深蓝学院
shenlanxueyuan.com

视觉SLAM进阶课程
从零开始手写VIO-第5期

第7讲 VINS系统构建 作业讲评

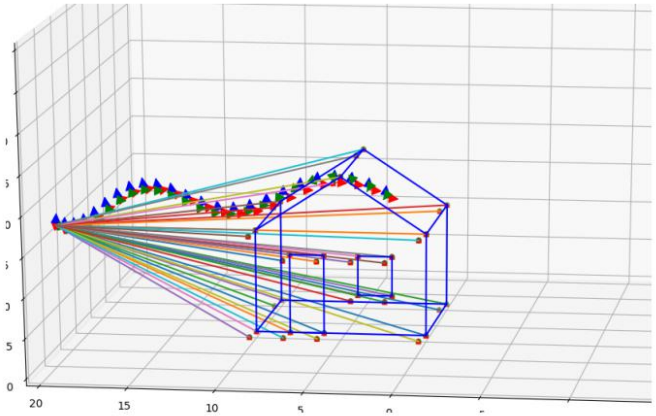
评讲人 吕华龙

【作业】

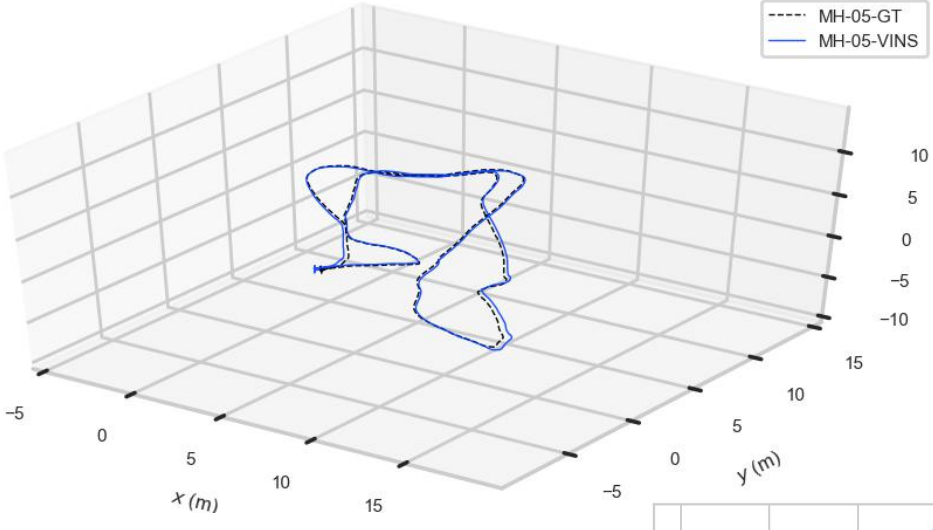
深蓝学院
shenlanxueyuan.com

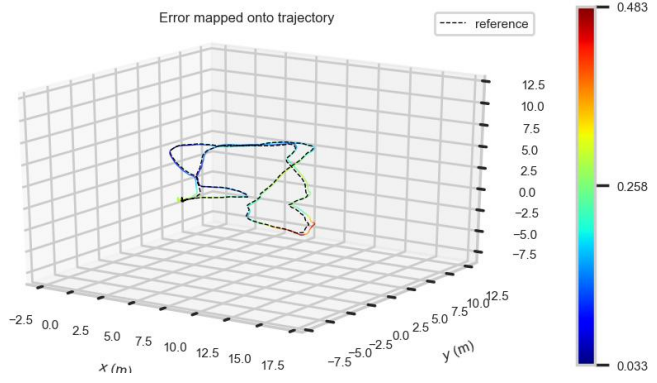
作业

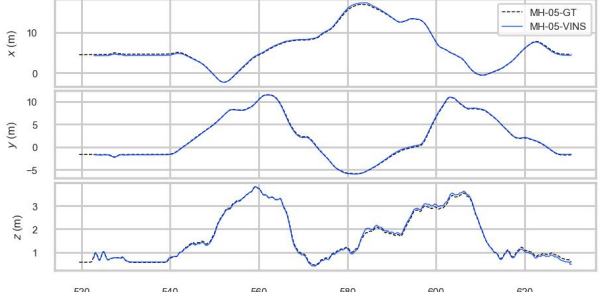
1. 将第二讲的仿真数据集（视觉特征，imu 数据）接入我们的 VINS 代码，并运行出轨迹结果。
   - 仿真数据集无噪声
   - 仿真数据集有噪声（不同噪声设定时，需要配置 vins 中 imu noise 大小。）

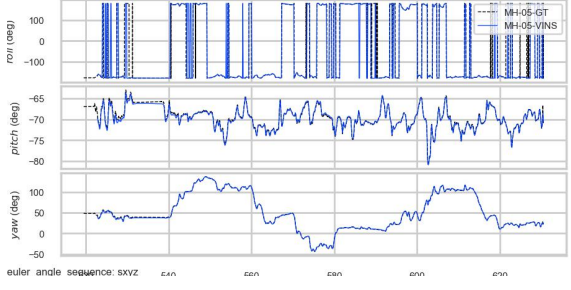


Test VINS Course Project with Euroc-MH-05

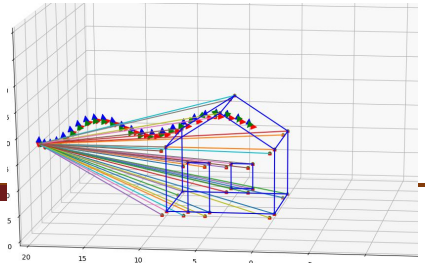








仿真数据的输入



深蓝学院
shenlanxueyuan.com

IMU测量数据

视觉观测数据

run_simulation pub_IMU


```cpp
void PubImageData()
{
    string sImage_file = sConfig_path + "MH_05_cam0.txt";

    cout << "1 PubImageData start sImage_file: " << sImage_file << endl;

    ifstream fsImage;
    fsImage.open(sImage_file.c_str());
    if (!fsImage.is_open())
    {
        cerr << "Failed to open image file! " << sImage_file << endl;
        return;
    }

    std::string sImage_line;
    double dStampNSec;
    string sImgFileName;

    // cv::namedWindow("SOURCE IMAGE", CV_WINDOW_AUTOSIZE);
    while (std::getline(fsImage, sImage_line) && !sImage_line.empty())
    {
        std::istringstream ssImuData(sImage_line);
        ssImuData >> dStampNSec >> sImgFileName;
        // cout << "Image t : " << fixed << dStampNSec << " Name: " << sImgFileName << endl;
        string imagePath = sData_path + "cam0/data/" + sImgFileName;

        Mat img = imread(imagePath.c_str(), 0);
        if (img.empty())
        {
            cerr << "image is empty! path: " << imagePath << endl;
            return;
        }
        pSystem->PubImageData(dStampNSec / 1e9, img);
        // cv::imshow("SOURCE IMAGE", img);
        // cv::waitKey(0);
        usleep(50000 * nDelayTimes);
    }
    fsImage.close();
}
```



```cpp
void PubImageData()
{
  string timeStamp_file = sData_path + "cam_pose_tum.txt";

  cout << "1 PubImageData start sImage_file: " << timeStamp_file << endl;

  ifstream ftime;
  ftime.open(timeStamp_file.c_str());
  if (!ftime.is_open())
  {
    cerr << "Failed to open image file! " << timeStamp_file << endl;
    return;
  }

  std::string sTime_line;
  double dStampNSec;

  int cnt = 0;
  while (std::getline(ftime, sTime_line) && !sTime_line.empty())
  {
    std::istringstream ssTimeData(sTime_line);
    ssTimeData >> dStampNSec;
    string feature_points_path =
        sData_path + "/keyframe/" + "all_points_" + std::to_string(cnt) + ".txt";
    std::vector<cv::Point2f> feature_points;

    ReadFeatureFromFile(feature_points_path, feature_points);

    if (feature_points.empty())
    {
      cout << "image feature points is empty! path: " << feature_points_path << endl;
      return;
    }

    pSystem->PubFeatureData(dStampNSec, feature_points);

    usleep(50000 * nDelayTimes);
    cnt++;
  }
}
```

run_simulation pub_Camera

深蓝学院 shenlanxueyuan.com

```
void System::PubImageData(double dStampSec, Mat &img)
{
    if (!init_feature)
    {...
    }

    if (first_image_flag)
    {...
    }
    // detect unstable camera stream
    if (dStampSec - last_image_time > 1.0 || dStampSec < last_image_time)
    {...
    }
    last_image_time = dStampSec;
    // frequency control
    if (round(1.0 * pub_count / (dStampSec - first_image_time)) <= FREQ)
    {...
    }
    else
    {...
    }

    TicToc t_r;
    // cout << "3 PubImageData t : " << dStampSec << endl;
    trackerData[0].readImage(img, dStampSec);

    for (unsigned int i = 0;; i++)
    {...
    }
    if (PUB_THIS_FRAME)
    {
        pub_count++;    Yijia He, a year ago • support for linux and maces
        shared_ptr<IMG_MSG> feature_points(new IMG_MSG());
        feature_points->header = dStampSec;
        vector<set<int>> hash_ids(NUM_OF_CAM);
        for (int i = 0; i < NUM_OF_CAM; i++)
        {
            auto &un_pts = trackerData[i].cur_un_pts;
            auto &cur_pts = trackerData[i].cur_pts;
            auto &ids = trackerData[i].ids;
            auto &pts_velocity = trackerData[i].pts_velocity;
            for (unsigned int j = 0; j < ids.size(); j++)
            {
                if (trackerData[i].track_cnt[j] > 1)
                {
                    int p_id = ids[j];
                    hash_ids[i].insert(p_id);
                    double x = un_pts[j].x;
                    double y = un_pts[j].y;
                    double z = 1;
                    feature_points->points.push_back(Vector3d(x, y, z));
                    feature_points->id_of_point.push_back(p_id * NUM_OF_CAM + i);
                    feature_points->u_of_point.push_back(cur_pts[j].x);
                    feature_points->v_of_point.push_back(cur_pts[j].y);
                    feature_points->velocity_x_of_point.push_back(pts_velocity[j].x);
                    feature_points->velocity_y_of_point.push_back(pts_velocity[j].y);
                }
            }
            //}
            // skip the first image; since no optical speed on frist image
            if (!init_pub)
            {...
            }
            else
            {
                m_buf.lock();
                feature_buf.push(feature_points);
                // cout << "5 PubImage t : " << fixed << feature_points->header
                //      << " feature_buf size: " << feature_buf.size() << endl;
                m_buf.unlock();
                con.notify_one();
            }
        }
    }
```



```
void System::PubFeatureData(double dStampSec,
                            std::vector<cv::Point2f> &feature_points)
{
    if (!init_feature)
    {...
    }

    if (first_image_flag)
    {...
    }
    // detect unstable camera stream
    if (dStampSec - last_image_time > 1.0 || dStampSec < last_image_time)
    {...
    }
    last_image_time = dStampSec;
    // frequency control

    // PUB_THIS_FRAME = true;
    if (round(1.0 * pub_count / (dStampSec - first_image_time)) <= FREQ)
    {...
    }
    else
    {...
    }

    TicToc t_r;
    trackerData[0].ReadFeature(feature_points, dStampSec);

    for (unsigned int i = 0;; i++)
    {...
    }
    // std::cout << "pub img is: " << PUB_THIS_FRAME << std::endl;
    if (PUB_THIS_FRAME)
    {
        pub_count++;
        shared_ptr<IMG_MSG> feature_points(new IMG_MSG());
        feature_points->header = dStampSec;
        vector<set<int>> hash_ids(NUM_OF_CAM);
        for (int i = 0; i < NUM_OF_CAM; i++)
        {
            auto &un_pts = trackerData[i].cur_un_pts;
            auto &cur_pts = trackerData[i].cur_pts;
            auto &ids = trackerData[i].ids;
            auto &pts_velocity = trackerData[i].pts_velocity;

            for (unsigned int j = 0; j < ids.size(); j++)
            {
                if (trackerData[i].track_cnt[j] > 1)
                {
                    int p_id = ids[j];
                    hash_ids[i].insert(p_id);
                    double x = un_pts[j].x;
                    double y = un_pts[j].y;
                    double z = 1;
                    feature_points->points.push_back(Vector3d(x, y, z));
                    feature_points->id_of_point.push_back(p_id * NUM_OF_CAM + i);
                    feature_points->u_of_point.push_back(cur_pts[j].x);
                    feature_points->v_of_point.push_back(cur_pts[j].y);
                    feature_points->velocity_x_of_point.push_back(pts_velocity[j].x);
                    feature_points->velocity_y_of_point.push_back(pts_velocity[j].y);
                }
            }
            //}
            // skip the first image; since no optical speed on frist image
            if (!init_pub)
            {...
            }
            else
            {
                m_buf.lock();
                feature_buf.push(feature_points);
                // cout << "5 PubImage t : " << fixed << feature_points->header
                //      << " feature_buf size: " << feature_buf.size() << endl;
                m_buf.unlock();
                con.notify_one();
```

run_simulation pub_Camera

深蓝学院 shenlanxueyuan.com

```cpp
void FeatureTracker::readImage(const cv::Mat &_img, double _cur_time)
{
    cv::Mat img;
    TicToc t_r;
    cur_time = _cur_time;

    if (EQUALIZE)
    {…
    }
    else
        img = _img;

    if (forw_img.empty())
    {…
    }
    else
    {…
    }

    forw_pts.clear();

    if (cur_pts.size() > 0)
    {
        TicToc t_o;
        vector<uchar> status;
        vector<float> err;
        cv::calcOpticalFlowPyrLK(cur_img, forw_img, cur_pts, forw_pts, status, err, cv::Size(21, 21), 3);

        for (int i = 0; i < int(forw_pts.size()); i++)
            if (status[i] && !inBorder(forw_pts[i]))
                status[i] = 0;
        reduceVector(prev_pts, status);
        reduceVector(cur_pts, status);
        reduceVector(forw_pts, status);
        reduceVector(ids, status);
        reduceVector(cur_un_pts, status);
        reduceVector(track_cnt, status);
        //ROS_DEBUG("temporal optical flow costs: %fms", t_o.toc());
    }

    for (auto &n : track_cnt)
        n++;
```

```cpp
if (PUB_THIS_FRAME)
{
    rejectWithF();
    //ROS_DEBUG("set mask begins");
    TicToc t_m;
    setMask();
    //ROS_DEBUG("set mask costs %fms", t_m.toc());

    //ROS_DEBUG("detect feature begins");
    TicToc t_t;
    int n_max_cnt = MAX_CNT - static_cast<int>(forw_pts.size());
    if (n_max_cnt > 0)
    {
        if (mask.empty())
            cout << "mask is empty " << endl;
        if (mask.type() != CV_8UC1)
            cout << "mask type wrong " << endl;
        if (mask.size() != forw_img.size())
            cout << "wrong size " << endl;
        cv::goodFeaturesToTrack(forw_img, n_pts, MAX_CNT - forw_pts.size(), 0.01, MIN_DIST, mask);
    }
    else
        n_pts.clear();
    //ROS_DEBUG("detect feature costs: %fms", t_t.toc());

    //ROS_DEBUG("add feature begins");
    TicToc t_a;
    addPoints();
    //ROS_DEBUG("selectFeature costs: %fms", t_a.toc());
}
prev_img = cur_img;
prev_pts = cur_pts;
prev_un_pts = cur_un_pts;
cur_img = forw_img;
cur_pts = forw_pts;
undistortedPoints();
prev_time = cur_time;
```

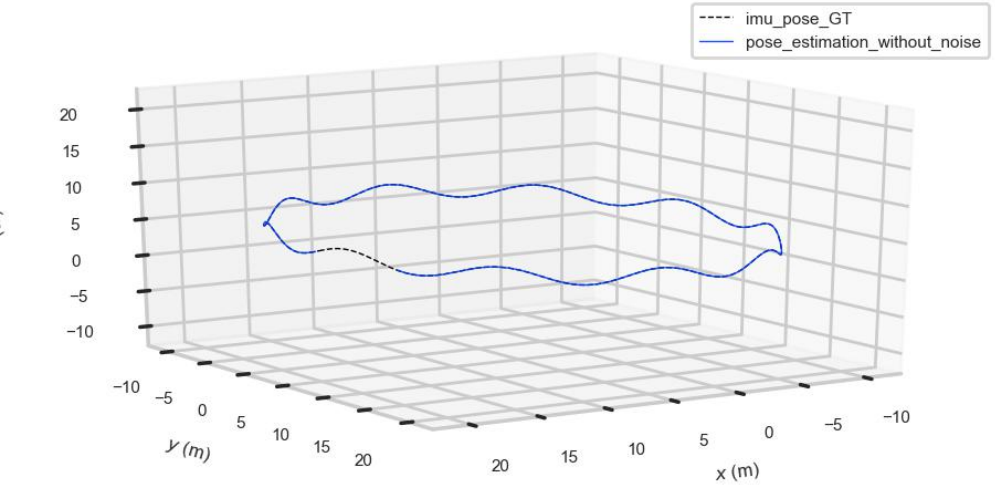
run_simulation pub_Camera

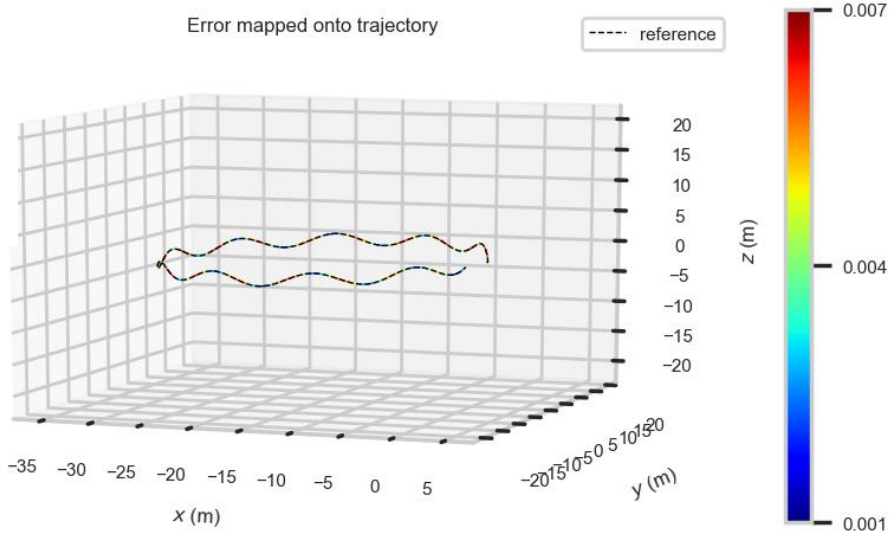
深蓝学院 shenlanxueyuan.com

```
void FeatureTracker::ReadFeature(
    const std::vector<cv::Point2f> &feature_points, double _cur_time)
{
    TicToc t_r;
    cur_time = _cur_time;

    forw_pts.clear();

    auto phnhole_cam = dynamic_cast<PinholeCamera *>(m_camera.get());
    PinholeCamera::Parameters parameters = phnhole_cam->getParameters();

    if (cur_pts.size() > 0)
    {
        for (auto feature : feature_points)
        {
            cv::Point2f pixel_feature;
            pixel_feature.x = parameters.fx() * feature.x + parameters.cx();
            pixel_feature.y = parameters.fy() * feature.y + parameters.cy();
            forw_pts.push_back(pixel_feature);
        }
        std::vector<uchar> status(feature_points.size(), 1);

        reduceVector(prev_pts, status);
        reduceVector(cur_pts, status);
        reduceVector(forw_pts, status);
        reduceVector(ids, status);
        reduceVector(cur_un_pts, status);
        reduceVector(track_cnt, status);
    }      You, a few seconds ago • Uncommitted changes

    for (auto &n : track_cnt)
        n++;
```

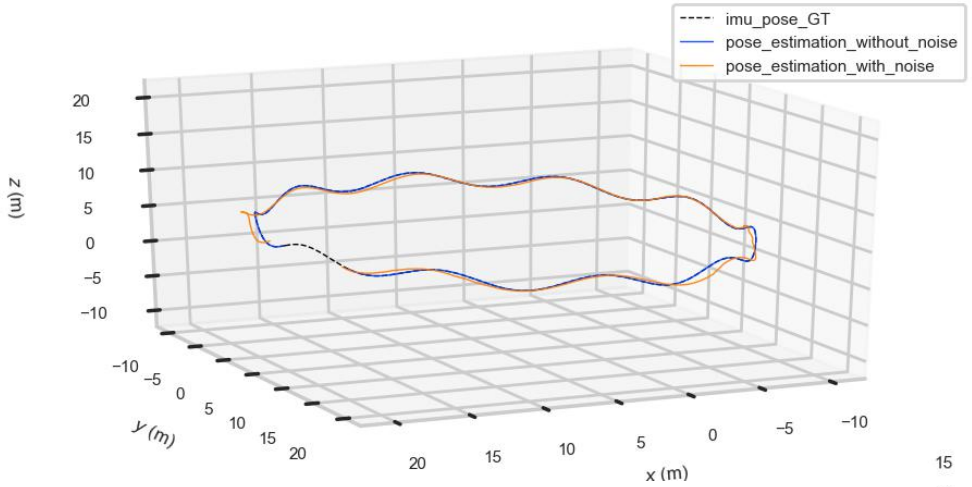
```
if (PUB_THIS_FRAME)
{
    TicToc t_m;
    // ROS_DEBUG("detect feature begins");
    int n_max_cnt = feature_points.size() - static_cast<int>(forw_pts.size());
    if (n_max_cnt > 0)
    {
        for (auto it = 0; it < feature_points.size(); it++)
        {
            cv::Point2f pixel_feature;
            pixel_feature.x =
                parameters.fx() * feature_points[it].x + parameters.cx();
            pixel_feature.y =
                parameters.fy() * feature_points[it].y + parameters.cy();
            n_pts.push_back(pixel_feature);
        }
    }
    else
    {
        n_pts.clear();
    }
    TicToc t_t;

    // ROS_DEBUG("add feature begins");
    TicToc t_a;
    addPoints();
    // ROS_DEBUG("selectFeature costs: %fms", t_a.toc());
}

prev_pts = cur_pts;
prev_un_pts = cur_un_pts;

cur_pts = forw_pts;
undistortedPoints();
prev_time = cur_time;
```

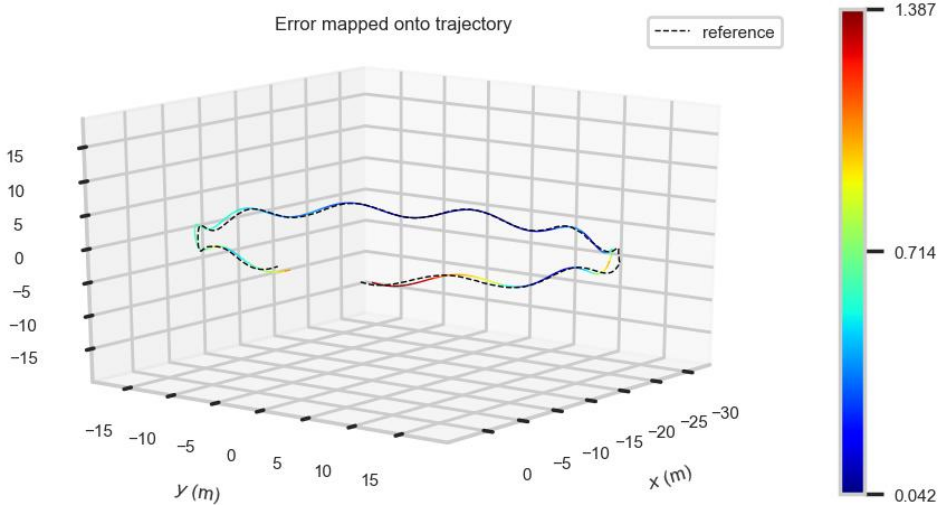
使用无噪声IMU测量数据





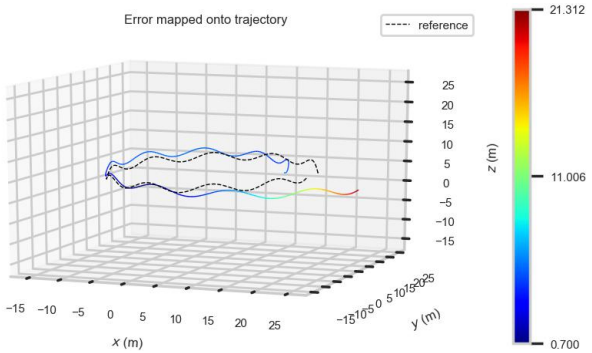
使用带噪声IMU测量数据



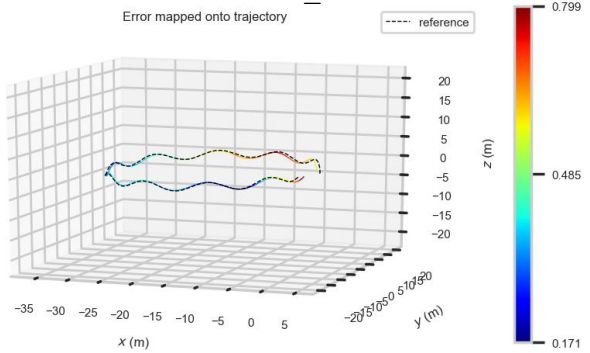


设定噪声的不同config参数

noise_0



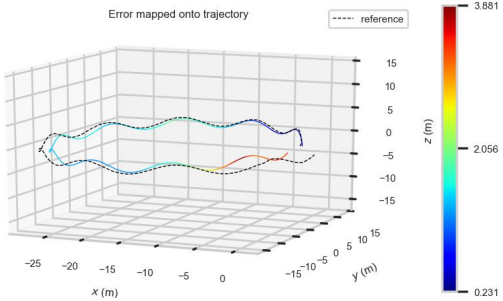
noise_0.5倍



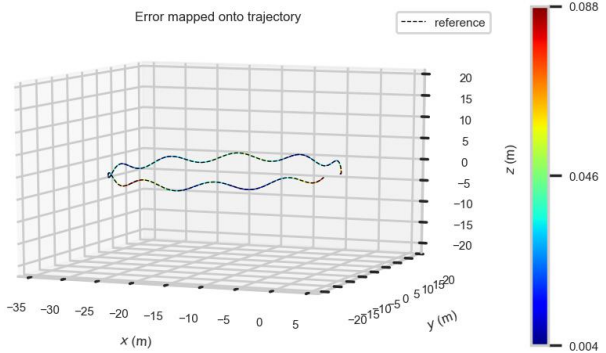
gyro_noise = 0.015
gyro_noise = 0.019

gyro_bias_sigma = 1.0e-5;
acc_bias_sigma = 0.0001;

noise_2倍



noise_10倍



深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening