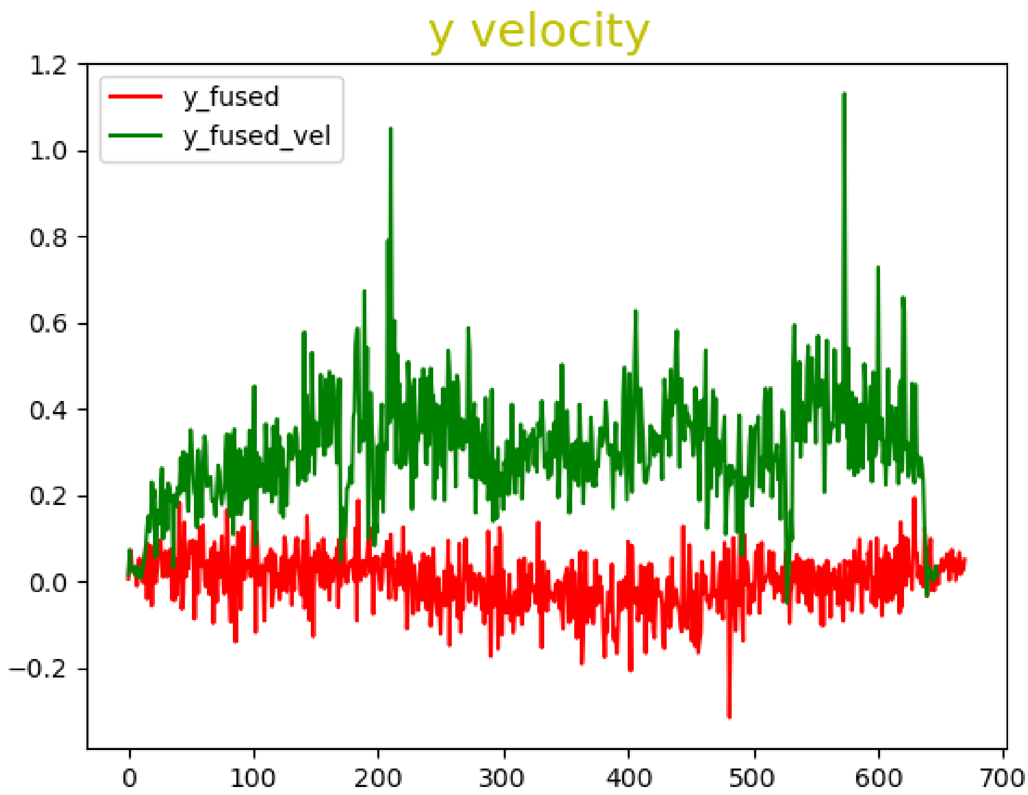
对比:

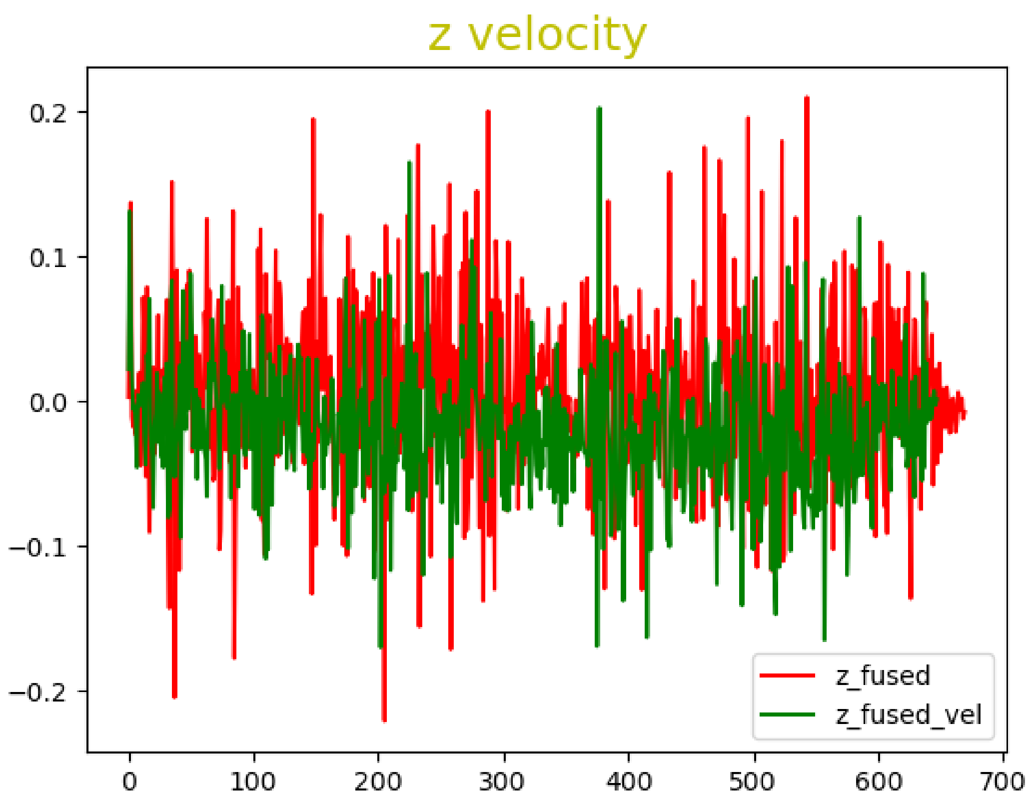
```
                   rmse      mean    median       std       min       max  \
fused_cons.zip  2.07569  2.057419   2.09106  0.274796  0.120037  2.325925
fused_vel.zip   0.70947  0.655296  0.671979  0.271911   0.04341  1.236915
fused.zip      0.666098   0.61052   0.58331  0.266369  0.013621  1.094957

                      sse
fused_cons.zip  2766.049187
fused_vel.zip    326.169547
fused.zip        297.270148
```

matplotlib 可视化数据

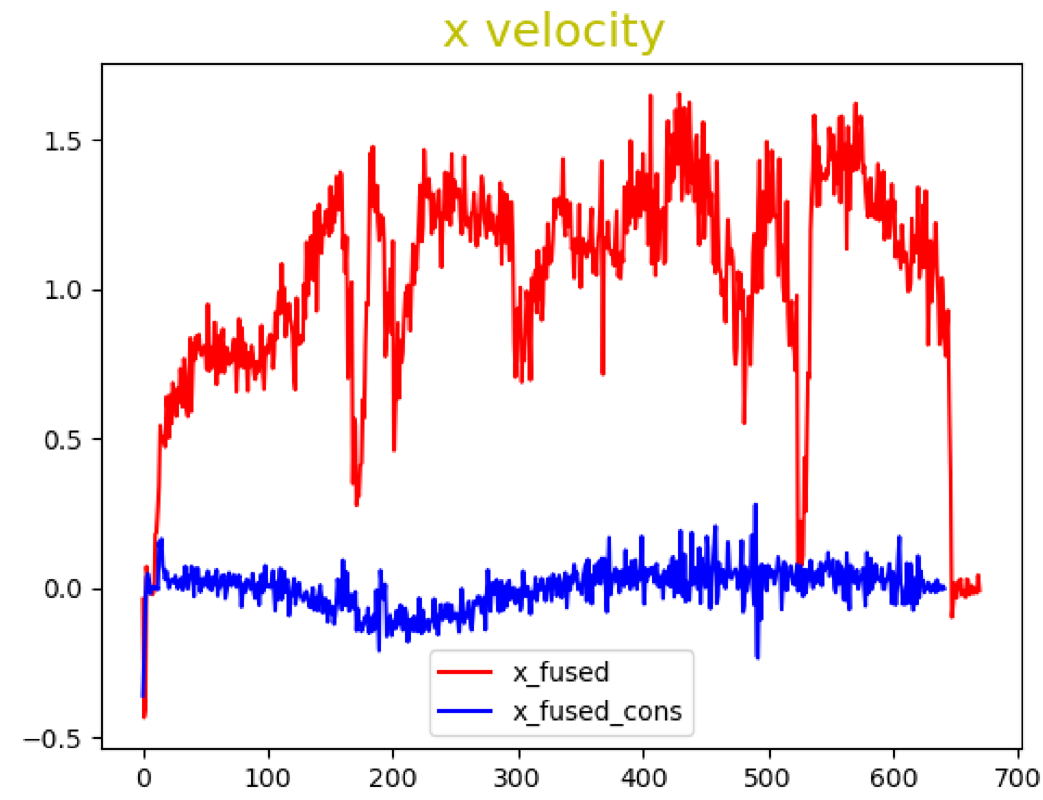
与加了速度观测的模型进行比较:

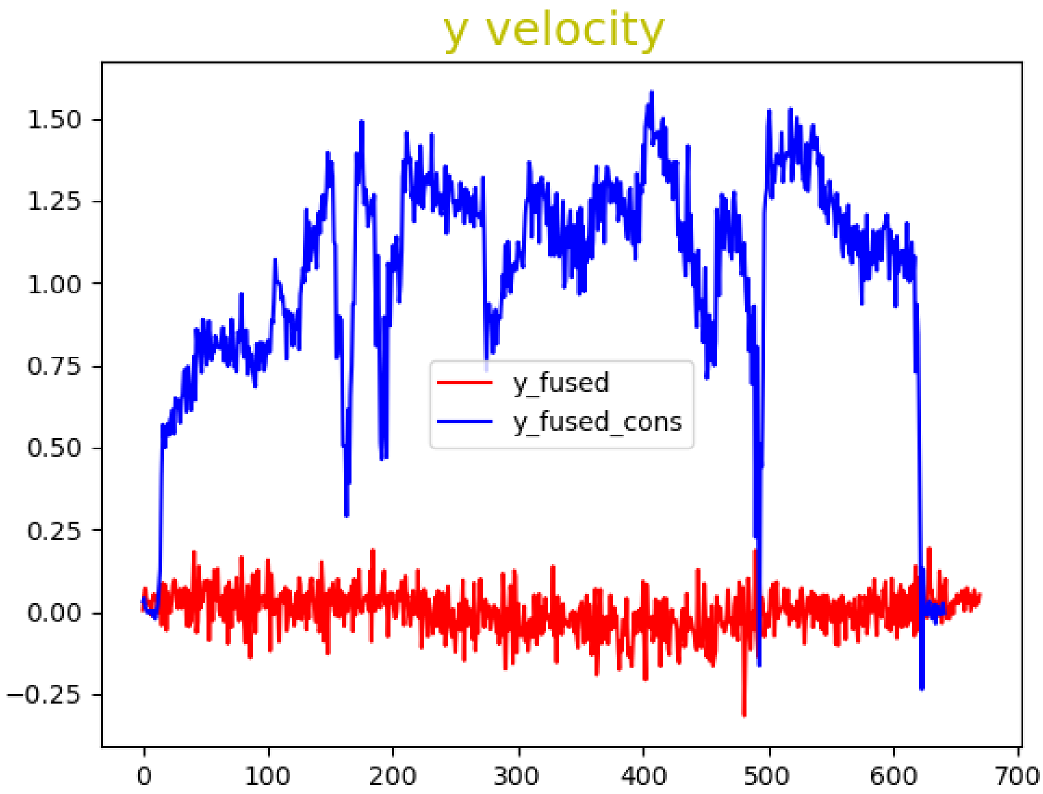


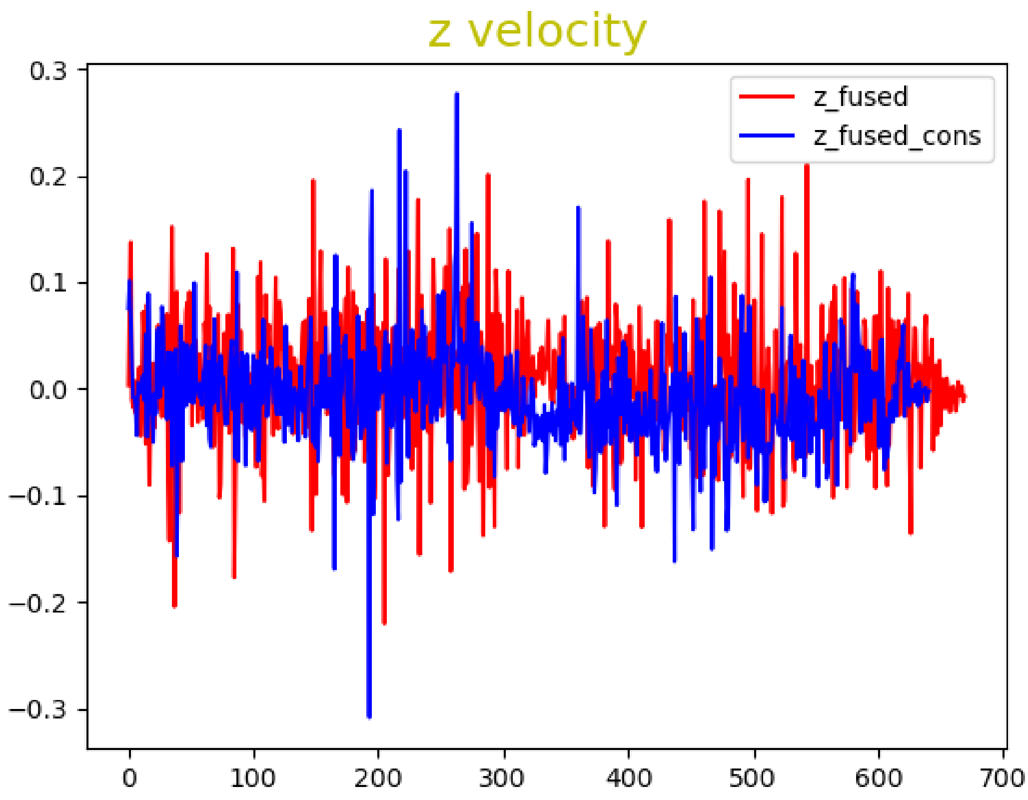




与加了运动约束的模型进行比较：


x velocity
1.5
1.0
0.5
0.0
−0.5
0
100
200
300
400
500
600
700
x_fused
x_fused_cons


y velocity
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0
100
200
300
400
500
600
700
y_fused
y_fused_cons



总结和思考

总结

添加速度观测、GNSS观测或运动约束能够提高融合系统的性能。

思考和疑问

1. 对于kitti数据集，添加运动约束对于X轴的数据波动没有很多提升，原因是小车是朝X轴运动的，并没有约束x轴。但添加速度观测也同样对于X轴的数据波动没有很大提升，原因是x轴是小车的运行方向，本身波动就会比较大?
2. 同样对于仿真数据集，添加GNSS里程观测，对于X轴的数据波动没有提升，反而加大了，原因是参数没调好?
3. 对于自采数据集，y轴的数据波动没有提升，反而加大了。这可能是因为小车运动方向是Y轴，还有参数没调好的原因。