第6章作业

一. 课程给定数据

中值法

补全代码

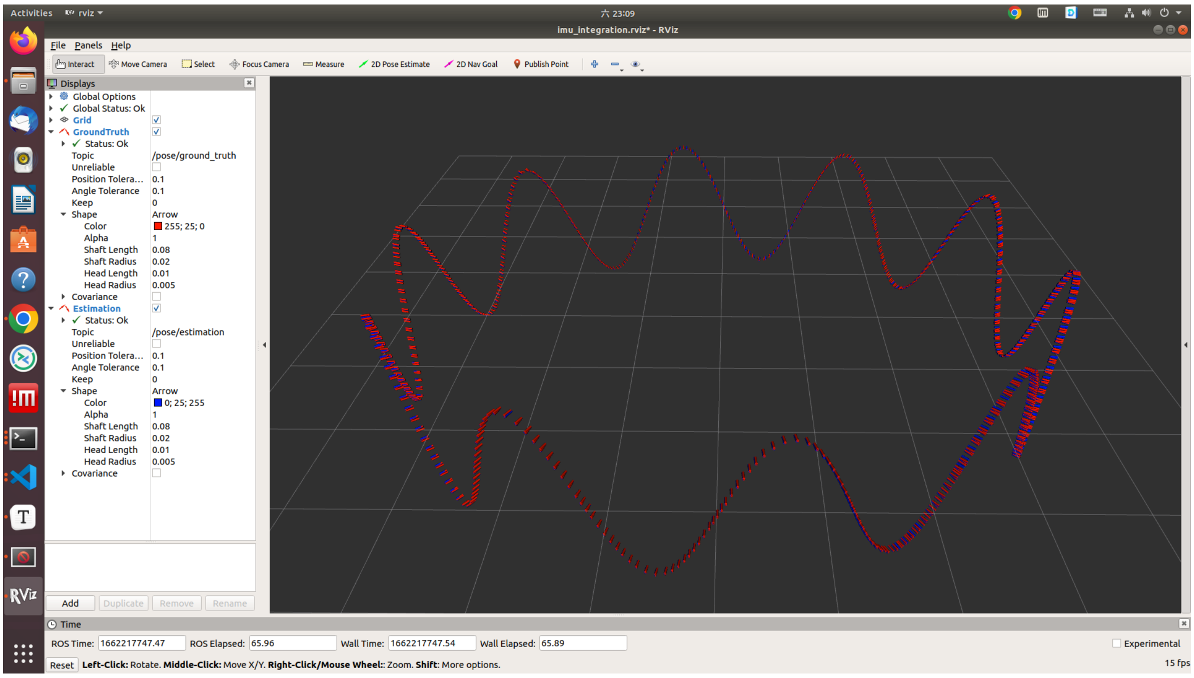
```
bool Activity::UpdatePose(void) {
    if (!initialized_) {
        // use the latest measurement for initialization:
        OdomData &odom_data = odom_data_buff_.back();
        IMUData imu_data = imu_data_buff_.back();

        pose_ = odom_data.pose;
        vel_ = odom_data.vel;

        initialized_ = true;

        odom_data_buff_.clear();
        imu_data_buff_.clear();

        // keep the latest IMU measurement for mid-value integration:
        imu_data_buff_.push_back(imu_data);
    } else {
 //
        // TODO: implement your estimation here
        //
        // get deltas:
            size_t  index_curr_ = 1;
            size_t  index_prev_ =0;
            Eigen::Vector3d angular_delta = Eigen::Vector3d::Zero();
            if(! (GetAngularDelta(index_curr_, index_prev_, angular_delta)) ){
            std::cout << "GetAngularDelta(): index error" << std::endl;
              //     获取等效旋转矢量
        }
        // update orientation:
            Eigen::Matrix3d  R_curr_  =  Eigen::Matrix3d::Identity();
            Eigen::Matrix3d  R_prev_ =  Eigen::Matrix3d::Identity();
            UpdateOrientation(angular_delta,R_curr_, R_prev_);
        //    更新四元数
        // get velocity delta:
            double  delta_t_;
            Eigen::Vector3d  velocity_delta_;
            if(! (GetVelocityDelta(index_curr_, index_prev_, R_curr_, R_prev_,
delta_t_, velocity_delta_)) ){
                std::cout << "GetVelocityDelta(): index error" << std::endl;
                  //     获取速度差值
            }
        // update position:
            UpdatePosition(delta_t_, velocity_delta_);

        // move forward --
```

```
        // NOTE: this is NOT fixed. you should update your buffer according to the method of your choice:
        imu_data_buff_.pop_front();

    }

    return true;
}
```

运行

```
roslaunch imu_integration imu_integration.launch
```

效果



欧拉法

用宏定义：

```
#define  MedianMethod    //  use  euler  method    or  median method
```

#define MedianMethod 则为中值法，注释掉则为欧拉法。

补全代码

计算VelocityDelta

```
bool Activity::GetVelocityDelta(
    const size_t index_curr, const size_t index_prev,
    const Eigen::Matrix3d &R_curr, const Eigen::Matrix3d &R_prev,
    double &delta_t, Eigen::Vector3d &velocity_delta
) {
    //
    // TODO: this could be a helper routine for your own implementation
    //
    if (
```

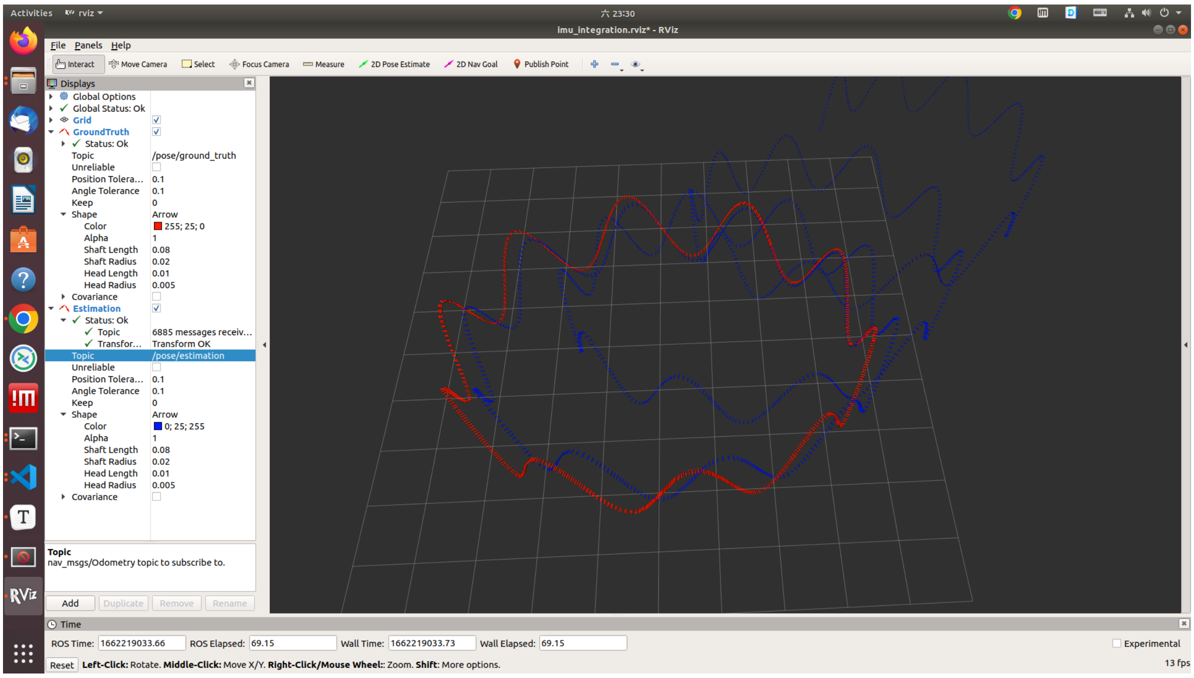
```
        index_curr <= index_prev ||
        imu_data_buff_.size() <= index_curr
    ) {
        return false;
    }

    const IMUData &imu_data_curr = imu_data_buff_.at(index_curr);
    const IMUData &imu_data_prev = imu_data_buff_.at(index_prev);

    delta_t = imu_data_curr.time - imu_data_prev.time;

    Eigen::Vector3d linear_acc_curr =
GetUnbiasedLinearAcc(imu_data_curr.linear_acceleration, R_curr);
    Eigen::Vector3d linear_acc_prev =
GetUnbiasedLinearAcc(imu_data_prev.linear_acceleration, R_prev);

    velocity_delta = 0.5*delta_t*(linear_acc_curr + linear_acc_prev);
#ifdef   MedianMethod
    velocity_delta = 0.5*delta_t*(linear_acc_curr + linear_acc_prev);
#else
    velocity_delta =  delta_t*linear_acc_prev;
#endif
    return true;
}
```

计算angular：

```
bool Activity::GetAngularDelta(
    const size_t index_curr, const size_t index_prev,
    Eigen::Vector3d &angular_delta
) {
    //
    // TODO: this could be a helper routine for your own implementation
    //
    if (
        index_curr <= index_prev ||
        imu_data_buff_.size() <= index_curr
    ) {
        return false;
    }

    const IMUData &imu_data_curr = imu_data_buff_.at(index_curr);
    const IMUData &imu_data_prev = imu_data_buff_.at(index_prev);

    double delta_t = imu_data_curr.time - imu_data_prev.time;

    Eigen::Vector3d angular_vel_curr =
GetUnbiasedAngularVel(imu_data_curr.angular_velocity);
    //    omega_k
    Eigen::Vector3d angular_vel_prev =
GetUnbiasedAngularVel(imu_data_prev.angular_velocity);
    //    omega_k-1
#ifdef   MedianMethod
    angular_delta = 0.5*delta_t*(angular_vel_curr + angular_vel_prev);
           //   中值法计算angular
    //  中值法计算angular
#else
```

```
    angular_delta = delta_t*angular_vel_prev;
    //  欧拉法计算angular
                                                                    //   欧拉法
#endif
    return true;
}
```

效果



对比

从效果可以明显看出，在使用课程数据时，中值法精度优于欧拉法。

二.IMU仿真

使用方法

定义误差模型

可以选择 'low-accuracy', 'mid-accuracy' and 'high accuracy' 三种不同精度的IMU模型，或自定义IMU模型

```
imu = imu_model.IMU(accuracy=imu_err, axis=6, gps=False)
imu = imu_model.IMU(accuracy='low accuracy', axis=9, gps=True)
imu_err = {
            # gyro bias, deg/hr
            'gyro_b': np.array([0.0, 0.0, 0.0]),
            # gyro angle random walk, deg/rt-hr
            'gyro_arw': np.array([0.25, 0.25, 0.25]),
            # gyro bias instability, deg/hr
            'gyro_b_stability': np.array([3.5, 3.5, 3.5]),
            # gyro bias instability correlation, sec.
            # set this to 'inf' to use a random walk model
            # set this to a positive real number to use a first-order Gauss-
Markkov model
            'gyro_b_corr': np.array([100.0, 100.0, 100.0]),
            # accelerometer bias, m/s^2
```

```
            'accel_b': np.array([0.0e-3, 0.0e-3, 0.0e-3]),
            # accelerometer velocity random walk, m/s/rt-hr
            'accel_vrw': np.array([0.03119, 0.03009, 0.04779]),
            # accelerometer bias instability, m/s^2
            'accel_b_stability': np.array([4.29e-5, 5.72e-5, 8.02e-5]),
            # accelerometer bias instability correlation, sec. Similar to
gyro_b_corr
            'accel_b_corr': np.array([200.0, 200.0, 200.0]),
            # magnetometer noise std, uT
            'mag_std': np.array([0.2, 0.2, 0.2])
          }
```

运动定义 command type

通过写入到csv中，进行运动定义，主要使用到两种指令格式， command type 1 和 command type 2

command type 1 定义在command duration 时间内的速率和角速率变化，可用于加速，匀速运动

command type 2 定义在command duration 时间内达到预设的角度(绝对) 和 速度

生成数据集

参考recorder_node_allan_variance_analysis.py 和GitHub上的写法，仿写生成dataset的代码 ,gnss-ins-sim 源码保存数据集的方式是csv，这里为了方便可视化，转为rosbag的方式保存，保存仿真的数据有:

imu : gyro accel ;

groundtruth : orientation(四元数)、position 、velocity。

代码

FILE: src/gnss_ins_sim/src/recorder_node_sim.py

```
#!/usr/bin/python

import os

import rospkg
import rospy
import rosbag

import math
import numpy as np
import pandas as pd

from gnss_ins_sim.sim import imu_model
from gnss_ins_sim.sim import ins_sim

# from gnss_ins_sim.geoparams import  geoparams
from std_msgs import msg

from std_msgs.msg import String
from sensor_msgs.msg import Imu
from nav_msgs.msg import Odometry

def get_gnss_ins_sim(motion_def_file, fs_imu, fs_gps):
```

```python
# set  origin  x y z
origin_x =  2849886.61825
origin_y =  -4656214.27294
origin_z =  -3287190.60046
'''
Generate simulated GNSS/IMU data using specified trajectory.
'''
# set IMU model:
D2R = math.pi/180.0
# imu_err = 'low-accuracy'
imu_err = {
    # 1. gyro:
    # a. random noise:
    # gyro angle random walk, deg/rt-hr
    'gyro_arw': np.array([0., 0., 0.]),
    # gyro bias instability, deg/hr
    'gyro_b_stability': np.array([0.0, 0.0, 0.0]),
    # gyro bias isntability correlation time, sec
    'gyro_b_corr': np.array([100.0, 100.0, 100.0]),
    # b. deterministic error:
    'gyro_b': np.array([0.0, 0.0, 0.0]),
    'gyro_k': np.array([1.0, 1.0, 1.0]),
    'gyro_s': np.array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0]),
    # 2. accel:
    # a. random noise:
    # accel velocity random walk, m/s/rt-hr
    'accel_vrw': np.array([0., 0., 0.]),
    # accel bias instability, m/s2
    'accel_b_stability': np.array([0., 0., 0.]),
    # accel bias isntability correlation time, sec
    'accel_b_corr': np.array([100.0, 100.0, 100.0]),
    # b. deterministic error:
    'accel_b': np.array([0.0, 0.0, 0.0]),
    'accel_k': np.array([1.0, 1.0, 1.0]),
    'accel_s': np.array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0]),
    # 3. mag:
    'mag_si': np.eye(3) + np.random.randn(3, 3)*0.0,
    'mag_hi': np.array([10.0, 10.0, 10.0])*0.0,
    'mag_std': np.array([0.1, 0.1, 0.1])
}
# generate GPS and magnetometer data:
imu = imu_model.IMU(accuracy=imu_err, axis=9, gps=True)

# init simulation:
sim = ins_sim.Sim(
    # here sync GPS with other measurements as marker:
    [fs_imu, fs_imu, fs_imu],
    motion_def_file,
    ref_frame=1,
    imu=imu,
    mode=None,
    env=None,
    algorithm=None
)

# run:
sim.run(1)
```

```
    # get simulated data:
    rospy.logwarn(
        'Simulated data size: Gyro-{}, Accel-{}, pos-{}'.format(
            len(sim.dmgr.get_data_all('gyro').data[0]),
            len(sim.dmgr.get_data_all('accel').data[0]),
            len(sim.dmgr.get_data_all('ref_pos').data)
        )
    )

    # calibration stages:
    step_size = 1.0 / fs_imu

    for i, (gyro, accel, ref_q, ref_pos, ref_vel) in enumerate(
        zip(
            # a. gyro:
            sim.dmgr.get_data_all('gyro').data[0],
            # b. accel:
            sim.dmgr.get_data_all('accel').data[0],
            # c. gt_pose:
            sim.dmgr.get_data_all('ref_att_quat').data,                #
groundtruth
            sim.dmgr.get_data_all('ref_pos').data,
            # d. true_vel :
            sim.dmgr.get_data_all('ref_vel').data
        )
    ):

        yield {
            'stamp': i * step_size,
             'data': {
                    # a. gyro:
                    'gyro_x': gyro[0],
                    'gyro_y': gyro[1],
                    'gyro_z': gyro[2],
                    # b. accel:
                    'accel_x': accel[0],
                    'accel_y': accel[1],
                    'accel_z': accel[2],
                    # c. true orientation:
                    'gt_quat_w': ref_q[0],
                    'gt_quat_x':  ref_q[1],
                    'gt_quat_y':  ref_q[2],
                    'gt_quat_z':  ref_q[3],
                    # d. true position:
                    'gt_pos_x': ref_pos[0]  + origin_x,
                    'gt_pos_y': ref_pos[1]  + origin_y,
                    'gt_pos_z': ref_pos[2]  + origin_z,
                    # d. true velocity:
                    'gt_vel_x': ref_vel[0],
                    'gt_vel_y': ref_vel[1],
                    'gt_vel_z': ref_vel[2]
             }
         }
    sim.results()
    sim.plot(['ref_pos', 'ref_vel'], opt={'ref_pos': '3d'})


def gnss_ins_sim_recorder():
```

```
    """
    Record simulated GNSS/IMU data as ROS bag
    """
    # ensure gnss_ins_sim_node is unique:
    rospy.init_node('gnss_ins_sim_recorder_node')

    # parse params:
    motion_def_name = rospy.get_param('/gnss_ins_sim_recorder_node/motion_file')
    sample_freq_imu =
rospy.get_param('/gnss_ins_sim_recorder_node/sample_frequency/imu')
    sample_freq_gps =
rospy.get_param('/gnss_ins_sim_recorder_node/sample_frequency/gps')
    topic_name_imu =
rospy.get_param('/gnss_ins_sim_recorder_node/topic_name_imu')
    topic_name_gt = rospy.get_param('/gnss_ins_sim_recorder_node/topic_name_gt')

    ## save scv
    output_path = rospy.get_param('/gnss_ins_sim_recorder_node/output_path')
    output_name = rospy.get_param('/gnss_ins_sim_recorder_node/output_name')
    ## save rosbag
    rosbag_output_path =
rospy.get_param('/gnss_ins_sim_recorder_node/output_path')
    rosbag_output_name =
rospy.get_param('/gnss_ins_sim_recorder_node/output_name')

    # generate simulated data:
    motion_def_path = os.path.join(
        rospkg.RosPack().get_path('gnss_ins_sim'), 'config', 'motion_def',
motion_def_name
    )
    imu_simulator = get_gnss_ins_sim(
        # motion def file:
        motion_def_path,
        # gyro-accel/gyro-accel-mag sample rate:
        sample_freq_imu,
        # GPS sample rate:
        sample_freq_gps
    )

    # write as csv:
    # data = pd.DataFrame(
    #     list(imu_simulator)
    # )
    # data.to_csv(
    #     os.path.join(output_path, output_name)
    # )

    #write  rosbag
    with rosbag.Bag(
        os.path.join(rosbag_output_path, rosbag_output_name), 'w'
    ) as bag:
        # get timestamp base:
        timestamp_start = rospy.Time.now()

        for measurement in imu_simulator:
            # init:
            msg_imu = Imu()
            # a. set header:
```

```python
            msg_imu.header.frame_id = 'inertial'
            msg_imu.header.stamp = timestamp_start +
rospy.Duration.from_sec(measurement['stamp'])
            # b. set orientation estimation:
            msg_imu.orientation.x = 0.0
            msg_imu.orientation.y = 0.0
            msg_imu.orientation.z = 0.0
            msg_imu.orientation.w = 1.0
            # c. gyro:
            msg_imu.angular_velocity.x = measurement['data']['gyro_x']
            msg_imu.angular_velocity.y = measurement['data']['gyro_y']
            msg_imu.angular_velocity.z = measurement['data']['gyro_z']
            msg_imu.linear_acceleration.x = measurement['data']['accel_x']
            msg_imu.linear_acceleration.y = measurement['data']['accel_y']
            msg_imu.linear_acceleration.z = measurement['data']['accel_z']

            # write:
            bag.write(topic_name_imu, msg_imu, msg_imu.header.stamp)

            # write:
            bag.write(topic_name_imu, msg_imu, msg_imu.header.stamp)

            # init :  groundtruth
            msg_odom = Odometry()
            # a.set header:
            msg_odom.header.frame_id = 'inertial'
            msg_odom.header.stamp =    msg_imu.header.stamp
            # b.set gt_pose
            msg_odom.pose.pose.position.x =  measurement['data']['gt_pos_x']
            msg_odom.pose.pose.position.y =  measurement['data']['gt_pos_y']
            msg_odom.pose.pose.position.z =  measurement['data']['gt_pos_z']

            msg_odom.pose.pose.orientation.w = measurement['data']['gt_quat_w']
            msg_odom.pose.pose.orientation.x = measurement['data']['gt_quat_x']
            msg_odom.pose.pose.orientation.y = measurement['data']['gt_quat_y']
            msg_odom.pose.pose.orientation.z = measurement['data']['gt_quat_z']
            #c.set gt_vel
            msg_odom.twist.twist.linear.x = measurement['data']['gt_vel_x']
            msg_odom.twist.twist.linear.y = measurement['data']['gt_vel_y']
            msg_odom.twist.twist.linear.z = measurement['data']['gt_vel_z']

            # write
            bag.write(topic_name_gt, msg_odom, msg_odom.header.stamp)

if __name__ == '__main__':
    try:
        gnss_ins_sim_recorder()
    except rospy.ROSInterruptException:
        pass
```

自定义motion 运动状态

FILE: src/gnss_ins_sim/config/motion_def

根据 gnss-ins-sim 的command type 定义和各量纲单位，修改csv，生成对应的rosbag，配置文件在config中

FILE: src/gnss_ins_sim/config/recorder_gnss_ins_sim.yaml

```
# motion def:
motion_file: recorder_gnss_ins_sim_speedup_down.csv
# IMU params:
imu: 1
# sample frequency of simulated GNSS/IMU data:
sample_frequency:
    imu: 100.0
    gps: 10.0
# topic name:
topic_name_imu: /sim/sensor/imu
topic_name_gt:  /pose/ground_truth
# output rosbag path:
output_path:
/home/qjs/code/ROS_Localization/shenlan/06/global_localization_chapter6_ws/src/d
ata/gnss_ins_sim/recorder_gnss_ins_sim
# output name:
output_name: speedup_down.bag
```

motion1： 绕"8"字

FILE: src/gnss_ins_sim/config/motion_def/recorder_gnss_ins_sim_8circle.csv

ini lat (deg)	**ini lon (deg)**	**ini alt (m)**	**ini vx_body (m/s)**	**ini vy_body (m/s)**	**ini vz_body (m/s)**	**ini yaw (deg)**	**ini pitch (deg)**	**ini roll (deg)**
31.224361	121.46917	0	5	0	0	0	0	0
command type	yaw (deg)	pitch (deg)	roll (deg)	vx_body (m/s)	vy_body (m/s)	vz_body (m/s)	command duration (s)	GPS visibility
1	10	0	0	0	0	0	36	1
1	-10	0	0	0	0	0	36	1
1	10	0	0	0	0	0	36	1
1	-10	0	0	0	0	0	36	1

motion2： 静止

FILE: src/gnss_ins_sim/config/motion_def/recorder_gnss_ins_sim_static.csv

ini lat (deg)	**ini lon (deg)**	**ini alt (m)**	**ini vx_body (m/s)**	**ini vy_body (m/s)**	**ini vz_body (m/s)**	**ini yaw (deg)**	**ini pitch (deg)**	**ini roll (deg)**
31.224361	121.46917	0	0	0	0	0	0	0
command type	yaw (deg)	pitch (deg)	roll (deg)	vx_body (m/s)	vy_body (m/s)	vz_body (m/s)	command duration (s)	GPS visibility
1	0	0	0	0	0	0	60	1

motion3：匀速

FILE: src/gnss_ins_sim/config/motion_def/recorder_gnss_ins_sim_speedconstant.csv

ini lat (deg)	ini lon (deg)	ini alt (m)	ini vx_body (m/s)	ini vy_body (m/s)	ini vz_body (m/s)	ini yaw (deg)	ini pitch (deg)	ini roll (deg)
31.224361	121.46917	0	5	5	5	0	0	0
command type	yaw (deg)	pitch (deg)	roll (deg)	vx_body (m/s)	vy_body (m/s)	vz_body (m/s)	command duration (s)	GPS visibility
1	0	0	0	0	0	0	60	1

motion4： 加速

FILE: src/gnss_ins_sim/config/motion_def/recorder_gnss_ins_sim_speedup.csv

ini lat (deg)	ini lon (deg)	ini alt (m)	ini vx_body (m/s)	ini vy_body (m/s)	ini vz_body (m/s)	ini yaw (deg)	ini pitch (deg)	ini roll (deg)
31.224361	121.46917	0	0	0	0	0	0	0
command type	yaw (deg)	pitch (deg)	roll (deg)	vx_body (m/s)	vy_body (m/s)	vz_body (m/s)	command duration (s)	GPS visibility
1	0	0	0	1	1	1	60	1
1	0	0	0	0	2	2	60	1
1	0	0	0	0	0	1	60	1
1	0	0	0	1	1	0	60	1
1	0	0	0	1	1	1	60	1

motion5: 先加速后减速

FILE: src/gnss_ins_sim/config/motion_def/recorder_gnss_ins_sim_speedup_down.csv

ini lat (deg)	ini lon (deg)	ini alt (m)	ini vx_body (m/s)	ini vy_body (m/s)	ini vz_body (m/s)	ini yaw (deg)	ini pitch (deg)	ini roll (deg)
31.224361	121.46917	0	5	0	0	0	0	0
command type	yaw (deg)	pitch (deg)	roll (deg)	vx_body (m/s)	vy_body (m/s)	vz_body (m/s)	command duration (s)	GPS visibility
1	0	0	0	10	10	10	30	1
1	0	0	0	-2	-2	-2	60	1

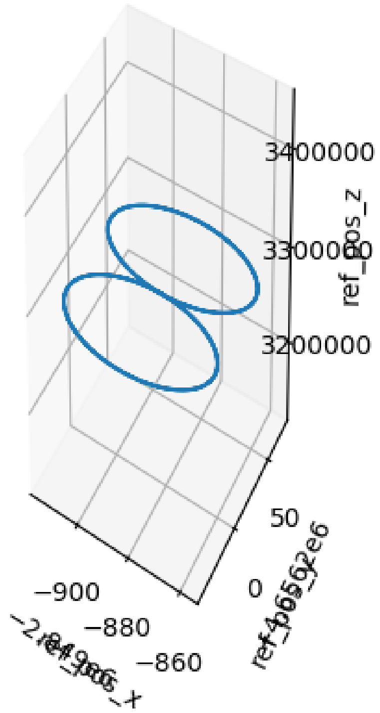
运行

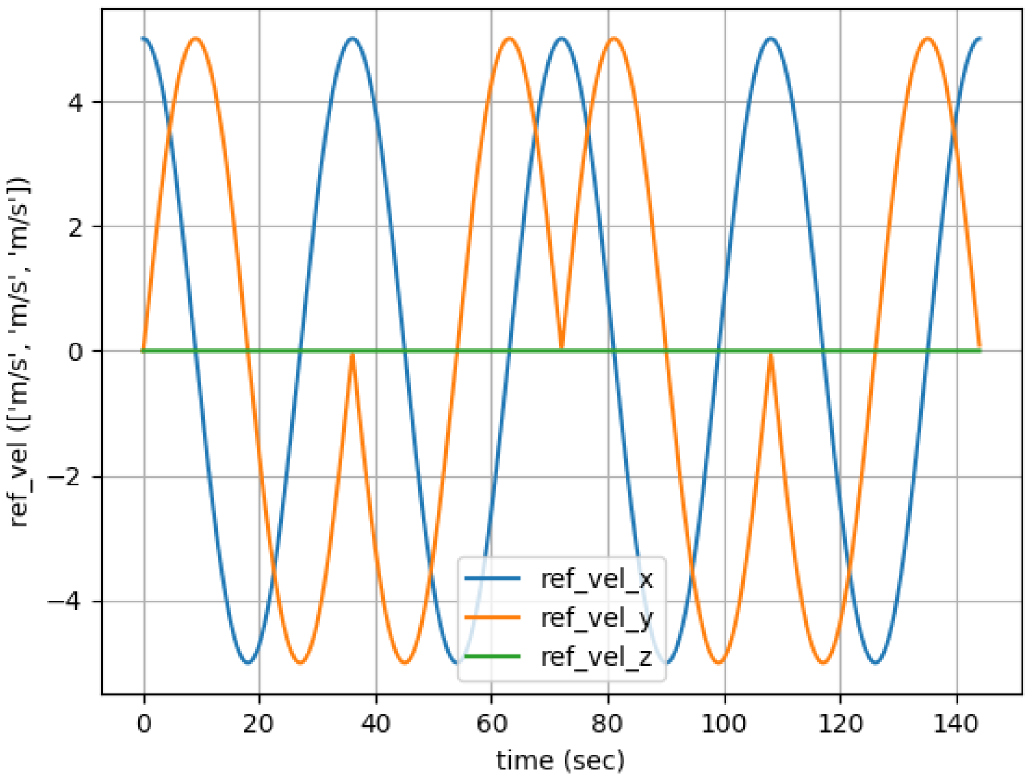
代码运行命令：

```
roslaunch gnss_ins_sim recorder_gnss_ins_sim.launch
```

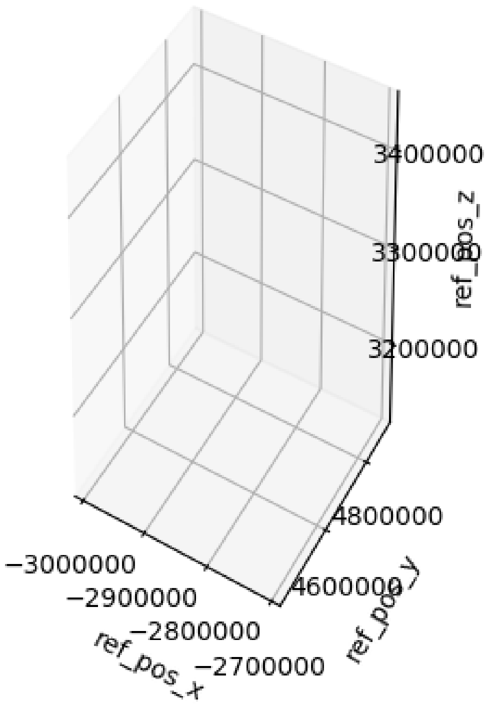
效果

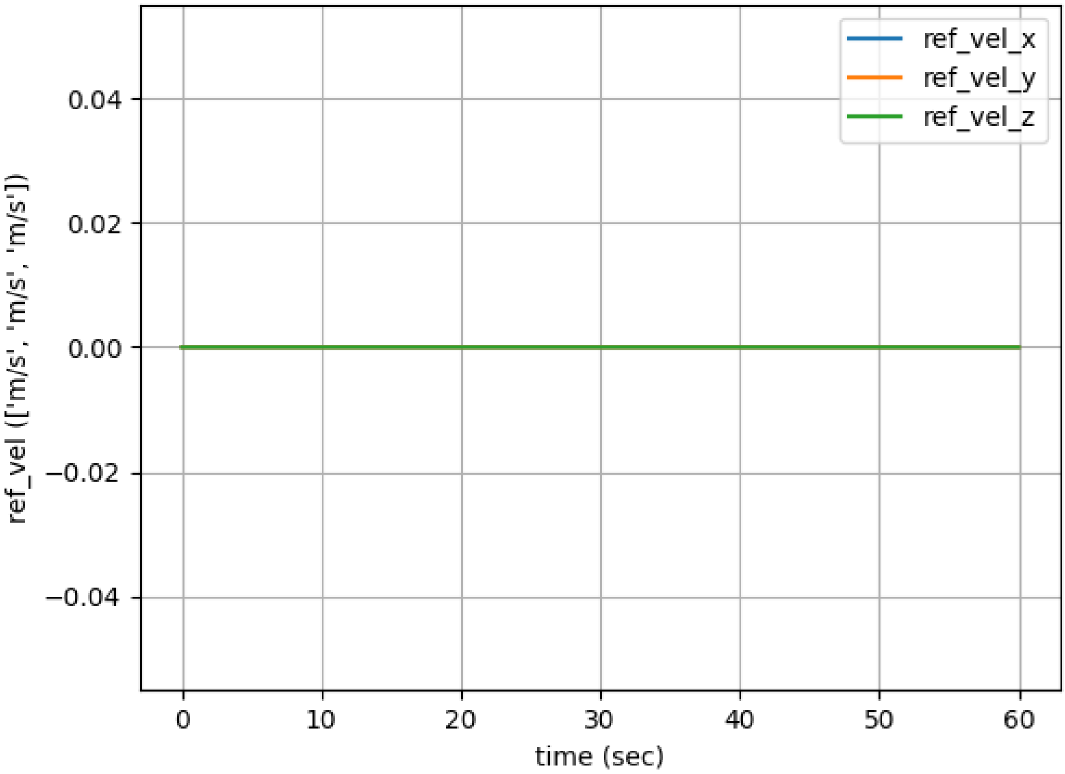
motion1： 绕“8”字


3400000
3300000
3200000
ref_pos_z
50
0
−900
−880
−860

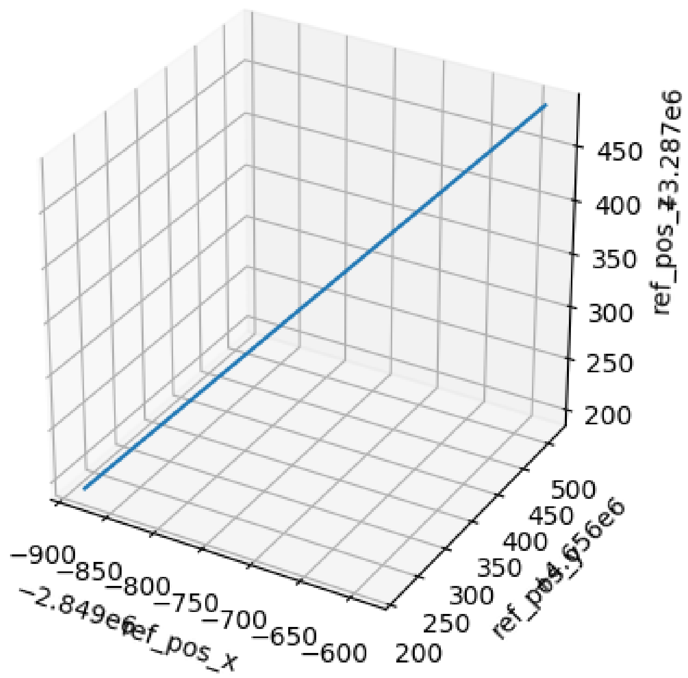

ref_vel (['m/s', 'm/s', 'm/s'])
4
2
0
−2
−4
0
20
40
60
80
100
120
140
time (sec)
ref_vel_x
ref_vel_y
ref_vel_z

motion2：静止

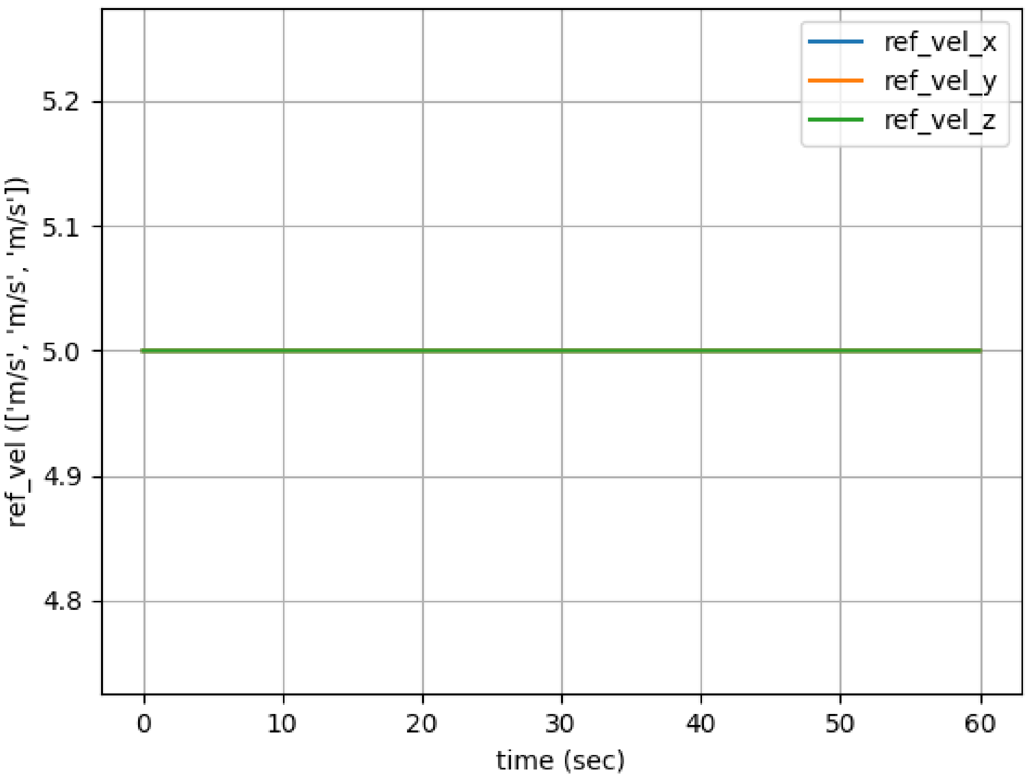

3400000
3300000
3200000
ref_pos_z
4800000
4600000
ref_pos_y
−3000000
−2900000
−2800000
−2700000
ref_pos_x


ref_vel_x
ref_vel_y
ref_vel_z
0.04
0.02
0.00
−0.02
−0.04
ref_vel (['m/s', 'm/s', 'm/s'])
0
10
20
30
40
50
60
time (sec)

motion3：匀速

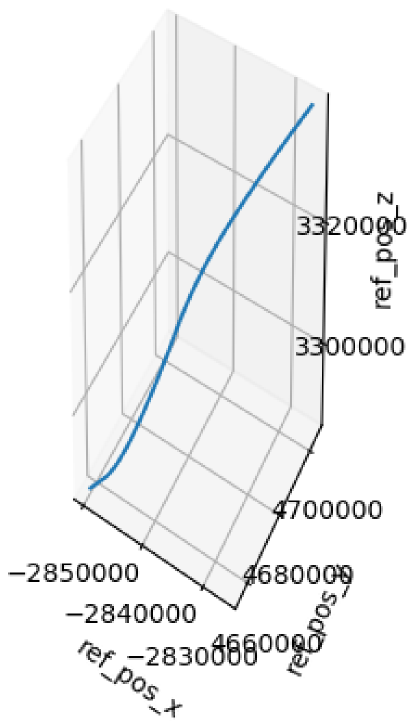


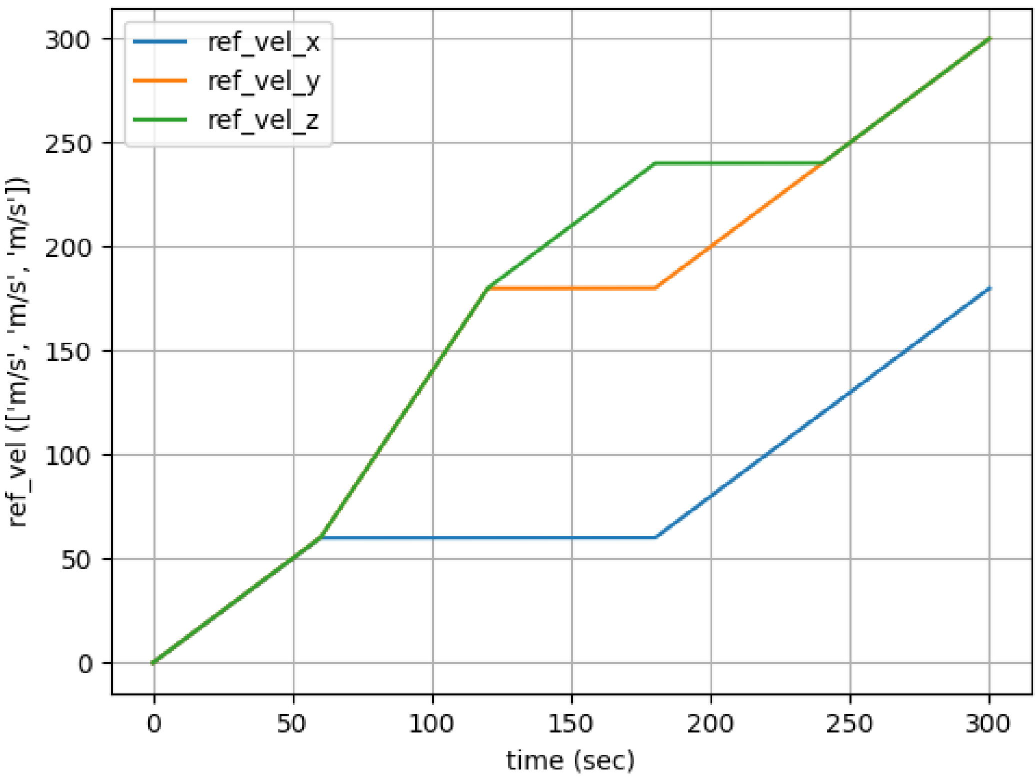
ref_pos_x
ref_pos_y
ref_pos_z
−2.849e6
3.287e6
−900
−850
−800
−750
−700
−650
−600
200
250
300
350
400
450
500
200
250
300
350
400
450



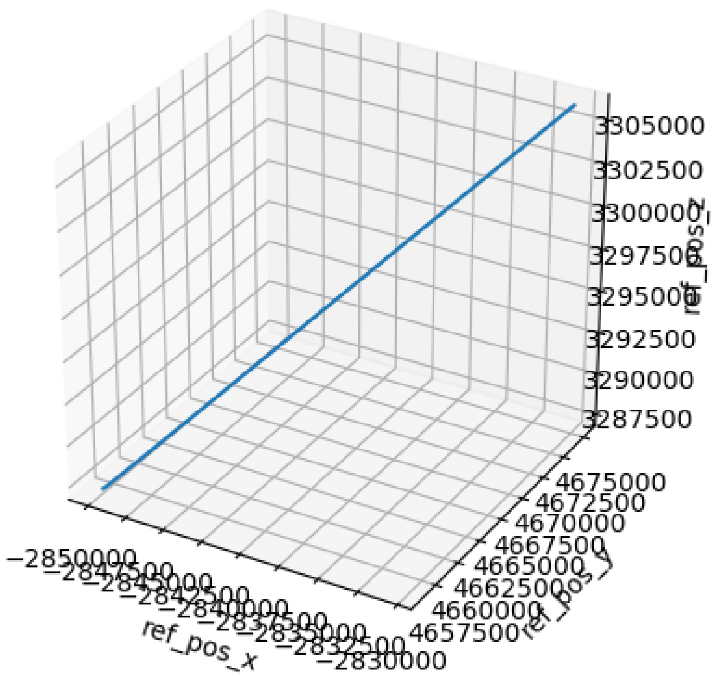
ref_vel_x
ref_vel_y
ref_vel_z
ref_vel (['m/s', 'm/s', 'm/s'])
time (sec)
5.2
5.1
5.0
4.9
4.8
0
10
20
30
40
50
60

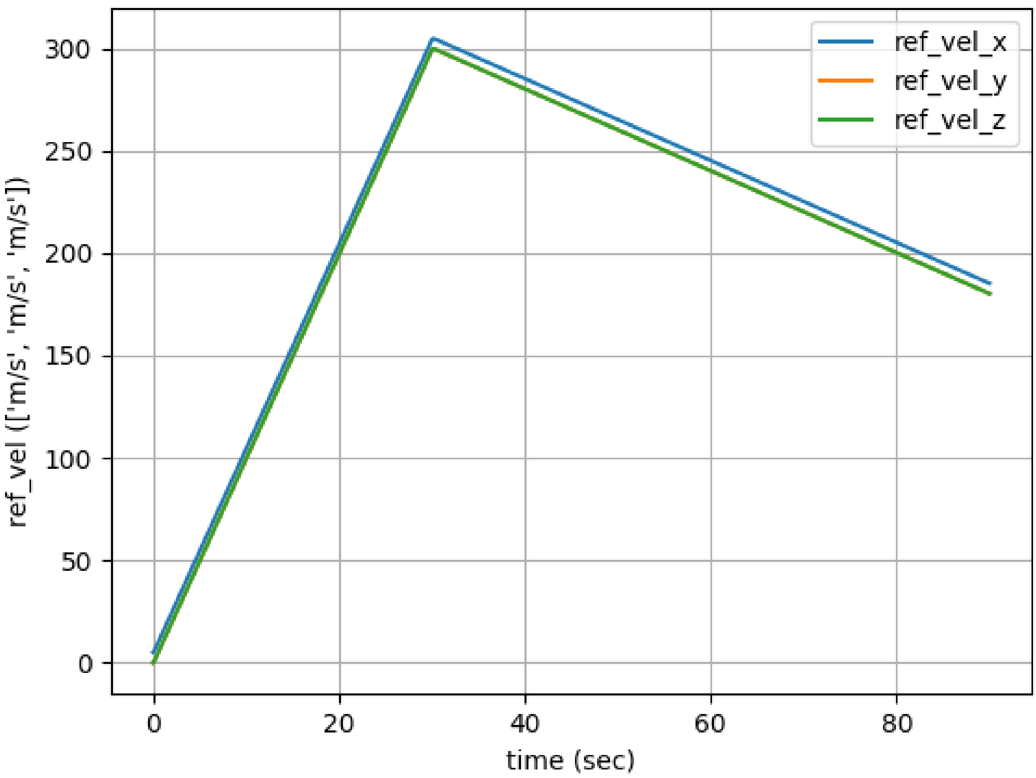
motion4：加速





motion5: 先加速后减速





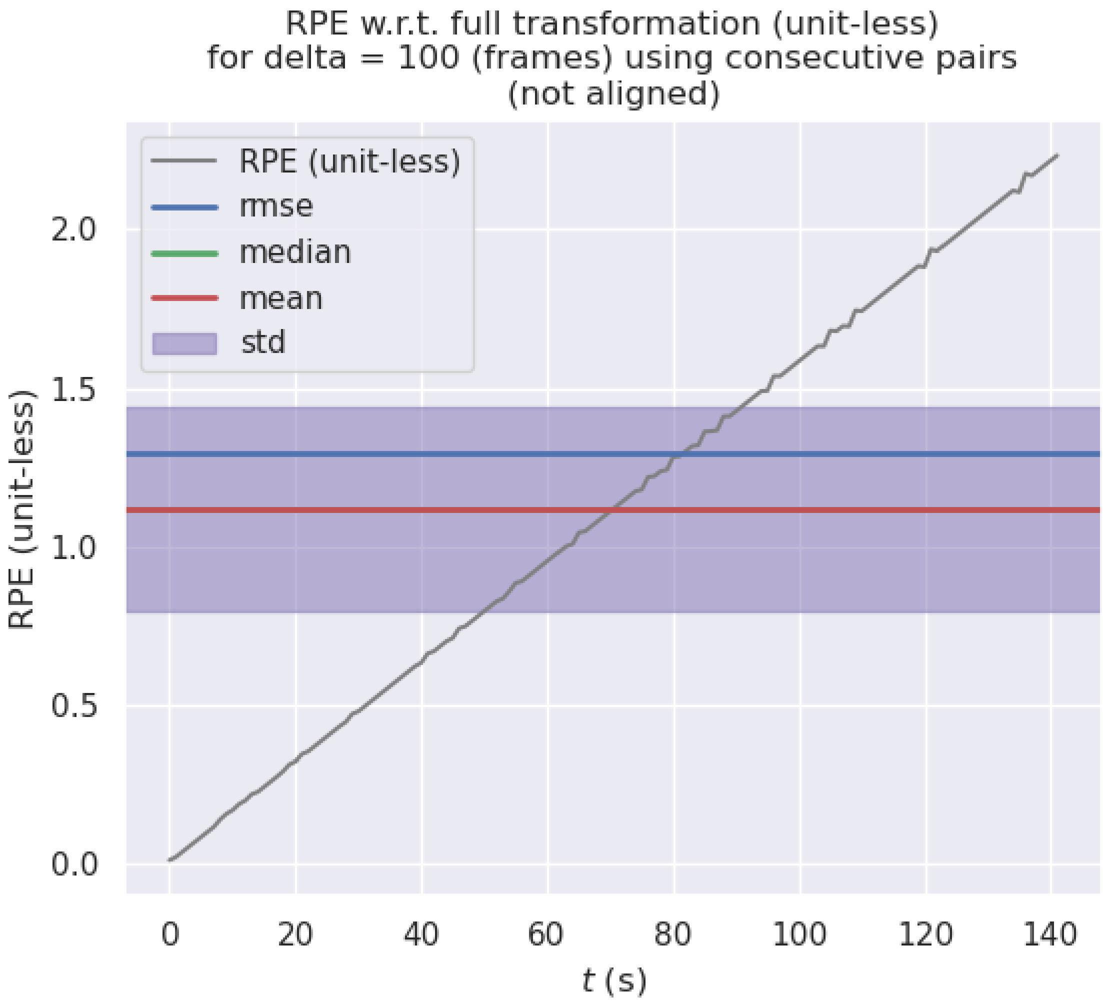
评估

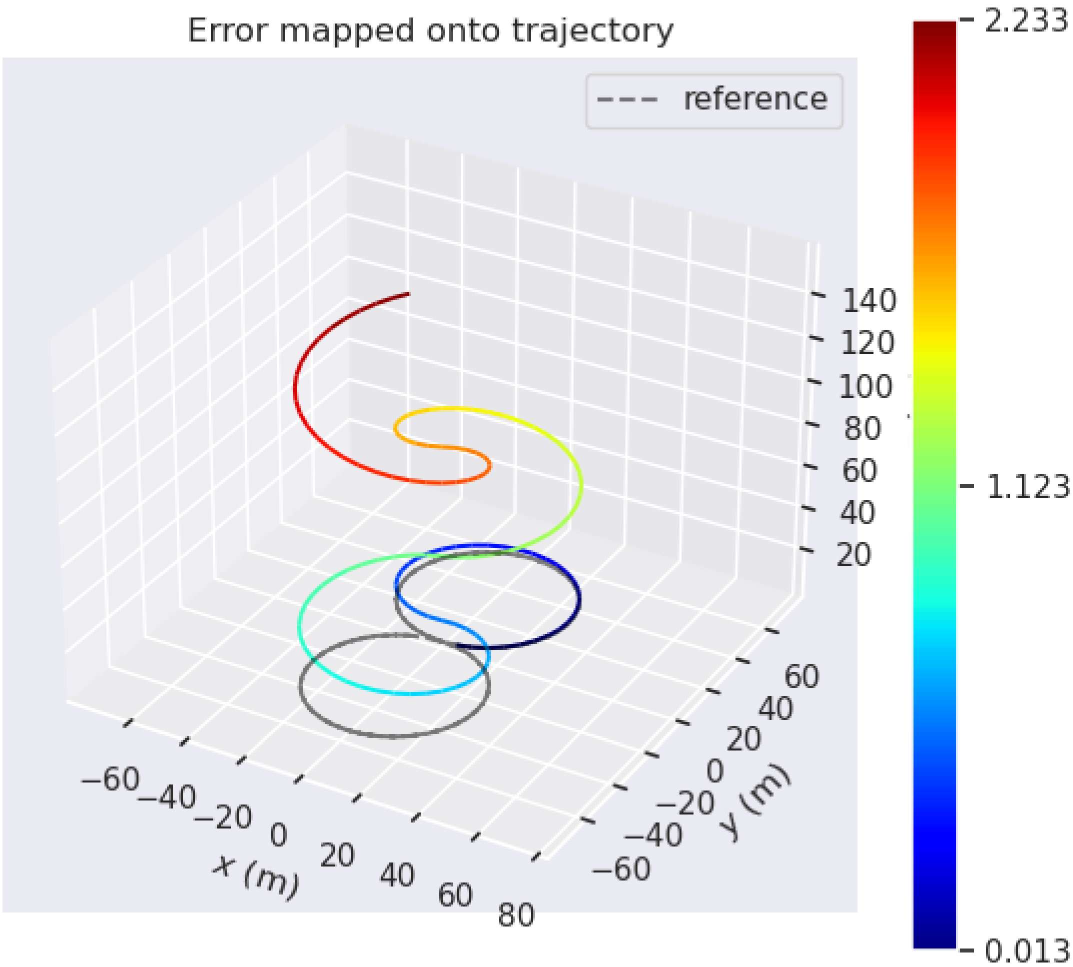
```
evo_rpe tum gt.txt ins.txt -r full --delta 100 --plot --plot_mode xyz
```

motion1： 绕“8”字

中值法

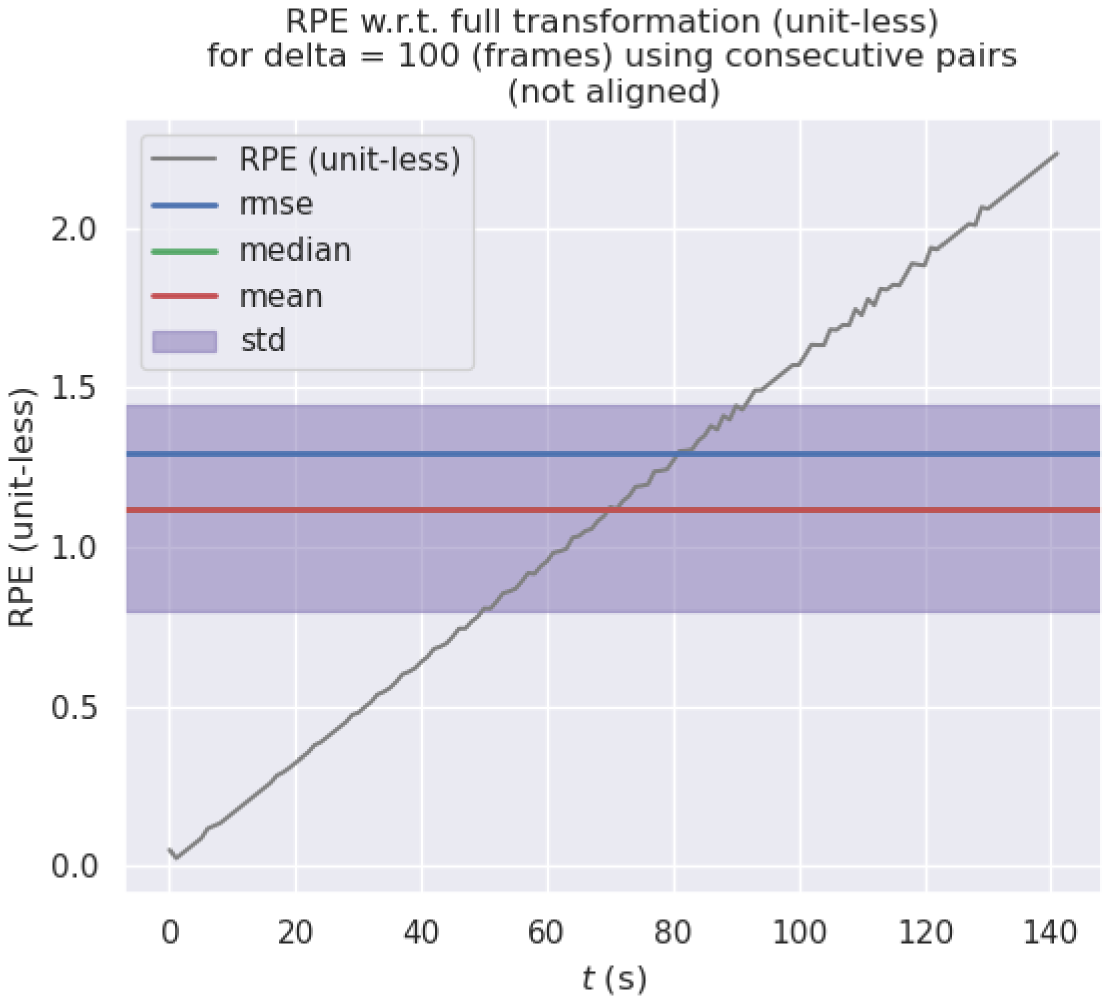
```
   max   2.233374
  mean   1.121340
median   1.120641
   min   0.013039
  rmse   1.294285
   sse   237.874536
   std   0.646351
```



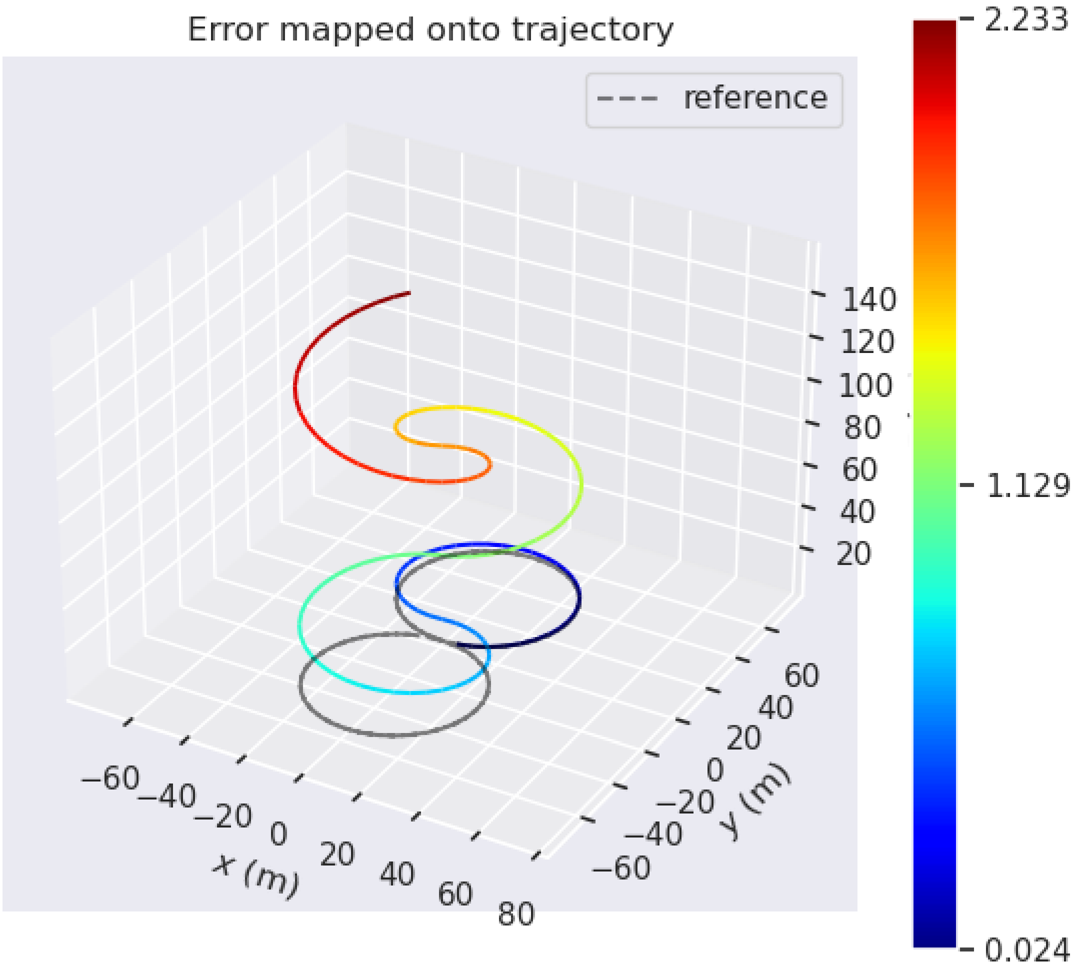


欧拉法

```
   max    2.233374
   mean   1.121636
 median   1.121582
    min   0.023918
   rmse   1.294462
    sse   237.939866
    std   0.646194
```



RPE w.r.t. full transformation (unit-less)
for delta = 100 (frames) using consecutive pairs
(not aligned)
RPE (unit-less)
rmse
median
mean
std
2.0
1.5
1.0
0.5
0.0
RPE (unit-less)
0
20
40
60
80
100
120
140
t (s)

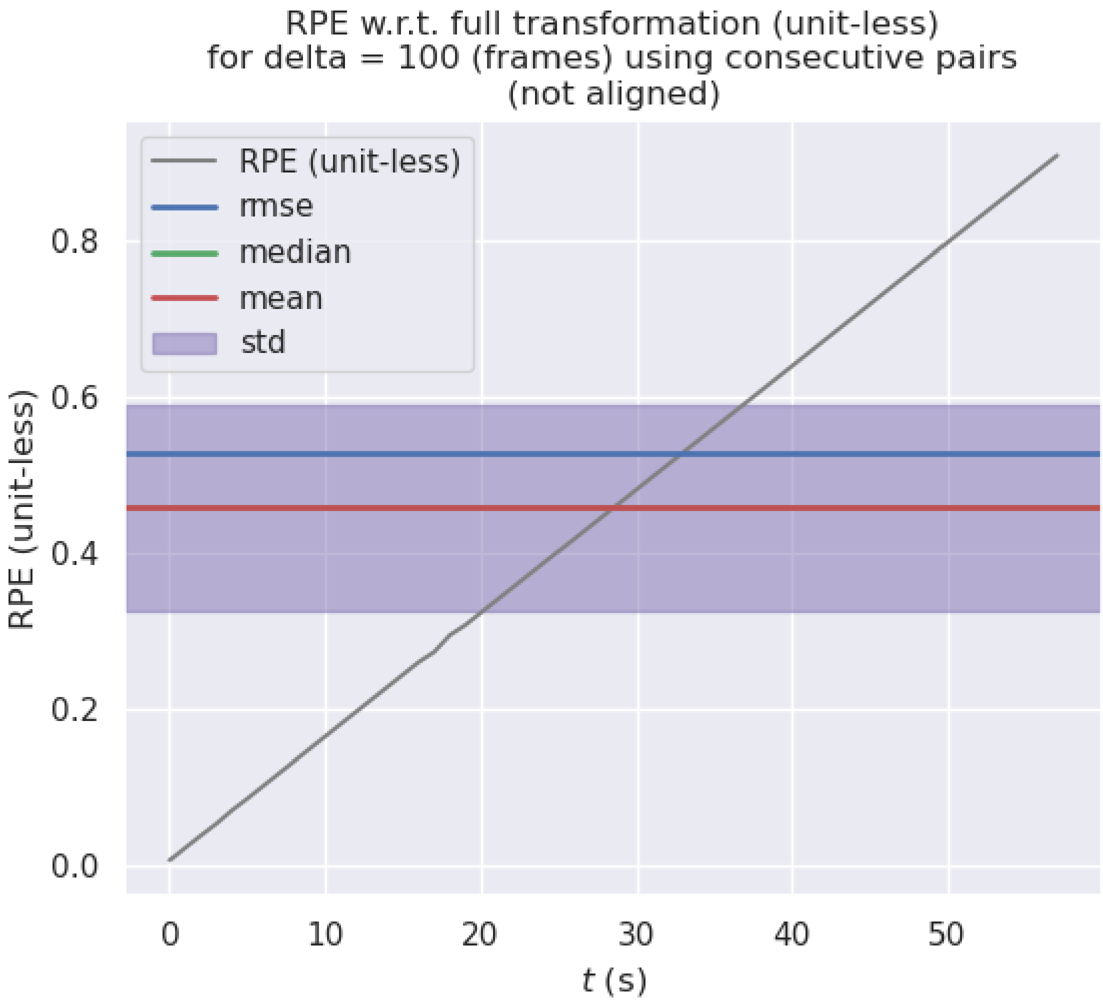


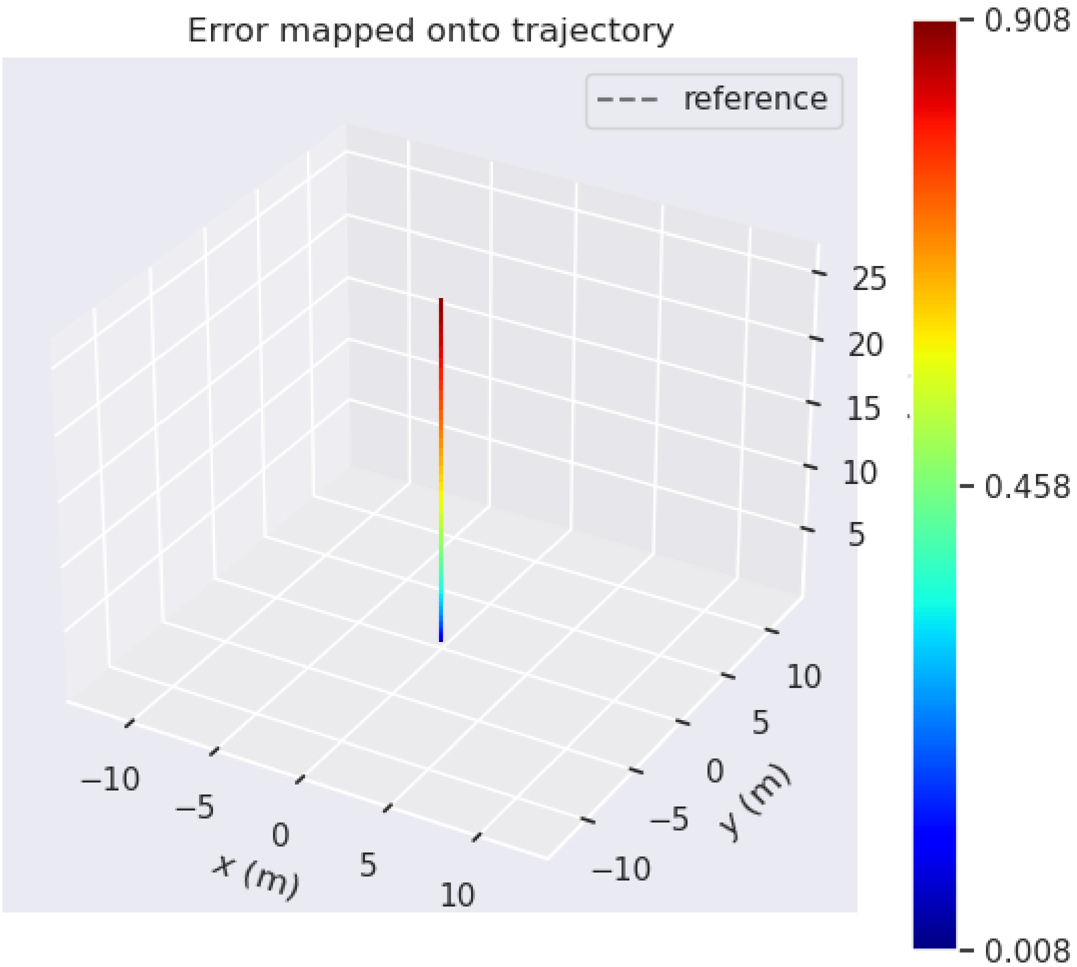
Error mapped onto trajectory
reference
140
120
100
80
60
40
20
60
40
20
0
−20
−40
−60
y (m)
−60
−40
−20
0
20
40
60
80
x (m)
2.233
1.129
0.024

motion2： 静止

中值法

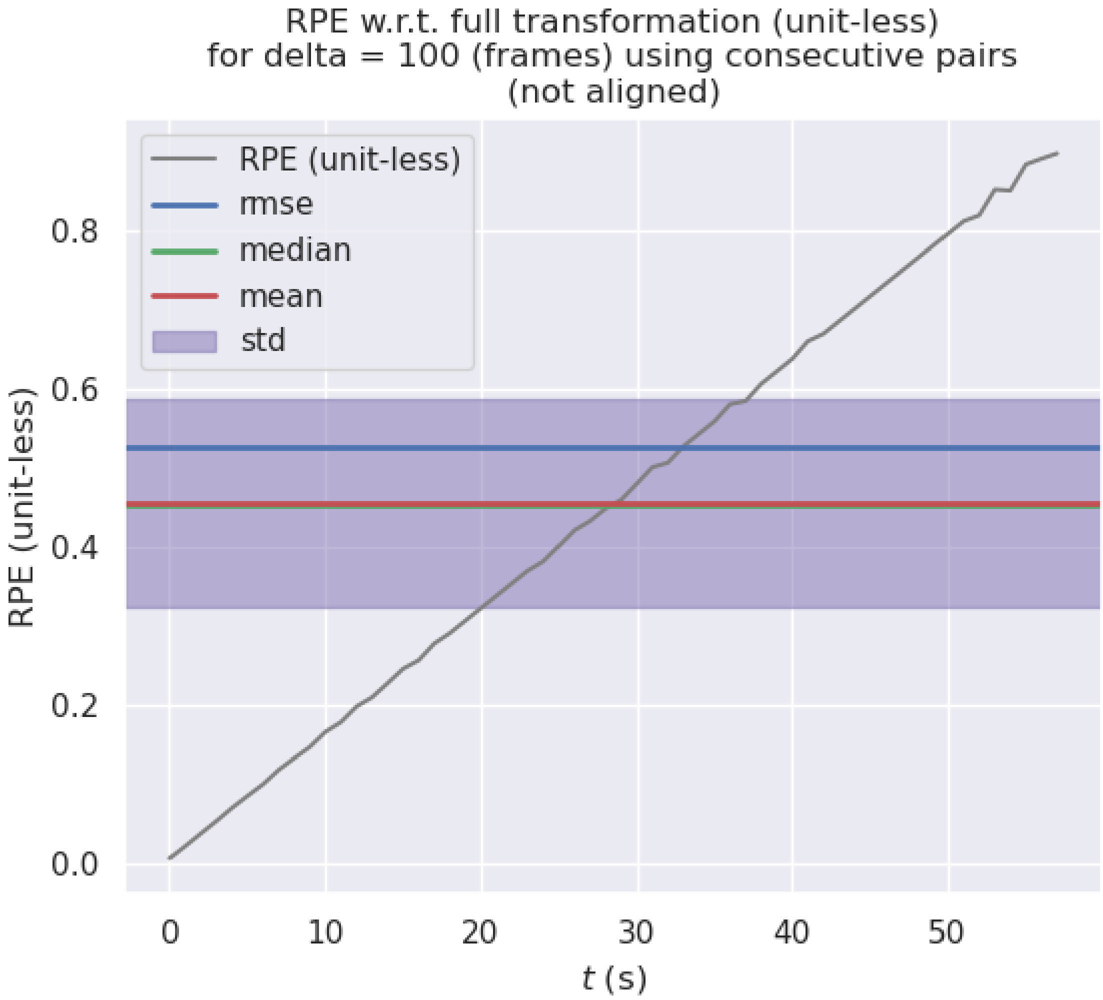
```
   max   0.907553
  mean   0.457722
median   0.457722
   min   0.007735
  rmse   0.528515
   sse   16.201016
   std   0.264231
```

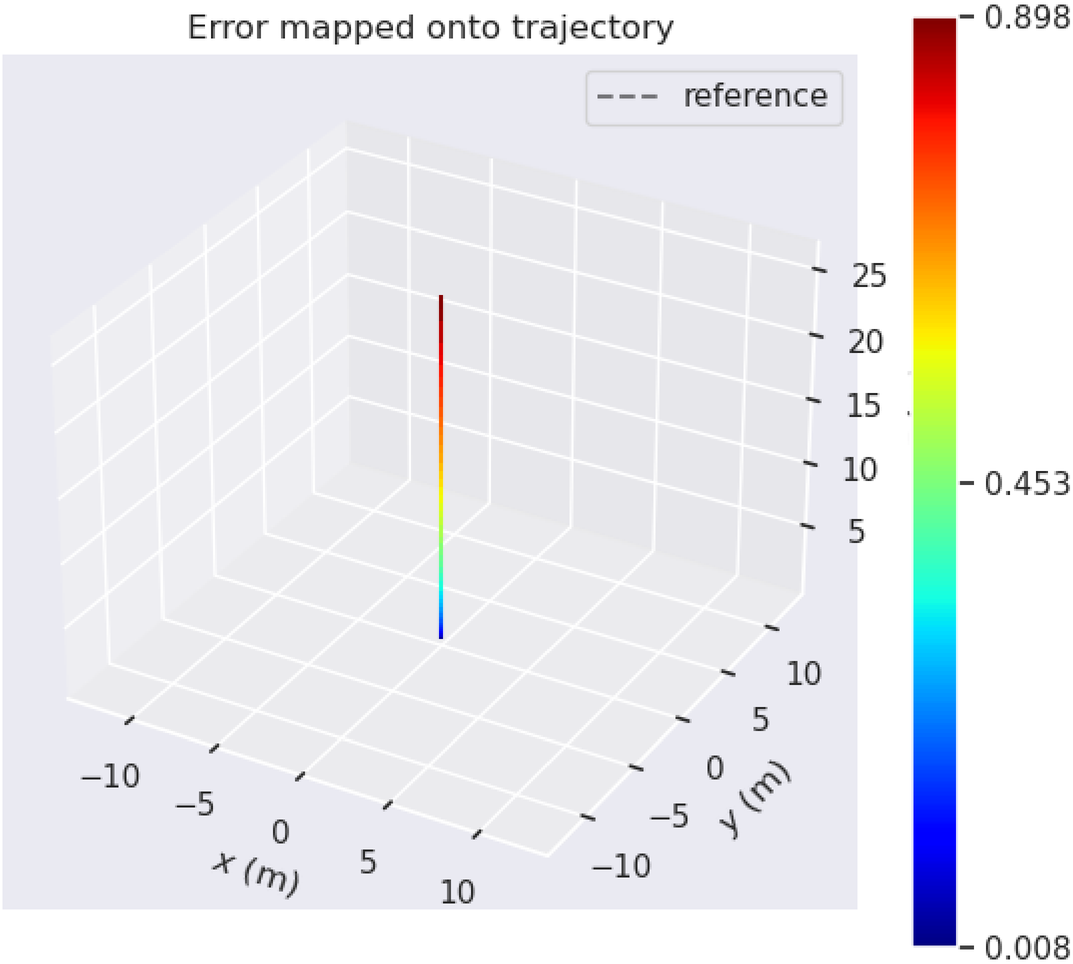




欧拉法

```
   max  0.898399
  mean  0.457565
median  0.455355
   min  0.007892
  rmse  0.528301
   sse  16.187888
   std  0.264076
```

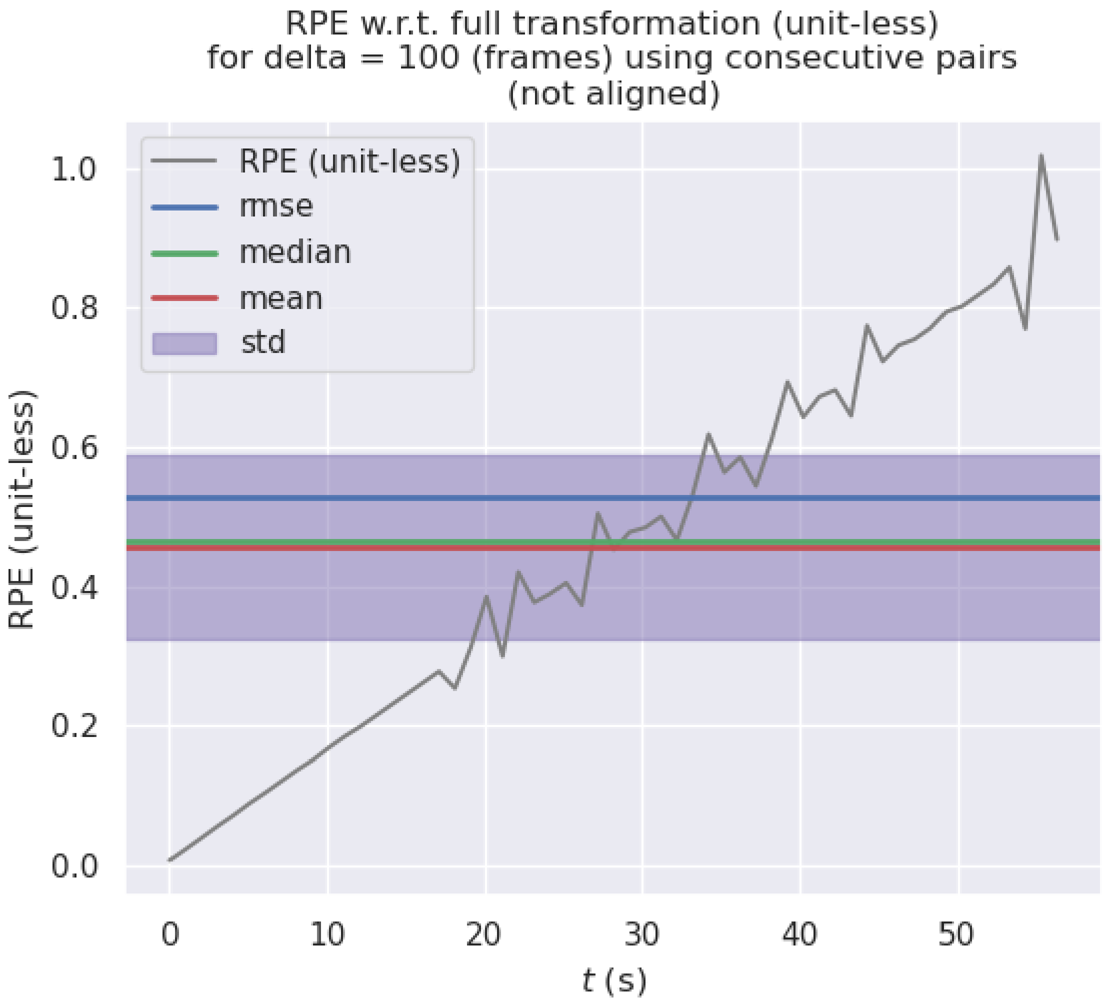

RPE w.r.t. full transformation (unit-less)
for delta = 100 (frames) using consecutive pairs
(not aligned)
RPE (unit-less)
rmse
median
mean
std
RPE (unit-less)
0.0
0.2
0.4
0.6
0.8
0
10
20
30
40
50
t (s)

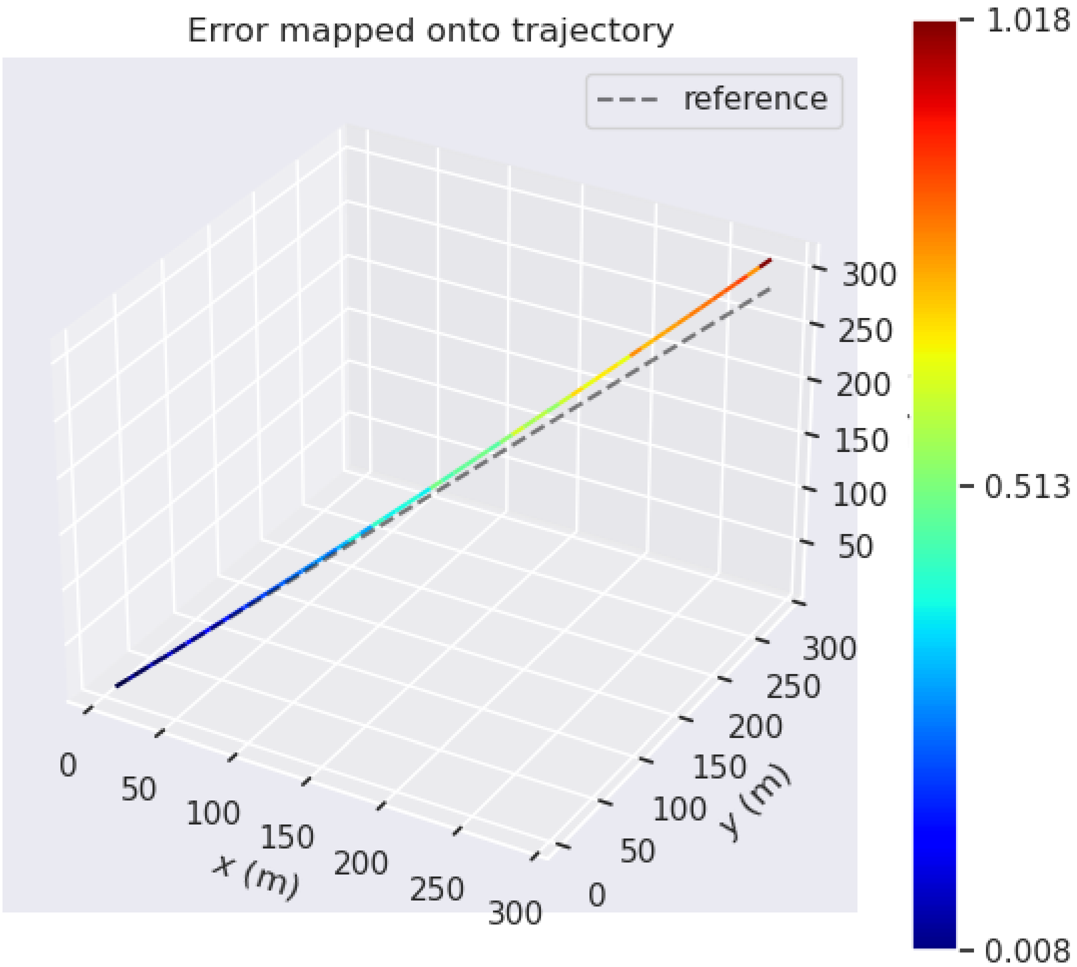

Error mapped onto trajectory
reference
0.898
0.453
0.008
x (m)
y (m)
−10
−5
0
5
10
25
20
15
10
5

motion3：匀速

中值法

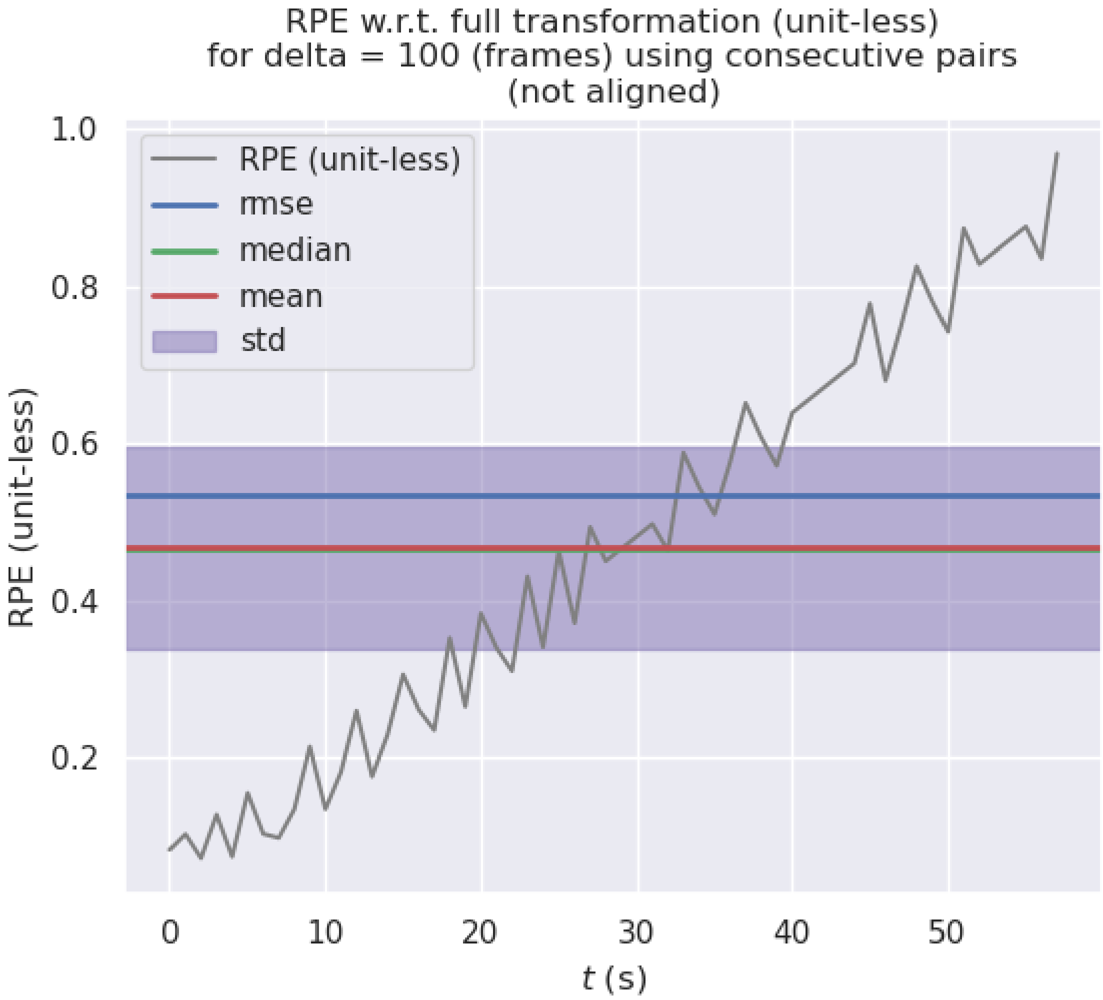
```
   max  1.017587
  mean  0.456960
median  0.466429
   min  0.008050
  rmse  0.528733
   sse  15.934814
   std  0.265981
```

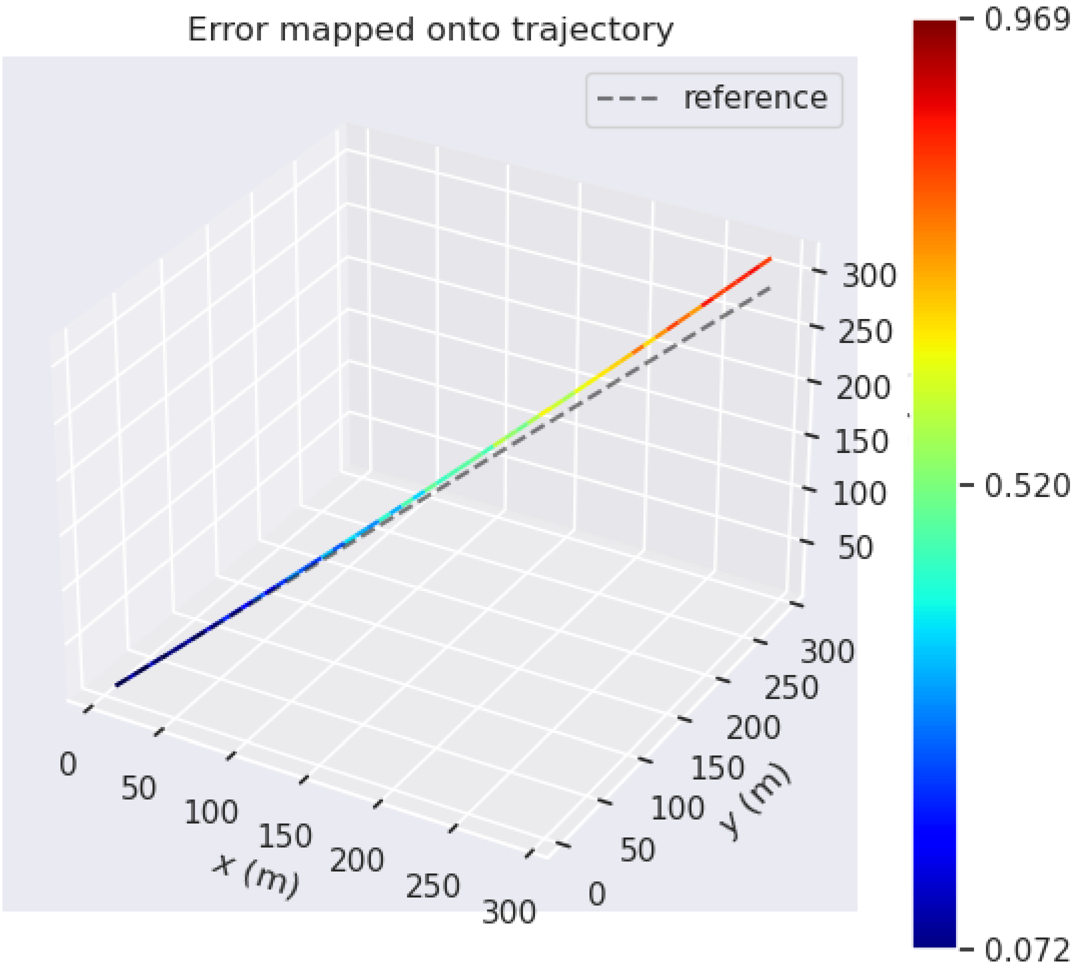




欧拉法

```
 max     0.969132
  mean   0.467047
median   0.464400
   min   0.071563
  rmse   0.533634
   sse   16.516409
   std   0.258134
```

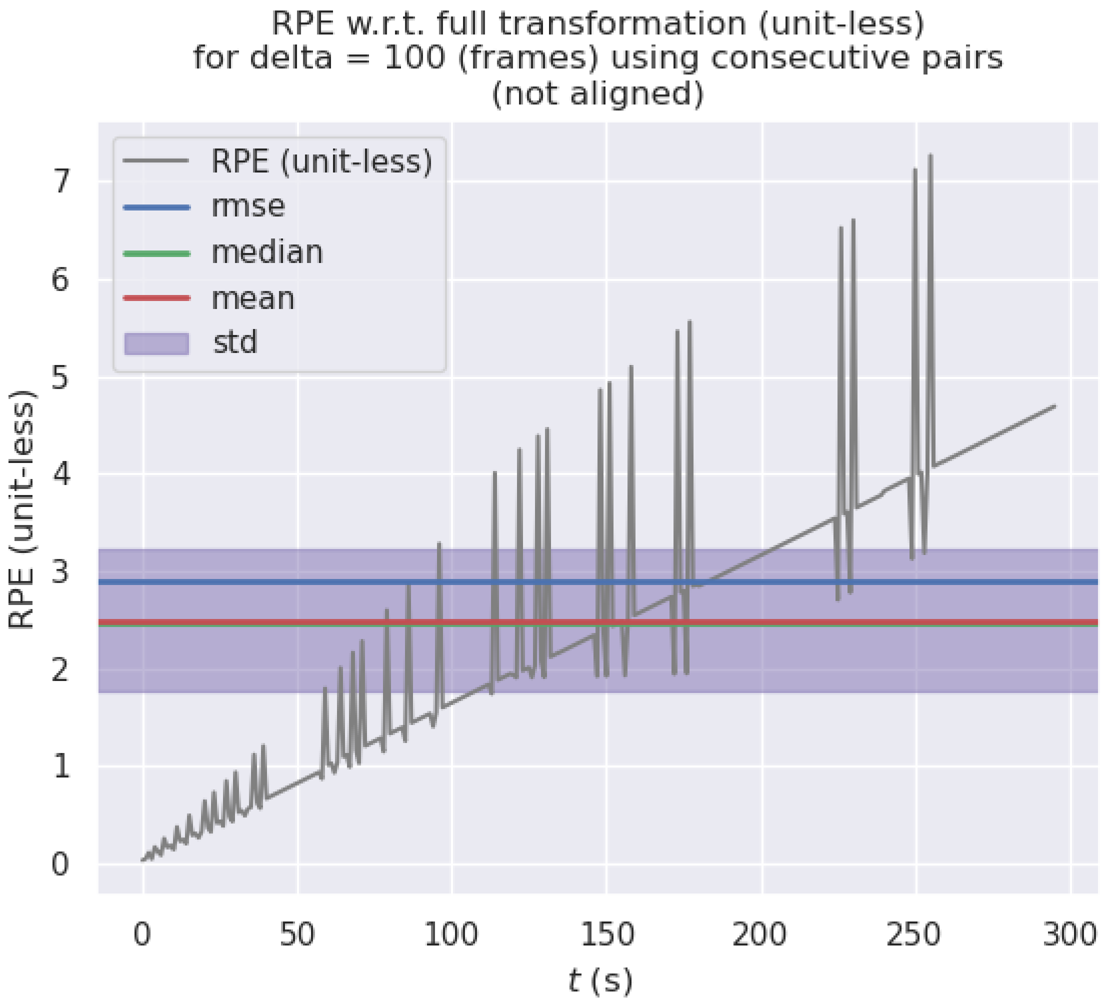

RPE w.r.t. full transformation (unit-less)
for delta = 100 (frames) using consecutive pairs
(not aligned)
RPE (unit-less)
rmse
median
mean
std
1.0
0.8
0.6
0.4
0.2
RPE (unit-less)
0
10
20
30
40
50
t (s)

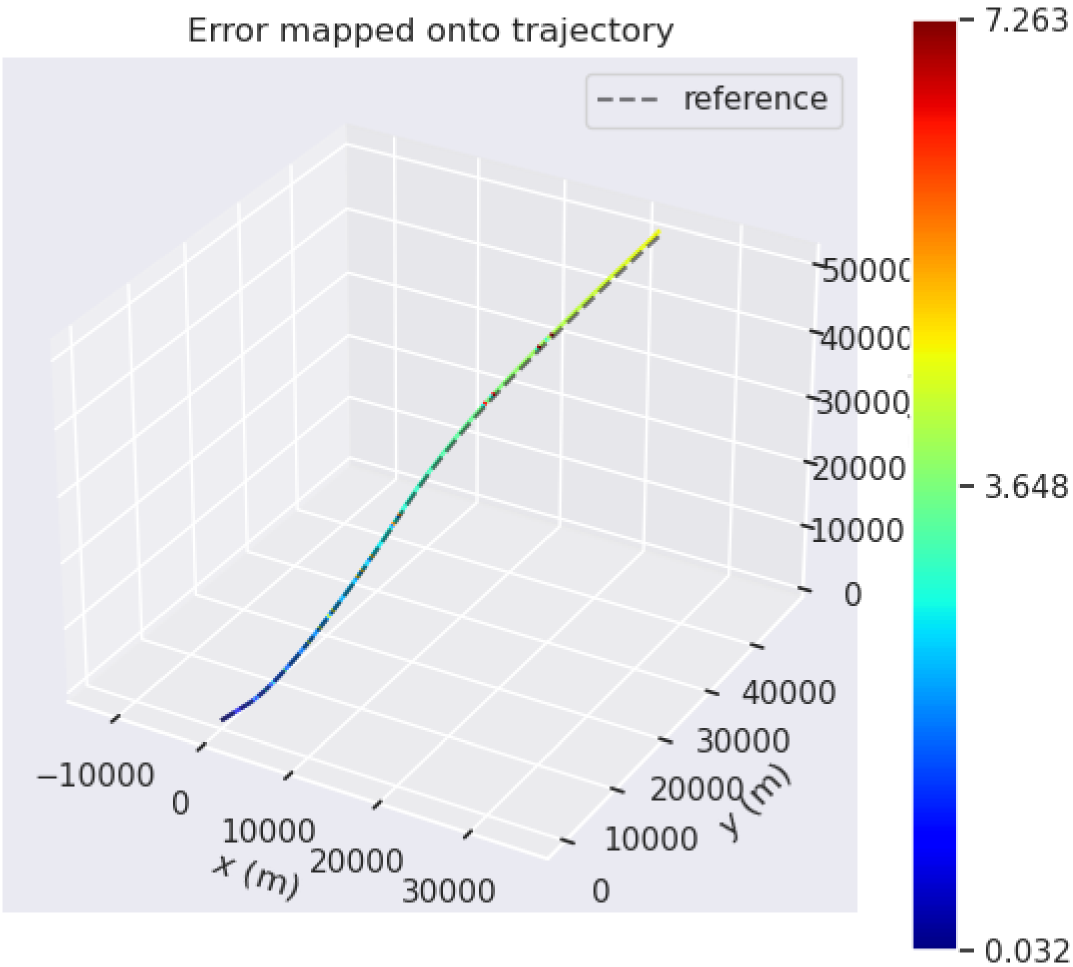

Error mapped onto trajectory
reference
0.969
0.520
0.072
300
250
200
150
100
50
0
50
100
150
200
250
300
x (m)
0
50
100
150
200
250
300
y (m)

motion4： 加速

中值法

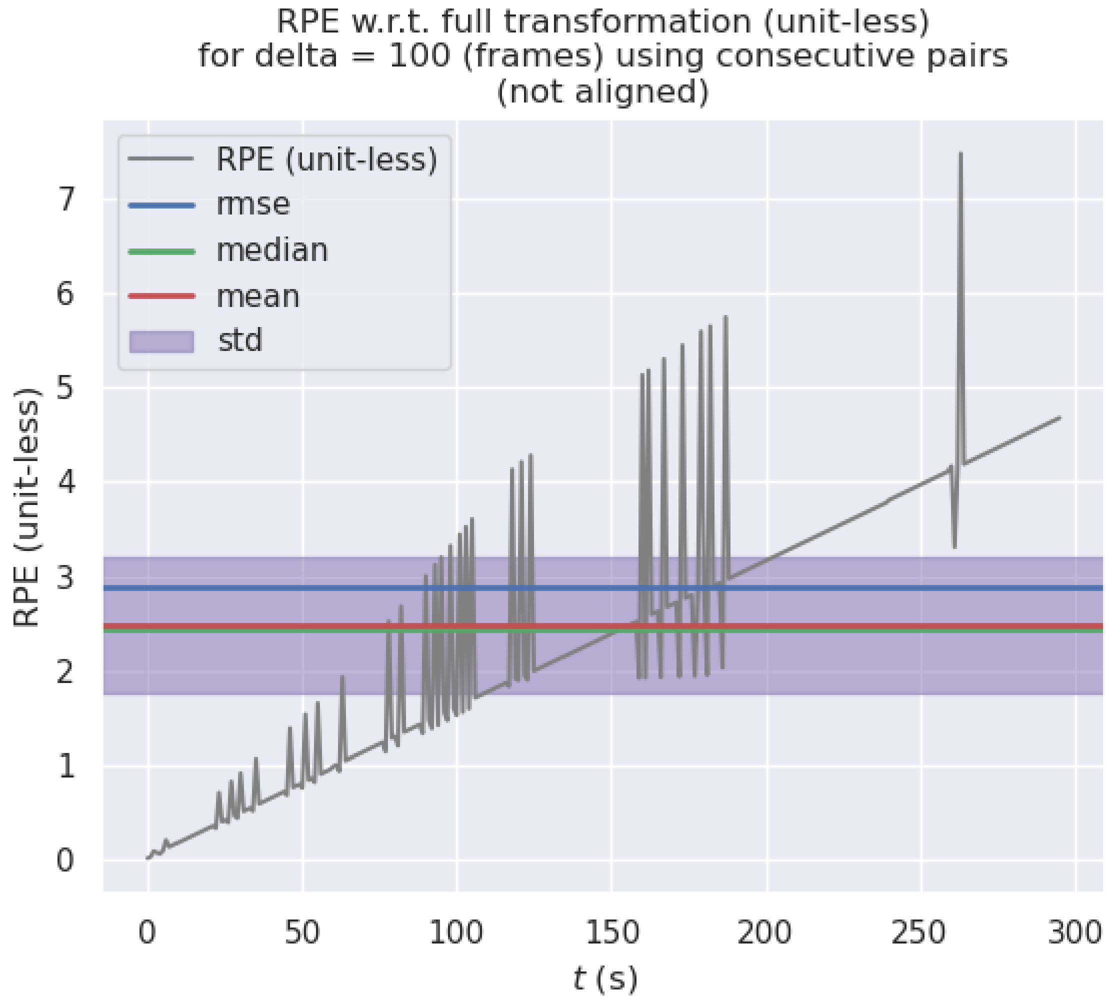
```
   max   7.263097
  mean   2.488803
median   2.457411
   min   0.032144
  rmse   2.887511
   sse   2467.964926
   std   1.464096
```

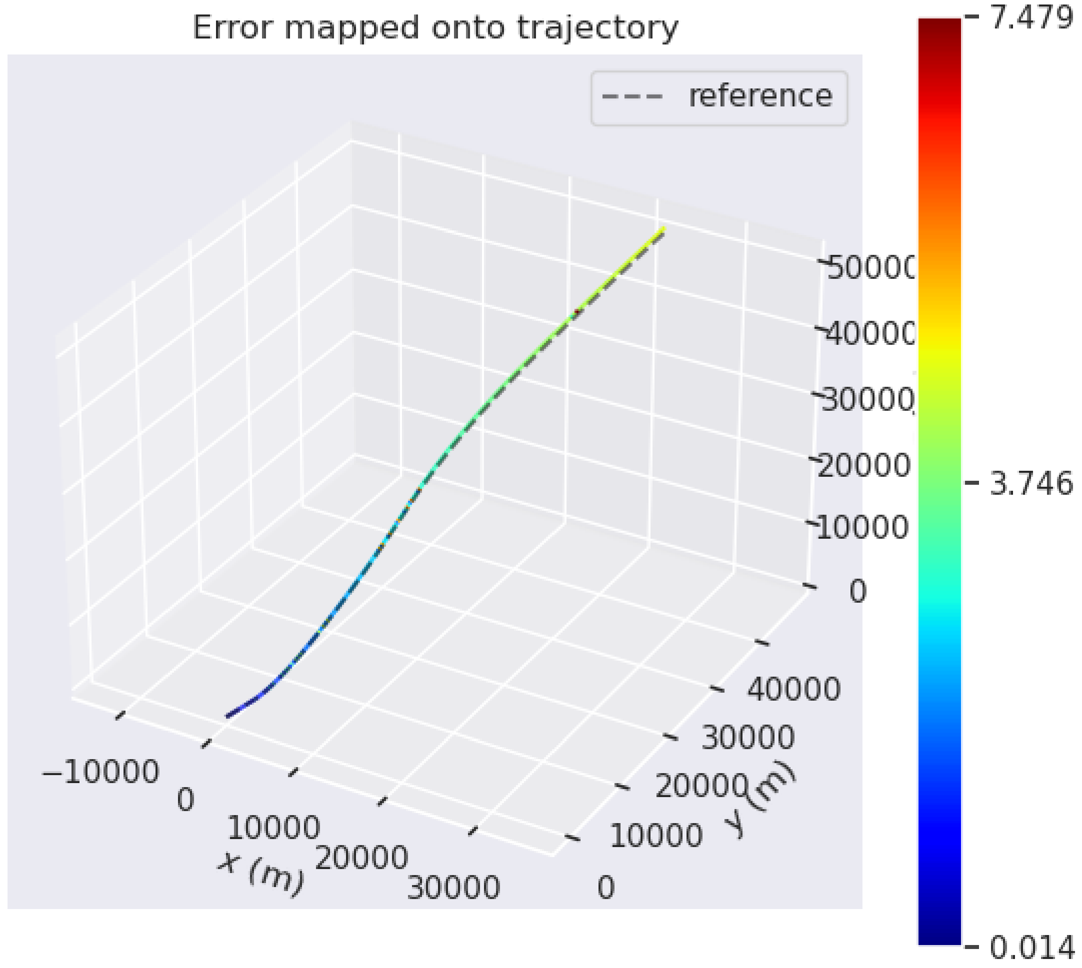




欧拉法

```
   max  7.478653
  mean  2.488508
median  2.429880
   min  0.013860
  rmse  2.877229
   sse  2450.420706
   std  1.444222
```

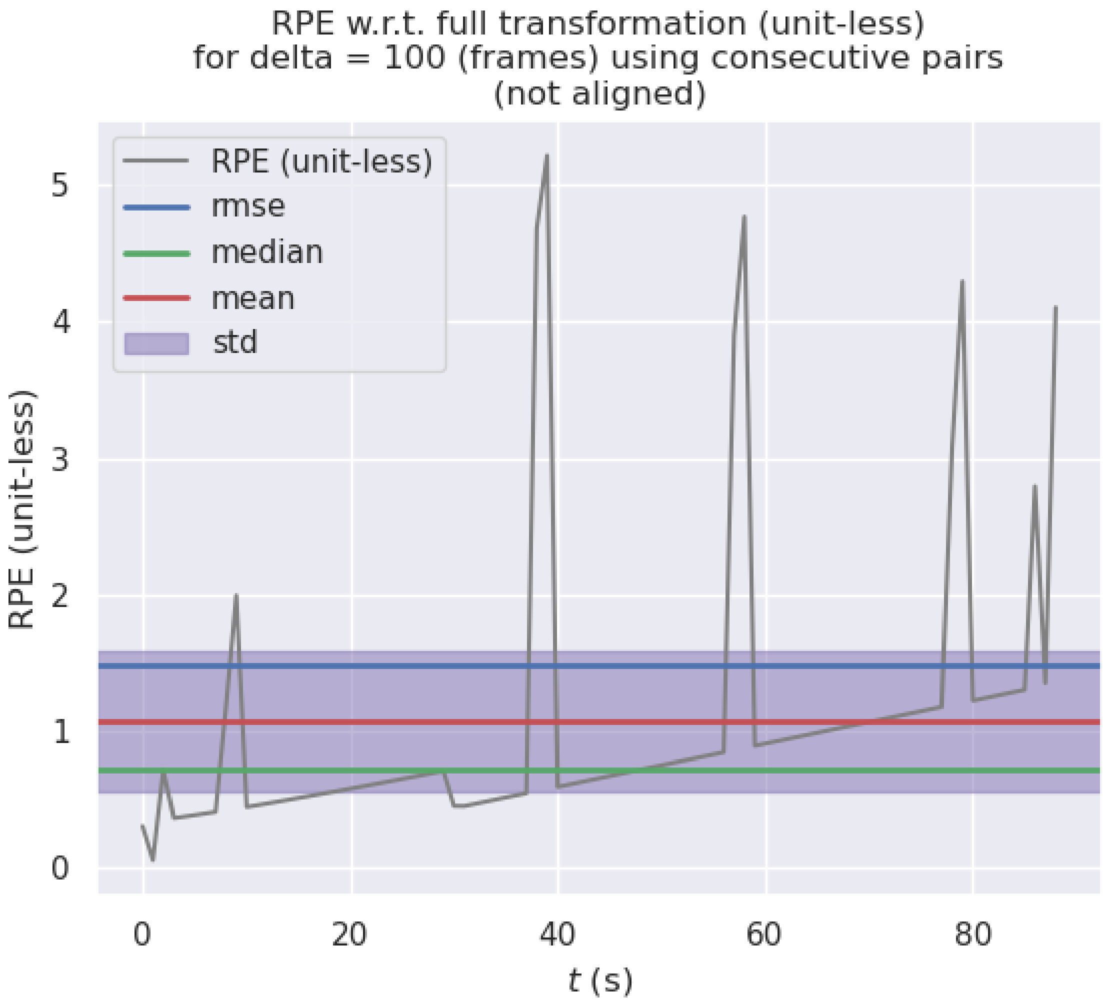

RPE w.r.t. full transformation (unit-less)
for delta = 100 (frames) using consecutive pairs
(not aligned)
RPE (unit-less)
rmse
median
mean
std
RPE (unit-less)
t (s)

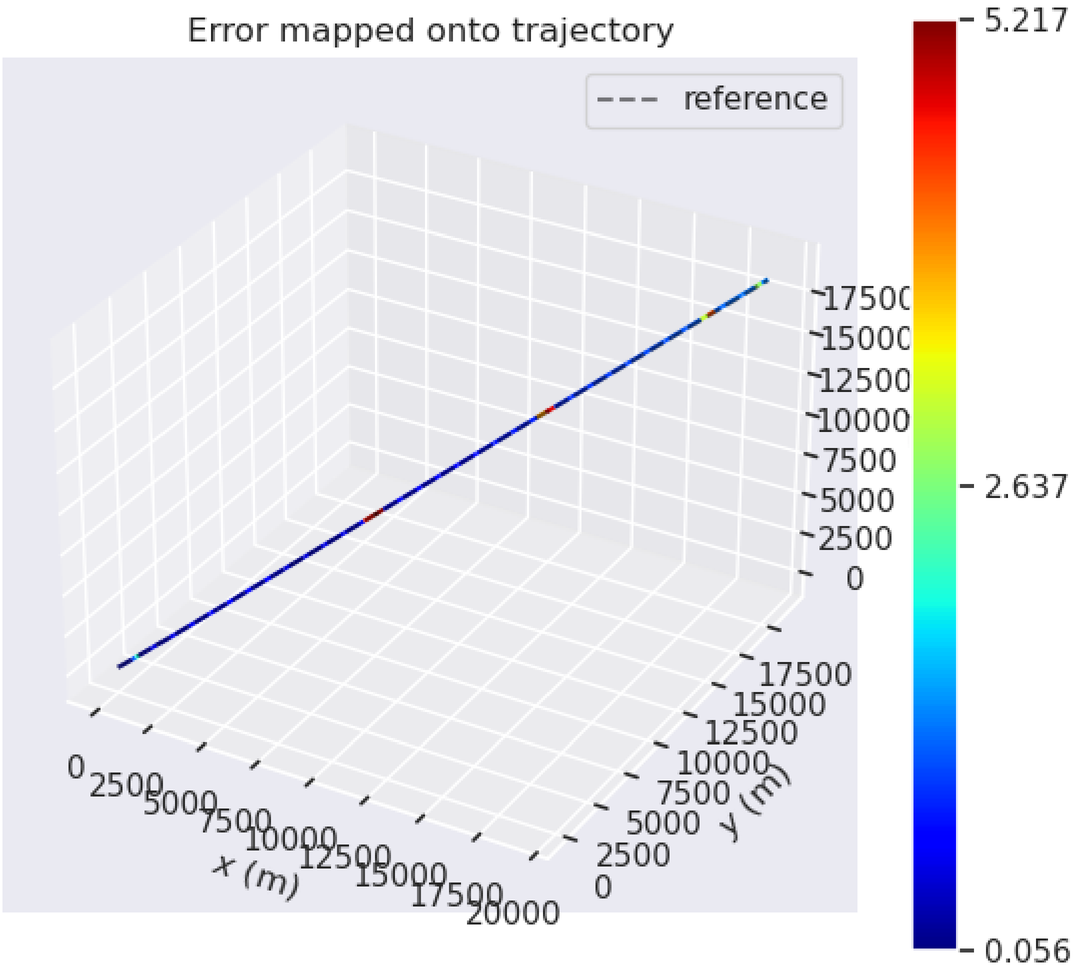

Error mapped onto trajectory
reference
x (m)
y (m)
7.479
3.746
0.014

motion5: 先加速后减速

中值法

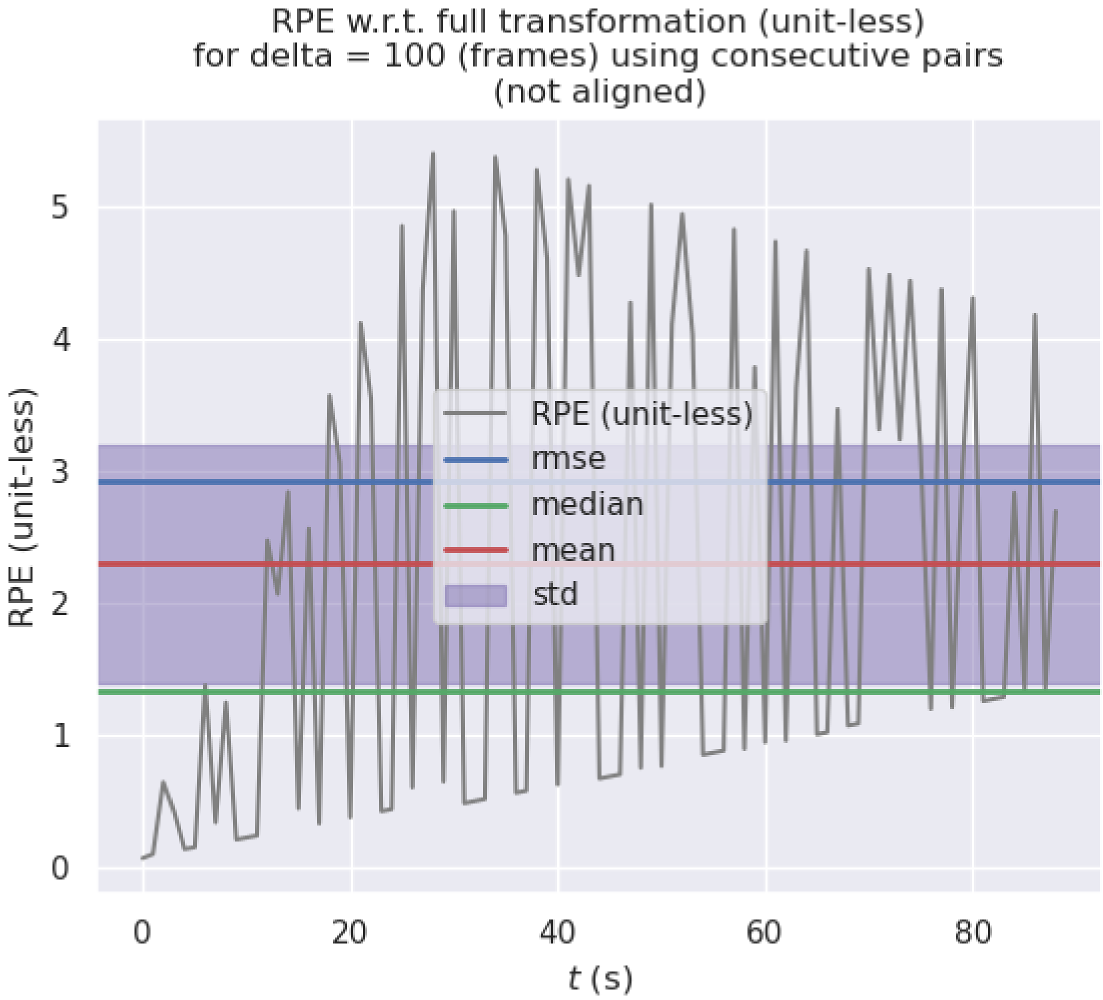
```
   max   5.217147
  mean   1.070638
median   0.719415
   min   0.055915
  rmse   1.483595
   sse   195.893711
   std   1.027029
```

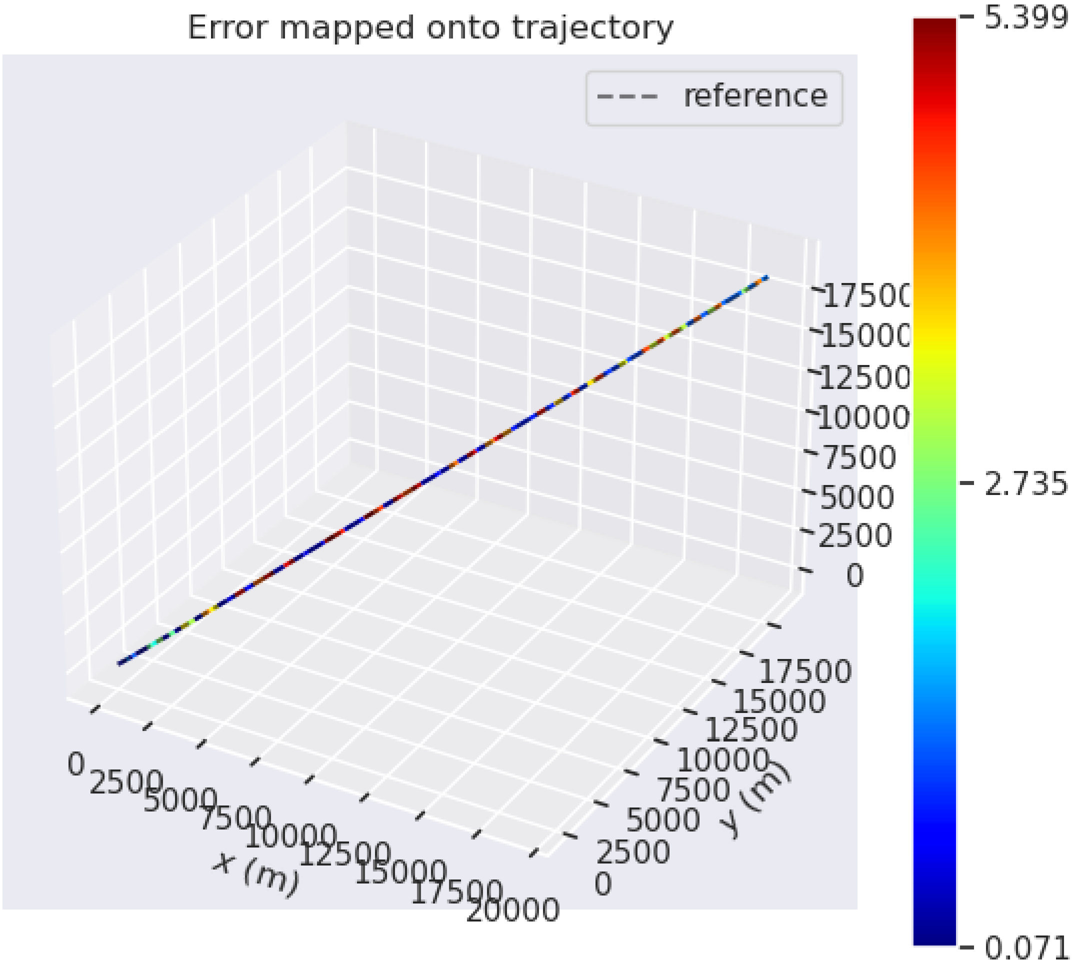




欧拉法

```
   max   5.399187
  mean   2.298571
median   1.339409
   min   0.070518
  rmse   2.917900
   sse   757.758390
   std   1.797418
```

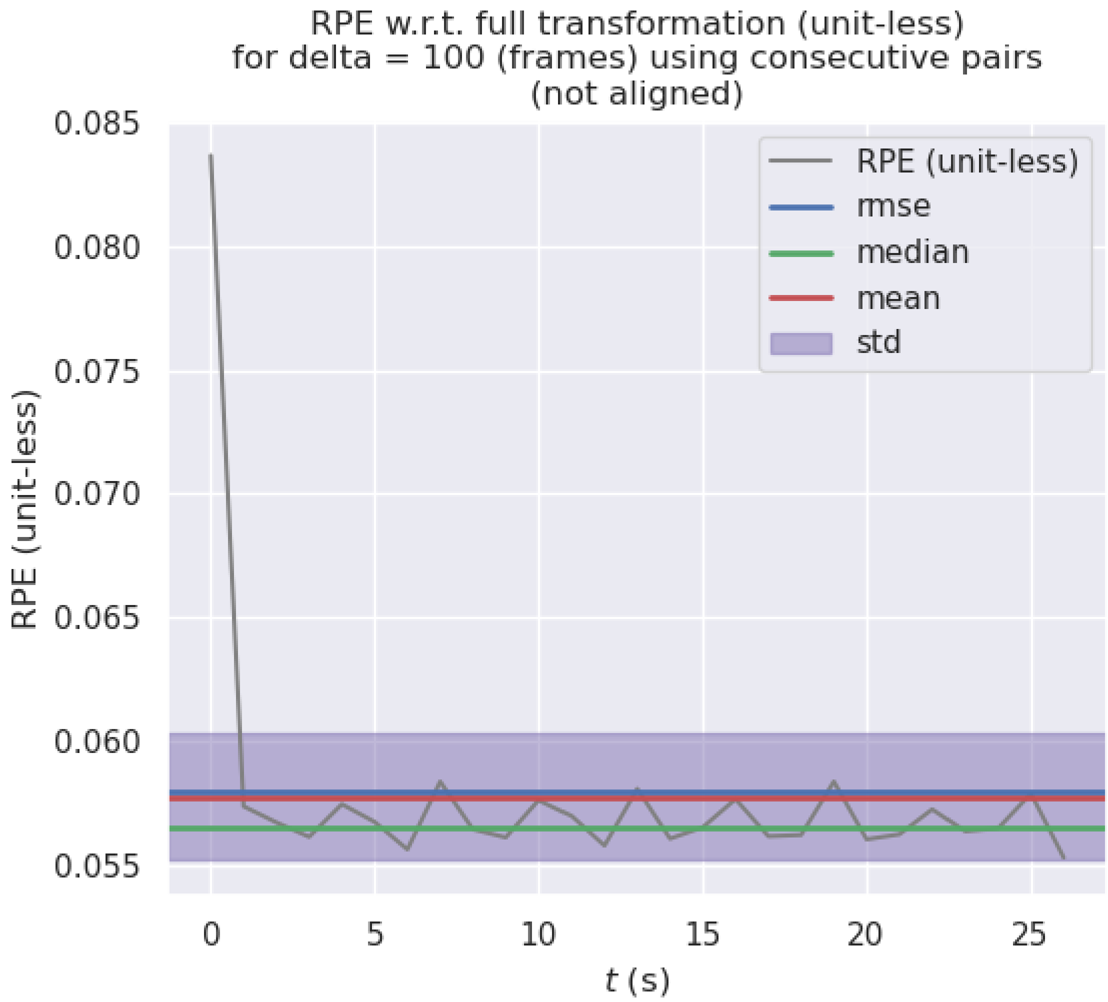

RPE w.r.t. full transformation (unit-less)
for delta = 100 (frames) using consecutive pairs
(not aligned)
RPE (unit-less)
rmse
median
mean
std
RPE (unit-less)
t (s)
0
20
40
60
80
0
1
2
3
4
5

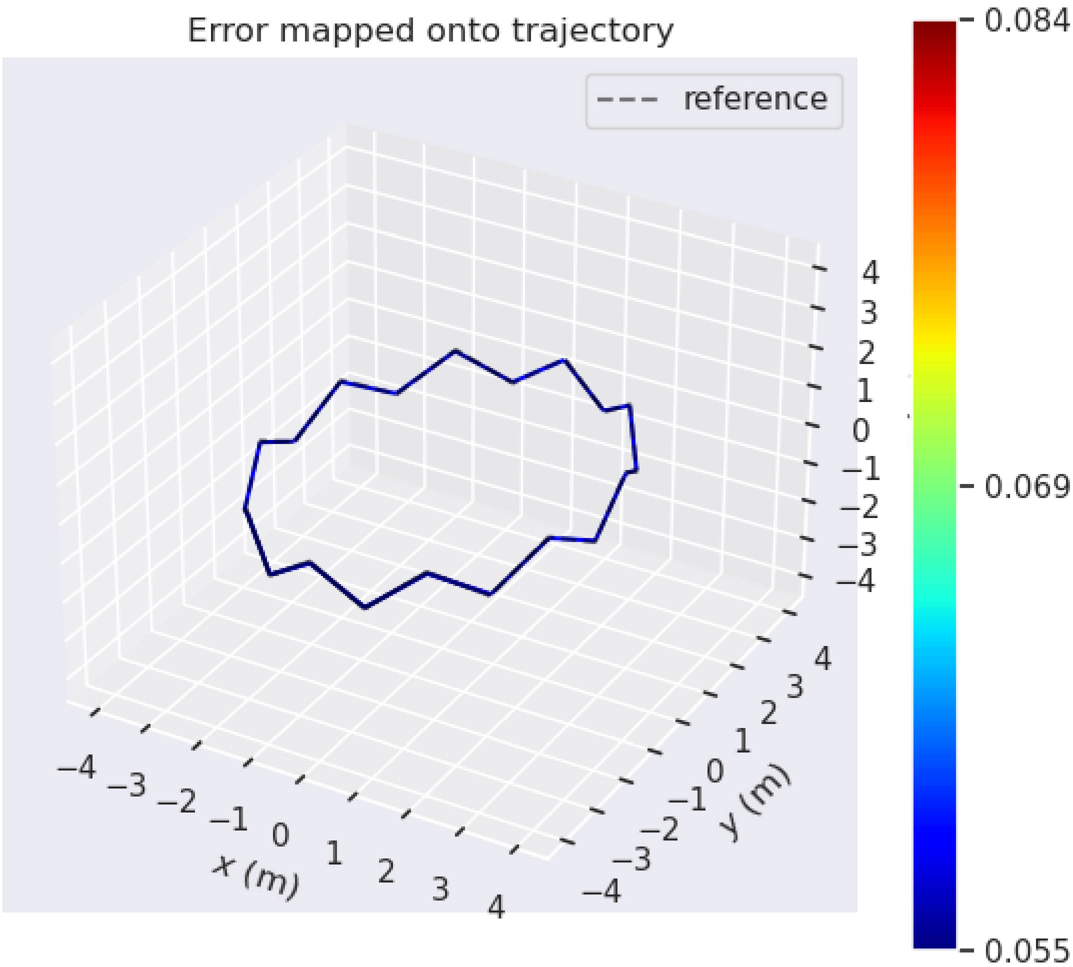

Error mapped onto trajectory
reference
x (m)
y (m)
0
2500
5000
7500
10000
12500
15000
17500
20000
5.399
2.735
0.071

课程提供

中值法

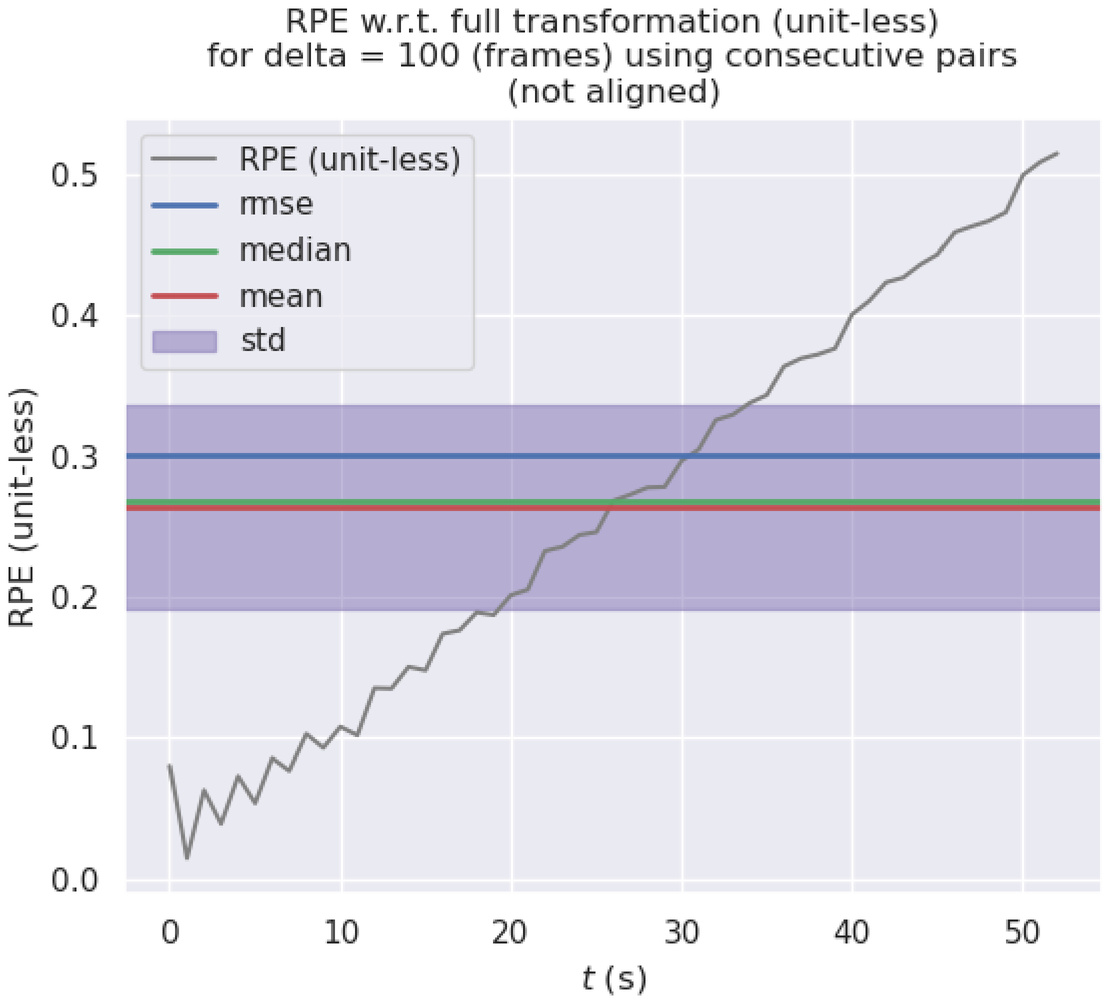
```
   max   0.083644
  mean   0.057792
median   0.056554
   min   0.055327
  rmse   0.058019
   sse   0.090889
   std   0.005137
```

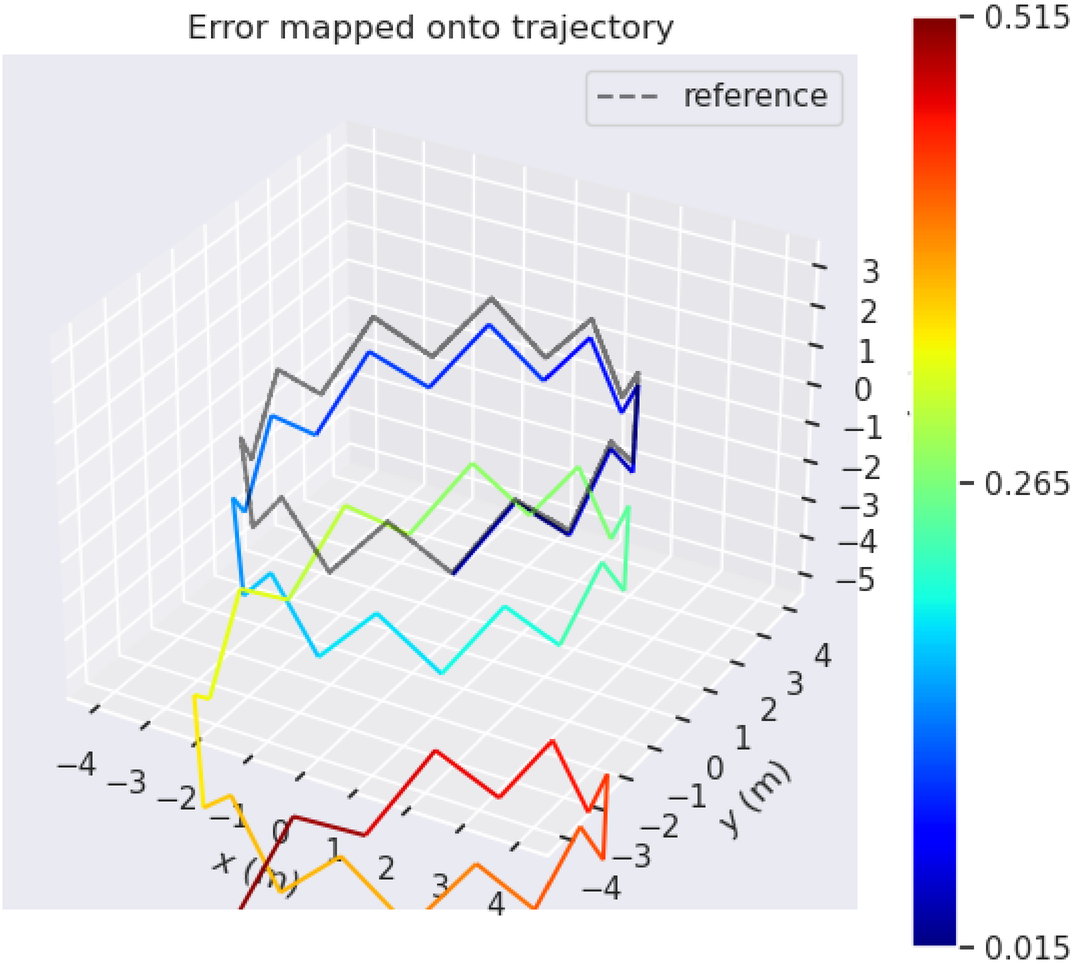




欧拉法

```
   max   0.514783
  mean   0.264081
median   0.268473
   min   0.014757
  rmse   0.300830
   sse   4.796445
   std   0.144084
```


RPE w.r.t. full transformation (unit-less)
for delta = 100 (frames) using consecutive pairs
(not aligned)
RPE (unit-less)
rmse
median
mean
std
0.5
0.4
0.3
0.2
0.1
0.0
RPE (unit-less)
0
10
20
30
40
50
t (s)


Error mapped onto trajectory
reference
0.515
0.265
0.015
x (m)
y (m)

总结和思考、疑问

1. 一般来说，imu的角速度精度高,线性加速度精度低。
2. 对于静止和匀速运动(加速度为0)，中值法精度比欧拉法低。

原因：imu测得的线性加速度和角速度并不为0,由于imu的角速度变化量小,所以误差较小,欧拉法和中值法效果差不多。而通过线性加速度计算得到的速度会累积误差,中值法取平均值会加大位置的误差(相对欧拉法)。

3. 对于加减速运动,中值法精度比欧拉法高

原因:在变速运动下中值法取平均值就比较合理,而且角加速度和线性加速度绝对值越大,欧拉法误差会越大。