Using OpenVINS to record a trajectory with a camera and IMU sensors

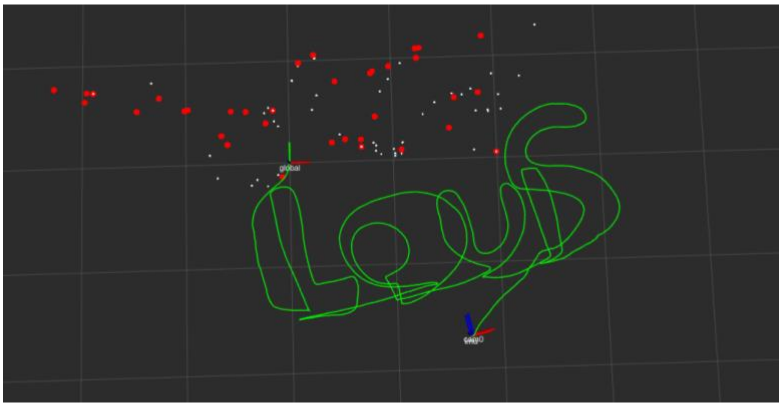


Figure 1. 3D Path of the D455 camera in Rviz

Summary

1 - Kalibr installation

1.1 - Dependencies

1.1.1 - System dependencies

```
me@myComputer:~$ sudo apt-get install-y \
  git wget autoconf automake nano \
  libeigen3-dev libboost-all-dev libsuitesparse-dev \
  doxygen libopencv-dev \
  libpoco-dev libtbb-dev libblas-dev liblapack-dev libv4l-dev
```

1.1.2 - Other dependencies

```
me@myComputer:~$ sudo apt-get install-y python3-dev python3-pip python3-scipy \
  python3-matplotlib ipython3 python3-wxgtk4.0 python3-tk python3-igraph python3-pyx
```

1.2 - Building the Kalibr project

1.2.1 - Create a new catkin workspace for kalibr

Create a **workspace** folder. This folder will contain all the workspaces you need. Then, create a new catkin workspace inside to build the Kalibr project and navigate into it.

```
me@myComputer:~$ mkdir workspace
me@myComputer:~$ cd ~/workspace
me@myComputer:~/workspace$ mkdir-p catkin_ws_kalibr/src
me@myComputer:~/workspace$ cd catkin_ws_kalibr
```

1.2.2 - Configurate the catkin workspace and clone the Kalibr project.

```
me@myComputer:~/workspace/catkin_ws_kalibr$ source /opt/ros/noetic/setup.bash
me@myComputer:~/workspace/catkin_ws_kalibr$ catkin init
me@myComputer:~/workspace/catkin_ws_kalibr$ catkin config--extend /opt/ros/noetic
me@myComputer:~/workspace/catkin_ws_kalibr$ catkin config--merge-devel
me@myComputer:~/workspace/catkin_ws_kalibr$ catkin config--cmake-args-DCMAKE_BUILD_TYPE=Release
me@myComputer:~/workspace/catkin_ws_kalibr$ cd src
me@myComputer:~/workspace/catkin_ws_kalibr/src$ git clone https://github.com/ethz-asl/kalibr.git
```

1.2.3 - Build the project

```
me@myComputer:~/workspace/catkin_ws_kalibr/src$ cd ..
me@myComputer:~/workspace/catkin_ws_kalibr$ catkin build-DCMAKE_BUILD_TYPE=Release-j4
```

2 - OpenVINS installation

2.1 - Dependencies

2.1.1 - System dependencies

Copy and paste this command line in the terminal:

```
me@myComputer:~$ sudo apt-get install libeigen3-dev libboost-all-dev libceres-dev
```

2.1.2 - ROS installation

Skip this part if you have already installed ROS on your computer.

```
me@myComputer:~$ sudo sh-c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release-sc) main" > /etc/apt/sources.list.d/ros-latest.list'

me@myComputer:~$ sudo apt-key adv--keyserver 'hkp://keyserver.ubuntu.com:80'--recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

me@myComputer:~$ sudo apt-get update

me@myComputer:~$ sudo apt-get install ros-noetic-desktop-full # replace noetic if using other distribution

me@myComputer:~$ sudo apt-get install python3-catkin-tools python3-osrf-pycommon # ubuntu 20.04

me@myComputer:~$ sudo apt-get install libeigen3-dev libboost-all-dev libceres-dev
```

2.2 - Building the OpenVINS project

```
me@myComputer:~$ cd ~/workspace/catkin_ws_kalibr/src/

me@myComputer:~/workspace/catkin_ws_kalibr/src$ git clone https://github.com/rpng/open_vins/

me@myComputer:~/workspace/catkin_ws_kalibr/src$ cd ..

me@myComputer:~/workspace/catkin_ws_kalibr$ catkin config--cmake-args-DCMAKE_BUILD_TYPE=Debug
```

The last step can take some time. If you are running this on the **iss-interns** computer of the Lab, add **-j1** at the end of the command. Otherwise, if your computer freezes during the build, delete the newly created **open_vins** folder and repeat the paragraph **2.2 - Building the OpenVINS project** then add **-j1** at the end of the command.

3 - Allan Variance ROS installation

This installation is easy. You just have to clone the project and re-build your workspace.

```
me@myComputer:~/workspace/catkin_ws_kalibr/src$ git clone https://github.com/ori-drs/allan_variance_ros.git

me@myComputer:~/workspace/catkin_ws_kalibr/src$ cd ..

me@myComputer:~/workspace/catkin_ws_kalibr $ catkin build
```

4 - Calibration of the Intel realsense D455

4.1 - April Grid

To determine the calibration parameters with Kalibr we need to record a bag where we will store information regarding the IMU of the camera and images of the required target in a variety of positions exiting all the degrees of liberty of the IMU (three rotations and three translations) independently and combined (movements and rotations in all directions). The target is a composition of unique squares that will be identify by the software to determine the real position of the points and the image position of them.

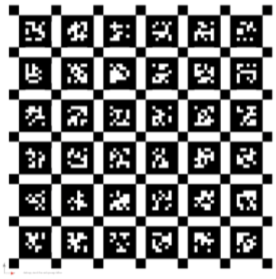


Figure 2. April Grid Augmented Reality Target

Before you start the calibration, measure with a ruler the side of 1 of the 16 tags (edge to edge) on the grid and note this value as ‘tagSize’. Then measure the spacing between two tags, divide it by the ‘tagSize’ and note this ratio as ‘tagSpacing’. (‘tagSpacing’ is the ratio of the ‘tagSize’ and the actual spacing …). If you print it in a A0 format you should get a tagSize of 0.088 (meters) and a tagSpacing of 0.3 (no unit).

4.2 - Calibrating the camera

This tutorial assumes that you previously installed the Official Intel realsense ROS wrapper : https://github.com/IntelRealSense/realsense-ros/tree/ros1-legacy. You can clone this package in your **~/workspace/catkin_ws_kalibr/src** folder.

4.2.1 - Record a ROS bag : /camera/color/image_raw

Launch the camera by running this command:

```
me@myComputer:~$ roslaunch realsense2_camera rs_camera.launch
```

You can now verify that the **/camera/color/image_raw** topic is published by running this command in a second terminal window:

```
me@myComputer:~$ rostopic list
```

If you see the **/camera/color/image_raw** topic in the list you can navigate to the folder of your choice and record a ROS bag. For this tutorial we will store the bag in a new folder. First, navigate to the **~/workspace/catkin_ws_kalibr**. Inside this workspace create a new folder named **tutorial** and three sub-folders called **static**, **static_imu** and **dynamic**. Then, navigate to your static folder and start recording.

```
me@myComputer:~/workspace/catkin_ws_kalibr/tutorial/static$ rosbag record-O static.bag
/camera/color/image_raw
```

Move the camera in all directions and all rotations while facing the April Grid. After around one minute you can stop the recording with Ctrl+C. the bag should appear in your **~/workspace/catkin_ws_kalibr/tutorial/static** folder.

Next, we need a configuration file with real measurements of the April Grid so the calibration program knows the genuine “size” of the environment that it sees. You can either download this file on the Kalibr wiki pages:

https://github.com/ethz-asl/kalibr/wiki/downloads or create your own file with Visual Studio Code, name it **"aprilgrid.yaml"**, then copy and paste the text below inside:

```
target_type: 'aprilgrid'        # gridtype
tagCols: 6                      # number of April tags
tagRows: 6                      # number of April tags
tagSize: 0.088                  # size of the April tag, edge to edge [m]
tagSpacing: 0.3                 # ratio of space between tags to tagSize
codeOffset: 0                   # code offset for the first tag in the aprilboard
```

Keep the **aprilgrid.yaml** file inside your **~/workspace/catkin_ws_kalibr/tutorial** folder (not in static, not in static_imu, not in dynamic).

4.2.2 - Camera data analysis

Run the **kalibr_calibrate_cameras** node with the following command:

```
me@myComputer:~/workspace/catkin_ws_kalibr/tutorial$ roslaunch kalibr kalibr_calibrate_cameras--target aprilgrid.yaml--models pinhole-radtan--topics /camera/color/image_raw--bag static.bag--bag-freq 30
```

Once the program stops running you get these nice graphs (below). You can see that the error measured was approximately one pixel. You also get a configuration file (**static-camchain.yaml**) that will be useful later.

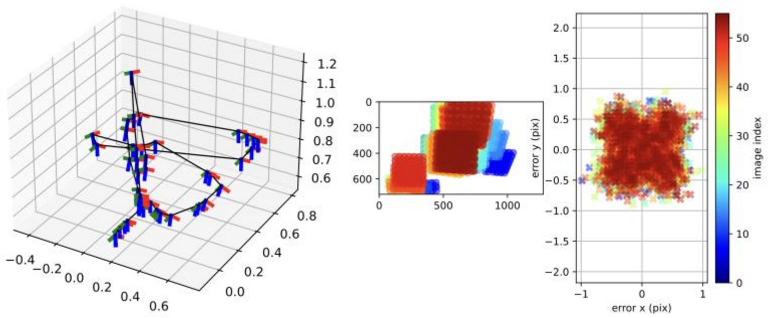


Figure 3. Pose estimation based on the camera images only (left). Error in pixels (right)

4.3 - Collecting the intrinsic noise parameters of the camera

4.3.1 - my_camera.launch

Navigate inside the **realsense2_camera** package. Go to **realsense2_camera/launch.** Duplicate the **rs_camera.launch** file and rename the copy "**my_camera.launch**". Inside your custom **my_camera.launch** file look for the following lines and set the default parameters to the following values:

```
<arg name="gyro_fps" default="-1"/>                  # set default to "200"
<arg name="accel_fps" default="-1"/>                 # set default to "63"
<arg name="enable_gyro" default="false"/>            # set default to "true"
<arg name="enable_gyro" default="false"/>            # set default to "true"
<arg name="enable_pointcloud" default="false"/>      # set default to "true"
<arg name="unite_imu_method" default=""/>            # set default to "linear_interpolation"
```

4.3.2 - Record a ROS bag : /camera/imu (20 hours)

This part requires 20 HOURS. Launch the realsense camera but this time with your custom launch file:

```
me@myComputer:~$ roslaunch realsense2_camera my_camera.launch
```

You can now verify if the **/camera/imu** topic is published by running this command in a second terminal window:

```
me@myComputer:~ $ rostopic list
```

If you see the **/camera/imu** topic in the list, you can navigate to the folder of your choice and record a ROS bag. For this tutorial we will store the bag in our **~/workspace/catkin_ws_kalibr/static_imu** folder. Now, place the camera in a place where it won't be moved by anybody. It should stay still for 20 hours. You can record the bag with the command below. **DON'T RECORD the /camera/color/image_raw topic**. Your bag file would be way too large:

```
me@myComputer:~/workspace/catkin_ws_kalibr/tutorial/static_imu$ rosbag record-O static_twenty.bag /camera/imu
```

The next day, you can stop the recording with Ctrl+C. The bag should appear in your **~/workspace/catkin_ws_kalibr/static_imu** folder.

4.3.3 - Collect the IMU noise data

Now that you have a ROS bag containing 20 hours of IMU data, check the information of the bag by running the following command:

```
me@myComputer:~/workspace/catkin_ws_kalibr/tutorial/static_imu$ rosbag info static_twenty.bag
```

```
louis@louis-All-Series:~/calibration_bags$ rosbag info static_twenty.bag
path:        static_twenty.bag
version:     2.0
duration:    20hr 37:21s (74241s)
start:       Dec 17 2024 16:30:22.38 (1734471022.38)
end:         Dec 18 2024 13:07:44.20 (1734545264.20)
size:        5.4 GB
messages:    14812006
compression: none [7194/7194 chunks]
types:       sensor_msgs/Imu [6a62c6daae103f4ff57a132d6f95cec2]
topics:      /camera/imu   14812006 msgs    : sensor_msgs/Imu
```

Figure 4. Information of the ROS bag

It takes some time to show the results, don't worry and wait. Once the information is showing up you can create a new file called **config.yaml** (using Visual Studio Code for example). To know which imu_rate and which measure_rate was used, divide the number of messages by the duration (in seconds). (Here, 14812006 divided by 74241 is approximately 200 Hz).

```
! config.yaml ×
tutorial > static_imu > ! config.yaml
1  imu_topic: "/camera/imu"
2  imu_rate: 200
3  measure_rate: 200
4  sequence_time: 74241
```

Figure 5. New config file

Since you must have gone to sleep in the meantime, source your workspace again:

```
me@myComputer:~/workspace/catkin_ws_kalibr$ source devel/setup.bash
```

You can extract the information into a csv file.

```
me@myComputer:~/workspace/catkin_ws_kalibr$ cd tutorial

me@myComputer:~/workspace/catkin_ws_kalibr/tutorial$ rosrun allan_variance_ros allan_variance static_imu/ static_imu/config.yaml
```

If you check your static_imu folder again, the **allan_variance.csv** file is created. Run the following command to analyze your data and collect important noise parameters.

```
me@myComputer:~/workspace/catkin_ws_kalibr/tutorial$ rosrun allan_variance_ros analysis.py –data static_imu/allan_variance.csv –config static_imu/config.yaml
```

You get the acceleration graph, the gyroscope graph and the **imu.yaml** file. **NEVER DELETE THIS IMPORTANT FILE** unless you want to start the 20 hours recording again.

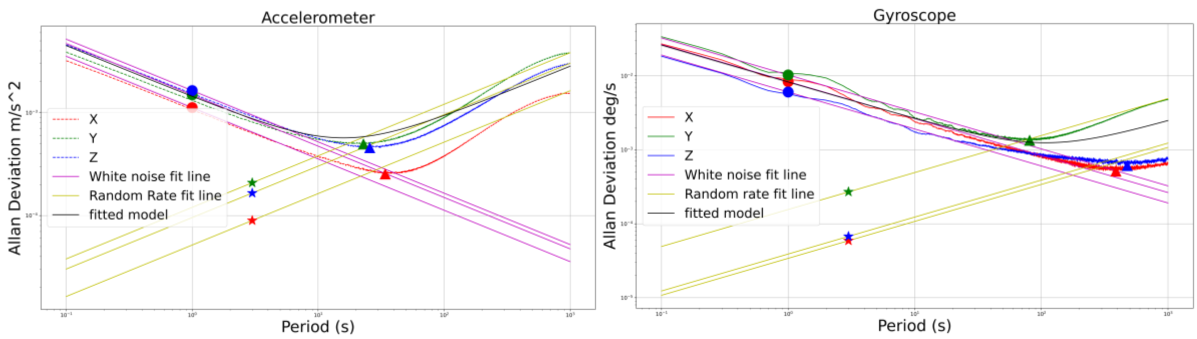


Figure 6. Allan deviation for both Accelerometer and Gyroscope

These graphs show the process of IMU noise recovery. The optimal values where derivative cancels are stored in the **imu.yaml** file.

4.4 - Calibrating the IMU

4.4.1 - Record a ROS bag : /camera/color/image_raw + /camera/imu

This is the last bag you will need to record. Use your custom launch file to record the IMU and camera data:

```
me@myComputer:~$ roslaunch realsense2_camera my_camera.launch
```

You can now verify that the **/camera/color/image_raw** and **/camera/imu** topics are being published by running this command in a second terminal window:

```
me@myComputer:~ $ rostopic list
```

If you see the **/camera/color/image_raw** and the **/camera/imu** topic in the list, you can navigate to the folder of your choice and record a new ROS bag. For this tutorial we will store the bag in the **~/workspace/catkin_ws_kalibr/tutorial/dynamic** folder.

```
me@myComputer:~/workspace/catkin_ws_kalibr/tutorial/dynamic$ rosbag record-O dynamic.bag /camera/color/image_raw /camera/imu
```

Move the camera in all directions and all rotations while facing the April Grid. After around one minute you can stop the recording with Ctrl+C. the bag should appear in your **~/workspace/catkin_ws_kalibr/tutorial/dynamic** folder.

4.4.2 - IMU Data analysis

Now run the the **kalibr_calibrate_imu_camera** node:

```
me@myComputer:~/workspace/catkin_ws_kalibr/tutorial/dynamic$ cd ..

me@myComputer:~/workspace/catkin_ws_kalibr/tutorial$ rosrun kalibr kalibr_calibrate_imu_camera--imu-models scale-misalignment--reprojection-sigma 1.0--target aprilgrid.yaml--imu static_imu/imu.yaml--cams static/static-camchain.yaml--bag dynamic/dynamic.bag--show-extraction
```

You will get multiple graphs. The most important one is the IMU sample rate graph as it indicates whether the gap between two IMU readings is constant on average.

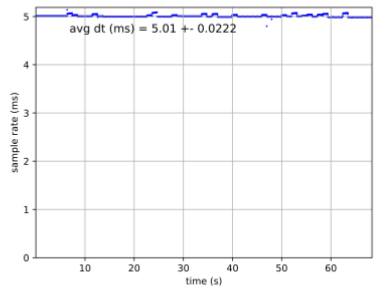


Figure 7. The IMU sample rate has a low variance (± 0.22 ms)

As for the other graphs, ensure that the error values are bounded by the 3-sigma bounds (dotted-lines).

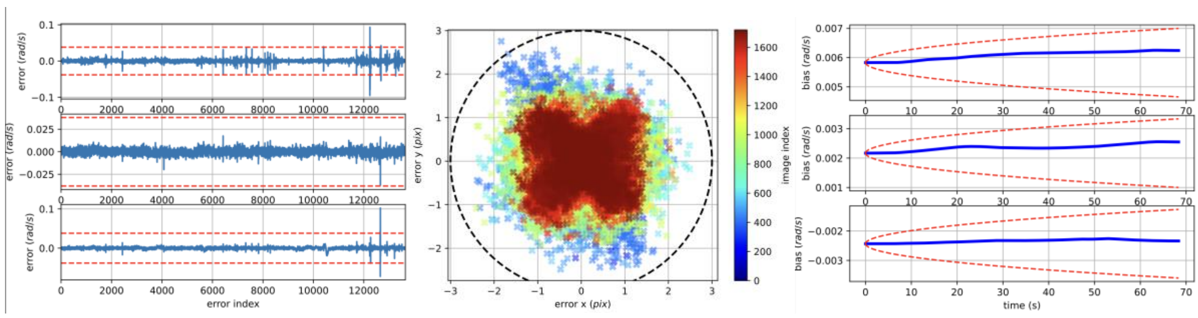


Figure 8. Angular velocities errors (left), Reprojection error (center), Estimated gyroscope bias (right)

5 - Live State Estimation

5.1 - OpenVINS configuration

5.1.1 - Collecting the intrinsic camera parameters

In your files manager, navigate to: **~/workspace/catkin_ws_kalibr/src/open_vins/config**. Copy the **rs_d455** folder and paste it into your tutorial folder. Rename it **"rs_custom"** to make sure you are not overwriting the original one. You can see that there are two files that are similar in **rs_custom** and in the **dynamic** folder:

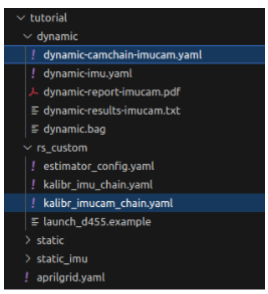


Figure 9. Similar files in the dynamic and rs_custom folders.

Comment the content of the **kalibr_imucam_chain.yaml** file. Copy the content of your new **dynamic-camchain-imucam.yaml** file below the commented code of the **kalibr_imucam_chain.yaml** file.

```yaml
tutorial > rs_custom > ! kalibr_imucam_chain.yaml
%YAML:1.0

# OLD CODE

# cam0:
#   T_cam_imu:
#     - [0.9999654398038452, 0.007342326779113337, -0.003899927610975742, -0.027534314618518095]
#     - [-0.0073452195116216765, 0.9999727585590525, -0.0007279355223411334, -0.0030587146933711722]
#     - [0.0038944766308488753, 0.0007565561891287445, 0.9999921303062861, -0.023605118842939803]
#     - [0.0, 0.0, 0.0, 1.0]
#   cam_overlaps: []
#   camera_model: pinhole
#   distortion_coeffs: [-0.045761895748285604, 0.03423951132164367, -0.00040139057556727315, 0.000431371425853453]
#   distortion_model: radtan
#   intrinsics: [416.85223429743274, 414.92069080087543, 421.02459311003213, 237.76180565241077]
#   resolution: [848, 480]
#   rostopic: /camera/color/image_raw
#   timeshift_cam_imu: 0.002524377913673846

# NEW CODE

cam0:
  T_cam_imu:
  - [0.9999997215713007, 0.0005696471863466849, -0.0004820367253875244, -0.02937369102068097]
  - [-0.0005760084422725549, 0.9999113715476651, -0.013301024920740239, -0.008189344562224143]
  - [0.00047441711179695774, 0.013301298874576477, 0.9999114212652307, -0.018388178186448327]
  - [0.0, 0.0, 0.0, 1.0]
  cam_overlaps: []
  camera_model: pinhole
  distortion_coeffs: [-0.04204473720425738, 0.031585000393194715, 0.0009160841048604084, 1.844179329865269e-05]
  distortion_model: radtan
  intrinsics: [637.005792773229, 636.3132501380801, 637.362819026199, 373.7050200685128]
  resolution: [1280, 720]
  rostopic: /camera/color/image_raw
  timeshift_cam_imu: 0.00041578881970301233
```

Figure 10. Replace the content of the **kalibr_imucam_chain.yaml** file with the content of the new **dynamic-camchain-imucam.yaml** file

5.1.2 - Collecting the intrinsic IMU parameters

Edit the **imu.yaml** file stored in the **~/workspace/catkin_ws_kalibr/tutorial/static_imu** folder. Multiply the accelerometer_noise_density and gyroscope_noise_density parameters by 5. Multiply the accelerometer_random_walk and gyroscope_random_walk parameters by 10. This enables a safe margin error. Comment the old values.

```yaml
tutorial > static_imu > ! imu.yaml
# #Accelerometer
# accelerometer_noise_density: 0.0016396831257351416
# accelerometer_random_walk: 0.00020807413742924183

# #Gyroscope
# gyroscope_noise_density: 0.0001793769055760414
# gyroscope_random_walk: 4.706667292550046e-06

# rostopic: '/camera/imu'
# update_rate: 200

#from 20 hour static imu dataset
# white x5
# random walk x10

#Accelerometer
accelerometer_noise_density: 0.00819841562
accelerometer_random_walk: 0.00208074137

#Gyroscope
gyroscope_noise_density: 0.00089688452
gyroscope_random_walk: 0.00004706667

rostopic: '/camera/imu'
update_rate: 200
```

Figure 11. In your **imu.yaml** file, multiply the noise by 5 and the random walk by 10.

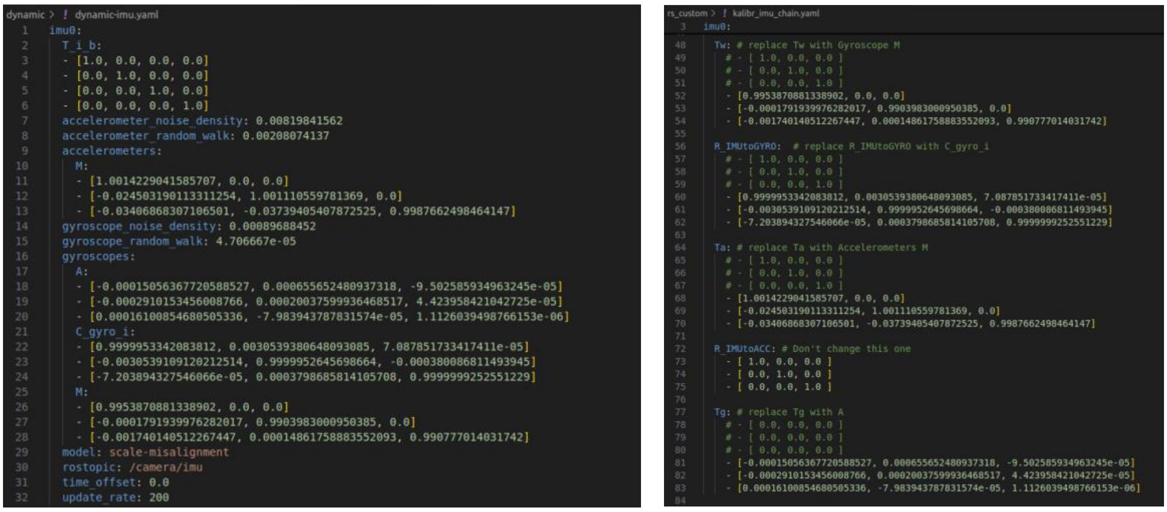
Locate the **kalibr_imu_chain.yaml** file in your **~/workspace/catkin_ws_kalibr/tutorial/rs_custom** folder. Paste the inflated results inside. Comment the old values.

```yaml
tutorial > rs_custom > ! kalibr_imu_chain.yaml
%YAML:1.0

imu0:
  T_i_b:
    - [1.0, 0.0, 0.0, 0.0]
    - [0.0, 1.0, 0.0, 0.0]
    - [0.0, 0.0, 1.0, 0.0]
    - [0.0, 0.0, 0.0, 1.0]

  # ///////////////// OLD //////////////////
  # Inflated values (to account for unmodelled effects)
  #  - white noise multiplied by 2
  #  - bias random walk multiplied by 10
  # accelerometer_noise_density: 0.00207649074
  # accelerometer_random_walk: 0.00041327852
  # gyroscope_noise_density: 0.00020544166
  # gyroscope_random_walk: 0.00001118622
  # rostopic: /camera/imu
  # time_offset: 0.0
  # update_rate: 400

  # ///////////////// NEW //////////////////
  #from 20 hour static imu dataset
  # white x5
  # random walk x10

  #Accelerometer
  accelerometer_noise_density: 0.00819841562
  accelerometer_random_walk: 0.00208074137

  #Gyroscope
  gyroscope_noise_density: 0.00089688452
  gyroscope_random_walk: 0.00004706667

  rostopic: '/camera/imu'
  update_rate: 200
```

Figure 12. Comment the old values and paste your inflated results into the **kalibr_imu_chain.yaml**

Edit the **/tutorial/rs_custom/kalibr_imu_chain.yaml** file.

- Replace the "Tw" matrix with the Gyroscope "M" matrix from the **/tutorial/dynamic/dynamic-imu.yaml** file.
- Replace the "R_IMUtoGYRO" matrix with the "C_gyro_i" matrix from the **/tutorial/dynamic/dynamic-imu.yaml** file.
- Replace the "Ta" matrix with the Accelerometer "M" matrix from the **/tutorial/dynamic/dynamic-imu.yaml** file.
- Don't change the "R_IMUtoACC" matrix.
- Replace the "Tg" matrix with the "A" matrix from the **/tutorial/dynamic/dynamic-imu.yaml** file.



dynamic > dynamic-imu.yaml

```yaml
imu0:
  T_i_b:
  - [1.0, 0.0, 0.0, 0.0]
  - [0.0, 1.0, 0.0, 0.0]
  - [0.0, 0.0, 1.0, 0.0]
  - [0.0, 0.0, 0.0, 1.0]
  accelerometer_noise_density: 0.00819841562
  accelerometer_random_walk: 0.00208074137
  accelerometers:
    M:
    - [1.0014229041585707, 0.0, 0.0]
    - [-0.024503190113311254, 1.001110559781369, 0.0]
    - [-0.03406868307106501, -0.03739405407872525, 0.9987662498464147]
  gyroscope_noise_density: 0.00089688452
  gyroscope_random_walk: 4.706667e-05
  gyroscopes:
    A:
    - [-0.00015056367720588527, 0.000655652480937318, -9.502585934963245e-05]
    - [-0.0002910153456008766, 0.00020037599936468517, 4.423958421042725e-05]
    - [0.00016100854680505336, -7.983943787831574e-05, 1.1126039498766153e-06]
    C_gyro_i:
    - [0.9999953342083812, 0.0030539380648093085, 7.087851733417411e-05]
    - [-0.0030539109120212514, 0.9999952645698664, -0.000380086811493945]
    - [-7.203894327546066e-05, 0.0003798685814105708, 0.9999999252551229]
    M:
    - [0.9953870881338902, 0.0, 0.0]
    - [-0.0001791939976282017, 0.9903983000950385, 0.0]
    - [-0.001740140512267447, 0.00014861758883552093, 0.990777014031742]
  model: scale-misalignment
  rostopic: /camera/imu
  time_offset: 0.0
  update_rate: 200
```

rs_custom > kalibr_imu_chain.yaml

```yaml
imu0:
  Tw: # replace Tw with Gyroscope M
    # - [ 1.0, 0.0, 0.0 ]
    # - [ 0.0, 1.0, 0.0 ]
    # - [ 0.0, 0.0, 1.0 ]
    - [0.9953870881338902, 0.0, 0.0]
    - [-0.0001791939976282017, 0.9903983000950385, 0.0]
    - [-0.001740140512267447, 0.00014861758883552093, 0.990777014031742]

  R_IMUtoGYRO:  # replace R_IMUtoGYRO with C_gyro_i
    # - [ 1.0, 0.0, 0.0 ]
    # - [ 0.0, 1.0, 0.0 ]
    # - [ 0.0, 0.0, 1.0 ]
    - [0.9999953342083812, 0.0030539380648093085, 7.087851733417411e-05]
    - [-0.0030539109120212514, 0.9999952645698664, -0.000380086811493945]
    - [-7.203894327546066e-05, 0.0003798685814105708, 0.9999999252551229]

  Ta: # replace Ta with Accelerometers M
    # - [ 1.0, 0.0, 0.0 ]
    # - [ 0.0, 1.0, 0.0 ]
    # - [ 0.0, 0.0, 1.0 ]
    - [1.0014229041585707, 0.0, 0.0]
    - [-0.024503190113311254, 1.001110559781369, 0.0]
    - [-0.03406868307106501, -0.03739405407872525, 0.9987662498464147]

  R_IMUtoACC: # Don't change this one
    - [ 1.0, 0.0, 0.0 ]
    - [ 0.0, 1.0, 0.0 ]
    - [ 0.0, 0.0, 1.0 ]

  Tg: # replace Tg with A
    # - [ 0.0, 0.0, 0.0 ]
    # - [ 0.0, 0.0, 0.0 ]
    # - [ 0.0, 0.0, 0.0 ]
    - [-0.00015056367720588527, 0.000655652480937318, -9.502585934963245e-05]
    - [-0.0002910153456008766, 0.00020037599936468517, 4.423958421042725e-05]
    - [0.00016100854680505336, -7.983943787831574e-05, 1.1126039498766153e-06]
```

Figure 13. **dynamic-imu.yaml** (left) **Kalibr_imu_chain.yaml** (right)

5.2 - Visualize the camera path in Rviz

5.2.1 - OpenVINS MSCKF launch modifications

Navigate to the **~/workspace/catkin_ws_kalibr/src/open_vins/ov_msckf/launch/** folder and edit the **subscribe.launch** file. Look for the "max_cameras", "dosave" and "dotime" parameters:

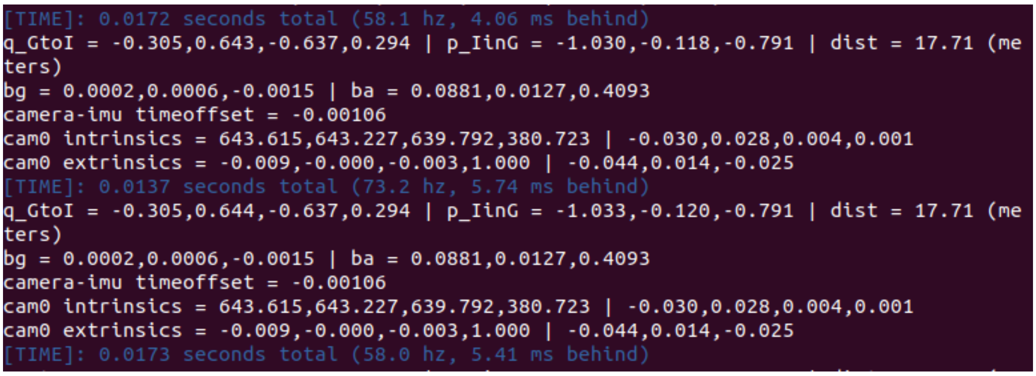
```
<arg name="max_cameras" default="2"/>        # set default to "1"
<arg name="dosave" default="false"/>         # set default to "true
<arg name="dotime" default="false"/>         # set default to "true
```

5.2.2 - Run the MSCKF subscribe node

In three separate terminal windows run the following commands:

```
me@myComputer:~/workspace/catkin_ws_kalibr$ rviz-d src/open_vins/ov_msckf/launch/display.rviz

me@myComputer:~/workspace/catkin_ws_kalibr$ roslaunch realsense2_camera my_camera.launch

me@myComputer:~/workspace/catkin_ws_kalibr $ roslaunch ov_msckf subscribe.launch
config_path:=/home/[insert your
username]/workspace/catkin_ws_kalibr/tutorial/rs_custom/estimator_config.yaml
```

Start moving the camera slowly. The warning messages should disappear, and this kind of information should be displayed instead.



```
[TIME]: 0.0172 seconds total (58.1 hz, 4.06 ms behind)
q_GtoI = -0.305,0.643,-0.637,0.294 | p_IinG = -1.030,-0.118,-0.791 | dist = 17.71 (me
ters)
bg = 0.0002,0.0006,-0.0015 | ba = 0.0881,0.0127,0.4093
camera-imu timeoffset = -0.00106
cam0 intrinsics = 643.615,643.227,639.792,380.723 | -0.030,0.028,0.004,0.001
cam0 extrinsics = -0.009,-0.000,-0.003,1.000 | -0.044,0.014,-0.025
[TIME]: 0.0137 seconds total (73.2 hz, 5.74 ms behind)
q_GtoI = -0.305,0.644,-0.637,0.294 | p_IinG = -1.033,-0.120,-0.791 | dist = 17.71 (me
ters)
bg = 0.0002,0.0006,-0.0015 | ba = 0.0881,0.0127,0.4093
camera-imu timeoffset = -0.00106
cam0 intrinsics = 643.615,643.227,639.792,380.723 | -0.030,0.028,0.004,0.001
cam0 extrinsics = -0.009,-0.000,-0.003,1.000 | -0.044,0.014,-0.025
[TIME]: 0.0173 seconds total (58.0 hz, 5.41 ms behind)
```

Figure 14. Console information messages generated by the MSCKF **subscribe.launch** file

If you can't make it work, there is still hope! Navigate to **~/workspace/catkin_ws_kalibr/tutorial/rs_custom** and tweak the IMU threshold parameter a little bit. Edit the **estimator_config.yaml** file and look for the init_imu_thresh parameter:

Init_imu_thresh: 1.5 # threshold for variance of the accelerometer to detect a "jerk" in motion

A value as low as 0.1 should make it work although a higher value will increase the stability. AVOID fast rotations. However, you can move forward or backward quickly. If your camera drifts away in the visualization panel, try enabling the "zero velocity update parameters". (This made a really big difference for me).

try_zupt: false # set this value to true

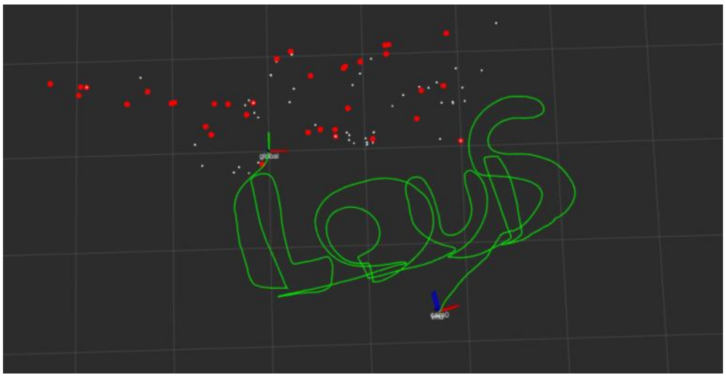


Figure 15. Visualization of the 3D path taken by the D455 camera in Rviz.

If you think the problem comes from the computing power of your computer, try different values of fast_threshold.

fast_threshold: 30 # threshold for fast extraction (warning: lower threshs can be expensive)

5.3 - Collect the positions and plot the results

Here is an example of a python script you can use to convert the ROS messages into csv files. Note that the exact content of the bag are temporary versions of the actual ROS messages. When you print the message type you get <class ‘tmpb_v70ssb._geometry_msgs__PoseWithCovarianceStamped’>.

```python
bag_to_csv.py > ...
import rosbag
import pandas as pd

def rosbag_to_csv_with_covariance(bag_path, topic_name, csv_path):
    data = []  # List to store all the message data

    with rosbag.Bag(bag_path, 'r') as bag:
        for topic, msg, t in bag.read_messages(topics=[topic_name]):
            try:
                data.append({
                    "seq": msg.header.seq,
                    "secs": msg.header.stamp.secs,
                    "nsecs": msg.header.stamp.nsecs,
                    "frame_id": msg.header.frame_id,
                    "position_x": msg.pose.pose.position.x,
                    "position_y": msg.pose.pose.position.y,
                    "position_z": msg.pose.pose.position.z,
                    "orientation_x": msg.pose.pose.orientation.x,
                    "orientation_y": msg.pose.pose.orientation.y,
                    "orientation_z": msg.pose.pose.orientation.z,
                    "orientation_w": msg.pose.pose.orientation.w,
                    **{f"cov_{i}": msg.pose.covariance[i] for i in range(36)}  # Flattened covariance fields
                })
            except AttributeError as e:
                print(f"Skipping message due to missing fields: {e}")

    # Convert the list of dictionaries to a pandas DataFrame
    df = pd.DataFrame(data)

    # Write the DataFrame to a CSV file
    df.to_csv(csv_path, index=False)
    print(f"Data successfully written to {csv_path}")

if __name__ == "__main__":
    import sys
    if len(sys.argv) != 4:
        print("Usage: python script.py <rosbag_path> <topic_name> <csv_path>")
    else:
        rosbag_path = sys.argv[1]
        topic_name = sys.argv[2]
        csv_path = sys.argv[3]
        rosbag_to_csv_with_covariance(rosbag_path, topic_name, csv_path)
```

Figure 16. Python script to write csv files from a ROS bag

With very simple manipulations, using pandas and matplotlib, you can plot the trajectory and positions over “time” for example.

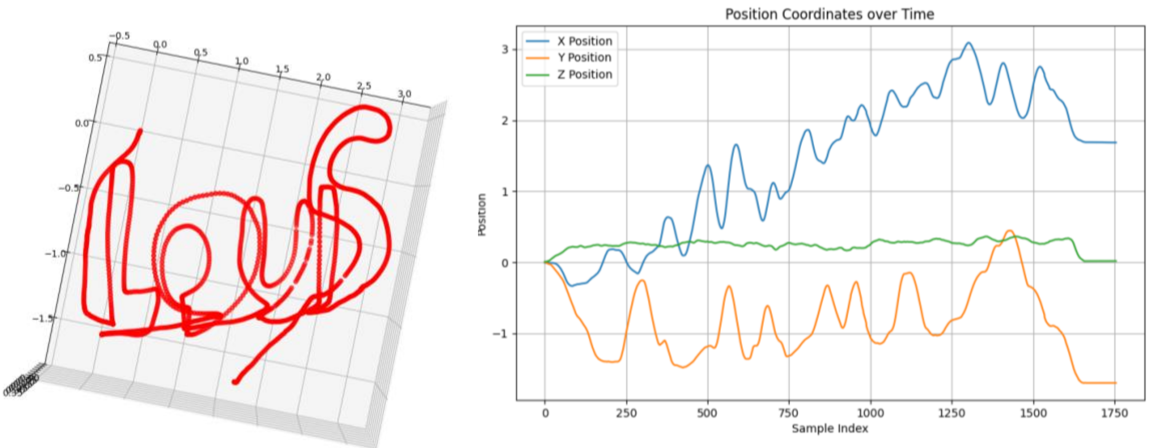


Figure 17. Trajectory and position plotted via matplotlib and a csv file