# GICI-LIB

# GNSS/INS/Camera Integrated Navigation Library

# User Manual

Cheng Chi, Jiahui Liu, Yulong Sun, Zihao Zhang

**Guidance, Navigation, and Control Laboratory**
**School of Aeronautics and Astronautics**
**Shanghai Jiao Tong University**

*June 2023*

# Contents

# Chapter 1

# Introduction

## 1.1 Overview

GICI-LIB is an open-source software package for Global Navigation Satellite System (GNSS), Inertial Navigation System (INS), and Camera integrated navigation. The features of GICI-LIB are:

(a) It is built under the Factor Graph Optimization (FGO) framework. It contains most of the possible GNSS loose and tight integration factors, INS factors, visual factors, and motion constraints, together with reliable initialization, measurement sparsification, and outlier rejection algorithms. The GNSS formulations are implemented towards four constellations and full frequencies.

(b) For ease of use, the software is developed under object-oriented programming features, and the graph is designed to enable the flexible addition of sensors. By simple instantiation, one can easily form any kind of multi-sensor fusion algorithm with considerable robustness.

(c) It supports multiple algorithms, including GNSS Single Point Positioning (SPP), Real-Time Differential (RTD), Single-Differenced GNSS (SDGNSS), Real-Time Kinematic (RTK), Precise Point Positioning (PPP), SPP-based loosely coupled (LC) and tightly coupled (TC) GNSS/Inertial Navigation System (GINS), SPP-based Solution/Raw/Raw (SRR) and Raw/Raw/Raw (RRR) GNSS/Visual/Inertial Navigation System (GVINS), RTK-based LC GINS, TC GINS, SSR GVINS, and RRR GVINS. Moreover, other integration algorithms can be instantiated by users.

(d) It supports multiple I/O ports, including serial, TCP/IP, NTRIP, V4L2, file, and ROS topics.

(e) It supports multiple message de/encoders, including RTCM2, RTCM3, Ublox raw, Septentrio raw, Novatel raw, Tersus raw, NMEA, DCB-file, ATX-file for GNSS, image-pack, image-v4l2 for image, and IMU-pack for IMU.

(f) It supports multiple stream and multi-algorithm processing. No maximum quantity is limited.

## 1.2 License

The GICI-LIB software package is distributed under GPL v3 license. Users are freedom to modify and distribute the software as they see fit, provided that they adhere to the terms and conditions set forth in the license. This includes the ability to incorporate or use GICI-LIB with other software, whether for non-commercial or commercial purposes. However, any modifications or derivative works must also be distributed under the GPL v3 license, ensuring that the software remains free and accessible to all users.

## 1.3 Acknowledgement

Many of the GNSS tools, I/O handlers, and message de/encoders are inherited from RTKLIB [1]. We extend our sincere gratitude to T. Takasu for generously providing such excellent software to the GNSS community. The software and its accompanying manual are valuable resources for those seeking to gain proficiency in GNSS.

The basic FGO management and the visual and IMU factors are partly inherited from OKVIS [2]. The feature handler is partly inherited from SVO 2.0 [3]. We also extend our thanks to the authors of these two software programs. Their implementations are very explicit and deeply influenced our programming style.

# Chapter 2

# Instructions

## 2.1 Installation

GICI supports two build modes: normal build or ROS build.

The normal build enables:

(a) Most of the streamer I/Os, including serial, TCP/IP server, TCP/IP client, Ntrip server, Ntrip client, V4L2, and file.

(b) All the formator decoder and encoders, including RTCM2, RTCM3, Ublox raw, Septentrio raw, Tersus raw, NMEA, DCB file, ATX file, V4L2 image pack, GICI image pack, and GICI IMU pack.

(c) All the estimators, including SPP, SDGNSS, DGNSS, RTK, PPP, SPP-based LC GINS, TC GINS, SRR GVINS, RRR GVINS, RTK-based LC GINS, TC GINS, SRR GVINS, RRR GVINS.

(d) Real-time or replay stream flow between the above modules.

The ROS build additionally enables:

(a) ROS stream I/Os that handles ROS topic advertising and subscribing, including sensor_msgs::Image, sensor_msgs::Imu, geometry_msgs::PoseStamped, geometry_msgs::PoseWithCovarianceStamped, nav_msgs::Odometry, visualization_msgs::Marker, and nav_msgs::Path.

(b) Real-time or replay stream flow between ROS and common GICI modules.

### 2.1.1 Requirements

- Ubuntu

  We are developing our code on Ubuntu 20.04, and tested on Ubuntu 18.04 and Ubuntu 22.04. We recommend you to use the same or similar environment if you do not familiar with cross-compiling.

- Eigen 3.3 or later. REQUIRED.

  Eigen is a C++ template library for linear algebra. You can find the releases on Eigen.

- OpenCV 4.2.0 or later. REQUIRED.

  OpenCV is a computer vision library. You can find the releases on Opencv.

- Yaml-cpp 0.6.0 or later. REQUIRED.

  Yaml-cpp is a decoder and encoder for YAML formats. We use YAML file to configure our workflow. You can find the releases on yaml-cpp.

- Glog 0.6.0 or later. REQUIRED.

  Glog is a logging control library. You can find the releases on Glog. You should install Glog together with Gflags. We suggest you install Glog from source code, rather than apt-get. Because installing from apt-get may make GICI fail to find the Glog library during compiling.

- Ceres-Solver 2.1.0 or later. REQUIRED.

  Ceres-Solver is a nonlinear optimization library. You can find the releases on Ceres-Solver.

- ROS. OPTIONAL.

  ROS is a library for robot applications. We provide a ROS wrapper to enable GICI handling some ROS messages. If you want to build GICI with ROS, you should install ROS. You can find the instrunctions on ROS.

### 2.1.2 Normal Build

```
cd <gici-root-directory>
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make -j8
```

Now you can run GICI via

```
./gici_main <gici-config-file>
```

### 2.1.3 Build with ROS

```
cd <gici-root-directory>/ros_wrapper
catkin_make -DCMAKE_BUILD_TYPE=Release
source ./devel/setup.bash
```

Now you can run GICI ROS wrapper via

```
rosrun gici_ros gici_ros_main <gici-config-file>
```

## 2.2 Configuration

GICI supports multithread streaming, de/encoding, and estimating. One can specify an unlimited number of nodes with our YAML configuration file. Here we will introduce how to configure a GICI workflow.

### 2.2.1 Structure of the Configuration File

GICI uses YAML file (.yaml) as configuration file. All sentences should meet the YAML format. There are several level of nodes defined in the configuration file. For the first-level, there are three kinds of nodes: stream, estimate, and logging, see below:

```
stream:
  # Defines all the streamer nodes and formator nodes.
estimate:
  # Defines all the estimator nodes.
logging:
  # Defines the run-time logging preferences for the software. We use
    Google glog for logging control.
```

Figure 2.1 shows the structure of the configuration file.

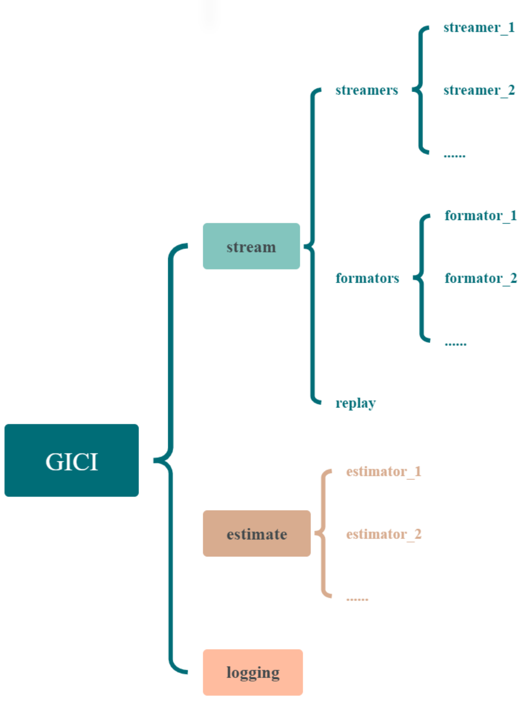


Figure 2.1: Structure of the configuration file

### 2.2.2 Stream Node

There are three second-level nodes under the stream node: streamers, formators, and replay.

**Streamers** In the streamers node, you can define multiple streamer nodes according to your requirements:

```
stream:
  streamers:
  - streamer:
      # Streamer tag, should start with "str_".
      tag: str_...

      # Streamer type.
      type: serial

      # Tags of nodes that the data should be sent to.
      output_tags: [...]

      # Tags of nodes where the data comes from.
      input_tags: [...]

      # Other options. There are different options for different types. See streamer common options.
      ...

  - streamer:
    # Another streamer node. The tag names should not be duplicated.
  ...
```

### Streamer Common Options

Table 2.1 is the streamer common options, which will be used in all streamers.

Table 2.1: Streamer common options

| Option | Description | Unit | Default |
|---|---|---|---|
| tag | Streamer tag, should start with ”str_”. | | ”” |
| type | Streamer type. | | None |
| output_tags | Tags of nodes that the data should be sent to. | | [] |
| input_tags | Tags of nodes where the data comes from. | | [] |
| buffer_length | The length of buffer that temporaly stores the binary stream. It should be determined by the stream load on this channel. | Bit | 32768 |
| loop_duration | Duration of loop. It should be determined by your package update rate. | s | 0.005 |

The ”type” could be ”serial”, ”tcp-client”, ”tcp-server”, ”file”, ”ntrip-client”, ”ntrip-server”, ”v4l2”, and ”ros”.

The ”output_tags” could be formators, other streamers, and estimators (only when setting ”type” as ”ros”). The ”input_tags” could be formators, an another streamer, and estimators (only when setting ”type” as ”ros”).

Note that ”tag” and ”type” are necessary, you must specify them in your configuration file. The other options are optional, you can set them according to your requirements.

### Streamer Serial Options

Table 2.2 is the streamer serial options, which will be used in streamers whose type is serial.

Table 2.2: Streamer serial options

| Option | Description | Unit | Default |
|---|---|---|---|
| port | Serial port. | | ”” |
| baudrate | Baudrate. | | 0 |
| bit_size | Bit size. | | 8 |
| parity | Parity check. Should be n, o, or e. | | n |
| stop_bit | Stop bit. | | 1 |
| flow_control | Hardware flow control for 9-pin serial. Should be off or rts. | | off |

Note that ”port” and ”baudrate” are necessary, you must specify them in your configuration file. The other options are optional, you can set them according to your requirements.

### Streamer TCP/IP Client Options

Table 2.3 is the streamer tcp/ip client options, which will be used in streamers whose type is tcp client.

Table 2.3: Streamer tcp/ip client options

| Option | Description | Unit | Default |
|---|---|---|---|
| ip | Host IP. | | ”” |
| port | Communication port. | | ”” |

Note that options in Table 2.3 are necessary, you must specify them in your configuration file.

### Streamer TCP/IP Server Options

Table 2.4 is the streamer tcp/ip server options, which will be used in streamers whose type is tcp server.

Table 2.4: Streamer tcp/ip server options

| Option | Description | Unit | Default |
|---|---|---|---|
| port | Communication port. | | ”” |

Note that ”port” is necessary.

### Streamer File Options

Table 2.5 is the streamer file options, which will be used in streamers whose type is file.

Table 2.5: Streamer file options

| Option | Description | Unit | Default |
|---|---|---|---|
| path | File path. | | ”” |
| swap_interval | Maximum time duration for storing data in one file. We will create a new file if it is exceeded. Set 0 to disable swap. | h | 0 |
| enable_time_tag | Whether to enable time tag for replay mode. If enabled, a ”.tag” file will be created together with the data file to store data reaching timestamps. This will make the file a replay file. In replay mode, if the file is not a replay file, all the containings will be instantaneously loaded. If it is a replay file, it will be generally loaded according to the ”.tag” file and the replay options. | | true |

Note that ”path” is necessary, you must specify it in your configuration file.

### Streamer Ntrip Client Options

Table 2.6 is the streamer ntrip client options, which will be used in streamers whose type is ntrip client.

Table 2.6: Streamer ntrip client options

| Option | Description | Unit | Default |
|---|---|---|---|
| ip | Host IP. | | ”” |
| port | Communication port. | | ”” |
| username | User name. | | ”” |
| passward | Passward. | | ”” |
| mountpoint | Mountpoint to subscribe. | | ”” |

Note that all the options in Table 2.6 are necessary, you must specify them in your configuration file.

### Streamer Ntrip Server Options

Table 2.7 is the streamer ntrip server options, which will be used in streamers whose type is ntrip server.

Table 2.7: Streamer ntrip server options

| Option | Description | Unit | Default |
|---|---|---|---|
| ip | Host IP. | | ”” |
| port | Communication port. | | ”” |
| passward | Passward to upload data. | | ”” |
| mountpoint | Mountpoint to publish. | | ”” |

Note that all the options in Table 2.7 are necessary, you must specify them in your configuration file.

### Streamer V4L2 Options

This function is developed only for the MT9V034 CMOS on our GICI board. For other CMOS sensors, you should modify the corresponding code to make it fit with your kernel driver.

Table 2.8 is the streamer v4l2 options, which will be used in streamers whose type is v4l2.

Table 2.8: Streamer v4l2 options

| Option | Description | Unit | Default |
|---|---|---|---|
| dev | Device name. | | ”” |
| width | Image width. | | 0 |
| height | Image height. | | 0 |
| buffer_count | V4L2 buffer count. | | 1 |

Note that only ”buffer_count” is optional, you must specify the other options in your configuration file.

### Streamer ROS Options

Table 2.9 is the streamer ros options, which will be used in streamers whose type is ros.

Table 2.9: Streamer ros options

| Option | Description | Unit | Default |
|---|---|---|---|
| io | I/O type, could be input, output, and log | | None |
| format | ROS message format, could be image, imu, gnss_raw, pose_stamped, pose_with_covariance_stamped, marker, path. | | ”” |
| topic_name | Name of ROS topic. | | ”” |
| queue_size | Queue parameter when instantiating ROS topic publisher or subscriber. | | 10 |
| subframe_id | Subframe ID for publishing ROS transform. | | ”” |

Except for the ”gnss_raw”, all the formats are common ROS message types. For our self-defined message types, you can find the definations on ros_wrapper/src/gici/msg, they are also concluded in subsection 2.4.2. For others, see https://wiki.ros.org/ for details.

Note that ”io”, ”format” and ”topic_name” are necessary, you must specify them in your configuration file. The other options are optional, you can set them according to your requirements.

**Formators** In the formators node, you can define multiple formator nodes:

```
stream:
  formators:
  - formator:
      # Formator tag, should start with "fmt_".
      tag: fmt_...

      # I/O type, could be input, output, and log.
      io: input

      # Formator type.
      type: gnss-rtcm-3

      # Tags of nodes that the data should be sent to.
      output_tags: [...]

      # Tags of nodes where the stream comes from.
      input_tags: [...]

      # Other options. There are different options for different types.
   See formator common options.
      ...

  - formator:
    # Another formator node. The tag names should not be duplicated.
  ...
```

The ”type” could be ”gnss-rtcm-2”, ”gnss-rtcm-3”, ”gnss-raw”, ”image-v4l2”, ”image-pack”, ”imu-pack”, ”nmea”, ”dcb-file”, and ”atx-file”.

When setting the ”io” option as ”input”, the ”output_tags” could be estimators and other formators, the ”input_tags” could be a streamer. When setting the ”io” option as ”output”, the ”output_tags” could be a streamer, the ”input_tags” should be a estimator. When setting the ”io” options as ”log”, the ”output_tags” could be a streamer, the ”input_tags” should be an another formator.

### Formator Common Options

Table 2.10 is the formator common options, which will be used in all formators.

Table 2.10: Formator common options

| Option | Description | Unit | Default |
|---|---|---|---|
| tag | Formator tag, should start with "fmt_". | | "" |
| type | Formator type. | | None |
| io | I/O type, could be input, output, and log. | | None |
| output_tags | Tags of nodes that the data should be sent to. | | [] |
| input_tags | Tags of nodes where the stream comes from. | | [] |

Note that "tag", "type", and "io" are necessary, you must specify them in your configuration file.

### Formator GNSS RTCM2/3 Options

Table 2.11 is the formator gnss rtcm2 and rtcm3 options, which will be used in formators whose type is gnss-rtcm-2 and gnss-rtcm-3.

Table 2.11: Formator gnss rtcm2/3 options

| Option | Description | Unit | Default |
|---|---|---|---|
| start_time | A coarse (accurate to day) data start time is need for decoding. | | System |

Note that "start_time" is optional, you can set it according to your requirements.

### Formator GNSS Raw Options

Table 2.12 is the formator gnss raw options, which will be used in formators whose type is gnss-raw.

Table 2.12: Formator gnss raw options

| Option | Description | Unit | Default |
|---|---|---|---|
| sub_type | Raw data type. Could be ublox, septentrio, novatel, or tersus. | | ”” |
| start_time | GICI needs a coarse (accurate to day) data start time. | | ”” |

Note that ”sub_type” is necessary, while ”start_time” is optional.

### Formator Image V4L2 and Image Pack Options

Table 2.13 is the formator image v4l2 and image pack options, which will be used in formators whose type is image-v4l2 and image-pack. These are developed only for GICI board.

Table 2.13: Formator image v4l2 and image pack options

| Option | Description | Unit | Default |
|---|---|---|---|
| width | Image width. | | 0 |
| height | Image height. | | 0 |

Note that ”width” and ”height” are necessary, you must specify them in your configuration file.

### Formator NMEA Options

Table 2.14 is the formator nmea options, which will be used in formators whose type is nmea.

Table 2.14: Formator nmea options

| Option | Description | Unit | Default |
|---|---|---|---|
| use_gga | Whether to use GxGGA message. | | true |
| use_rmc | Whether to use GxRMC message. | | true |
| use_esa | Whether to use GxESA message. | | false |
| use_esd | Whether to use GxESD message. | | false |
| talker_id | NMEA talker ID. | | "GN" |

The GxESA is our self-defined sentence, which stands for Extended Speed and Attitude (ESA). The format is as follows:

```
  $GxESA,tod,Ve,Vn,Vu,Ar,Ap,Ay*checksum
```

where tod is the Time of Day. Ve, Vn, Vu is the velocity in East, North, and Up respectively. Ar, Ap, Ay is the attitude in roll, pitch, and yaw respectively.

The GxESD is also our self-defined sentence, which stands for Extended STD (ESD). The format is as follows:

```
  $GxESD,tod,STD_Pe,STD_Pn,STD_Pu,STD_Ve,STD_Vn,STD_Vu,STD_Ar,STD_Ap,
 STD_Py*checksum
```

where STD_Pe, STD_Pn, STD_Pu are position STD in east, north, and up. STD_Ve, STD_Vn, STD_Vu are the velocity STD in east, north, and up. STD_Ar, STD_Ap, STD_Py are the attitude STD in roll, pitch, and yaw.

Note that all the options in Table 2.14 are optional, you can set them according to your requirements.

### Other Formator Types Options

Other formator types do not have specific options.

**Replay** The replay node defines the replay functions. One can firstly store streams into replay files with the log mode, and then replay the files by replacing the input streams with replay files and setting the replay options. The replay options are defined as follows:

```
stream:
  replay:
    # Whether to enable replay mode.
    enable: false

    # Replay speed.
    speed: 1.0

    # Replay start offset in seconds.
    start_offset: 5.0
```

If set ”enable” as true, all the streams except for file streams are disabled, and the files will be replayed according to its tags and the following options.

### 2.2.3 Estimate Node

You can define the estimators in the second-level node:

```
estimate:
- estimator:
    # Estimator tag, should start with "est_".
    tag: est_...

    # Estimator type.
    type: spp

    # Tags of nodes that the solution should be sent to.
    output_tags: [...]

    # Tags of nodes where the data comes from.
    input_tags: [...]

    # Roles of input tags. xxx is input tag name.
    xxx_roles: [...]
    ...

    # Other options. There are different options for different types.
  See estimator common options.
    ...

- estimator:
    # Another estimator node. The tag names should not be duplicated.
```

The "type" could be "spp", "sdgnss", "dgnss", "rtk", "ppp", "gnss_imu_lc", "spp_imu_tc", "rtk_imu_tc", "gnss_imu_camera_srr", "spp_imu_camera_rrr", and "rtk_imu_camera_rrr".

The "output_tags" could be formators, other estimators, and ROS streamers. The "input_tags" could be formators, other estimators, and ROS streamers.

The "input_tag_roles" could be "rover", "reference", "ephemeris", "ssr_ephemeris", "code_bias", "phase_bias", "heading", and "phase_center" for GNSS data, "major" and "minor" for IMU data, "mono", "stereo_major", "stereo_minor", and "array" for image data.

**Note**: Defining the "output_tags" at the source side equivalents with defining the "input_tags" at the destination side. So is okay to just define the tag connections at one side.

### Estimator Common Options

Table 2.15 is the estimator common options, which will be used in all estimators.

Table 2.15: Estimator common options

| Option | Description | Unit | Default |
|---|---|---|---|
| tag | Estimator tag, should start with "est_". | | "" |
| type | Estimator type. | | None |
| output_tags | Tags of nodes that the solution should be sent to. | | [] |
| input_tags | Tags of nodes where the data comes from. | | [] |
| xxx_roles | Role of the input tags, the xxx should be replaced by the tag of input tags. | | [] |
| output_align_tag | Align the output rate to the data rate of an input formator or streamer. | | "" |
| output_downsample_rate | Downsample rate of output solutions. Set as 1 if you do not want to downsample this data stream. | | [1,...] |
| compute_covariance | Whether to compute covariance. | | true |
| enable_input_align | Whether to align input measurement timestamps. If enabled, the input measurements will be reordered in chronological increments within a short buffer. This will slightly increase the precision and efficiency of estimation but will cause latency. | | false |
| input_align_latency | Latency tolerant if the enable_input_align is enabled. | | 0.0 |
| enable_backend_data_sparsify | Whether to automatically sparsify measurements if backend pending is detected. | | false |

Note that "tag", "type", and "output_align_tag" are necessary, "xxx_roles" is necessary if the optional "input_tags" is specified. The other options are optional, you can set them according to your requirements.

## Estimator Base Options

Table 2.16 is the estimator base options, which will be used in all estimators.

Table 2.16: Estimator base options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_iteration | Max iteration number for ceres optimization | | 10 |
| num_threads | Number of threads used for ceres optimization | | 2 |
| max_solver_time | Maximum time in second for which the optimizer should run for | s | 0.05 |
| solver_type | Ceres solver type | | DENSE_SCHUR |
| trust_region_ strategy_type | Ceres trust region strategy type | | DOGLEG |
| verbose_output | Verbose optimization output | | false |
| force_initial_global_position | Forcely set initial position in global frame | | false |
| initial_global_position | If force_initial_global_position = ture, this parameter sets the initial position in LLA coordinate | deg | $\mathbf{0}_{3\times1}$ |
| compute_covariance | If we should compute covariance | | true |
| log_intermediate_ data | Log estimator intermediate information | | false |
| log_intermediate_ data_directory | Log estiamtor intermediate information to this directory if enabled | | " " |

### GNSS Common Options

Table 2.17 is the GNSS estimator common options, which will be used in GNSS-related estimators.

Table 2.17: GNSS Common Options

| Option | Description | Unit | Default |
|---|---|---|---|
| system_exclude | Usage of satellite systems | | all systems |
| satellite_exclude | Usage of specific satellite | | all satellites |
| code_exclude | Usage of code types | | all code types |
| min_elevation | Minimum elevation angle | deg | 12 |
| min_SNR | Minimum SNR for frequencies 1575.42 MHz (L1) and 1176.45 MHz (L5), SNR masks for other frequencies will be interpolated by a linear model | | (25, 20) |
| max_gdop | Maximum GDOP as valid solution | | 20 |
| mw_slip_thres | Threshold for Melbourne-Wubbena (MW) cycle-slip detection | m | 0.5 |
| gf_slip_thres | Threshold for Geometry-Free (GF) cycle-slip detection | m | 0.05 |
| gf_sd_slip_thres | Threshold for single differenced GF cycle-slip detection | m | 0.05 |
| receiver_pco | Receiver Phaes-Center-Offset (PCO) | m | $\mathbf{0}_{3\times1}$ |

### GNSS Error Parameters

Table 2.18 holds GNSS error factors, which is saved for GNSS base estimator and will be used in GNSS-ralated estimators.

Table 2.18: GNSS Error Parameters

| Option | Description | Unit | Default |
|---|---|---|---|
| code_to_phase_ratio | code noise = phase noise × ratio | | 100 |
| phase_error_factor | Error factor a/b/c according to RTKLIB | | $[0.003, 0.003, 0.0]$ |
| system_error_ratio | System error ratio | | G:1; R:5<br>C:2; E:1.5 |
| ionosphere_broadcast _ factor | Ionosphere model error factor | | 0.5 |
| ionosphere_dual_ frequency | Dual-frequency ionosphere error | m | 0.2 |
| ionosphere_augment | Augmentation ionosphere error | m | 0.03 |
| troposphere_model_factor | Troposphere model error factor | m | 0.2 |
| troposphere_augment | ugmentation troposphere error | m | 0.01 |
| ephemeris_broadcast | Broadcast ephemeris error | m | 3 |
| ephemeris_precise | Precise ephemeris error | m | 0.1 |
| initial_troposphere | Initial troposphere error | m | 0.1 |
| initial_ionosphere | Initial ionosphere error | m | 10 |
| initial_ambiguity | Initial ambiguity error | m | 10 |
| relative_position | Relative position error in ENU used in GNSS-only positioning | m/sqrt (Hz) | $[100, 100, 0]$ |

Continued on next page

Table 2.18 continued from previous page

| Option | Description | Unit | Default |
|---|---|---|---|
| relative_velocity | Relative velocity error in ENU used in GNSS-only positioning, if we estimate receiver velocity, specify this parameter, or, specify the above parameter. | m/sqrt (Hz) | $[10, 10, 10]$ |
| relative_troposphere | relative troposphere delay error | m/sqrt (Hz) | $3 \times 10^{-4}$ |
| relative_ionosphere | Relative ionosphere delay error | m/sqrt (Hz) | $3 \times 10^{-2}$ |
| relative_ambiguity | Relative ambiguity error | m/sqrt (Hz) | $1 \times 10^{-4}$ |
| relative_frequency | Relative receiver frequency error | m/sqrt (Hz) | $1 \times 10^{-2}$ |
| relative_gps_ifcb | Relative GPS Inter-Frequency Clock Bias (IFCB) error | m/sqrt (Hz) | $5 \times 10^{-4}$ |
| residual_gps_ifcb | Residual amplitude for GPS L5 (influenced by uncalibrated IFCB) | m/sqrt (Hz) | 0.02 |

### GNSS Estimator Base Options

Table 2.19 is the GNSS base estimator options, which will be used in all GNSS-related estimators.

Table 2.19: GNSS Estimator Base Options

| Option | Description | Unit | Default |
|---|---|---|---|
| GnssCommonOptions | See Table 2.17 | | |
| GnssErrorParameter | See Table 2.18 | | |
| use_outlier_rejection | Use Fault Detection and Exclusion (FDE) | | true |
| reject_one_outlier_once | Reject outlier at a time or reject all | | false |
| max_pesudorange_error | Maximum pseudorange error to exclude | m | 4.0 |
| max_phaserange_error | Maximum phaserange error to exclude | m | 0.03 |
| max_doppler_error | Maximum doppler error to exclude | m/s | 0.5 |
| good_observation_min_num_satellites | Minimum number of satellite to be considered as good observation environment | | 10 |
| good_observation_max_gdop | Maximum GDOP value to be considered as good observation environment | | 2.0 |
| good_observation_max_reject_ratio | Maximim outlier rejection ratio to be considered as good observation environment | | 0.1 |
| reset_ambiguity_min_num_continuous_unfix | Minimum number of continuous unfix under good observation to reset ambiguities | | 10 |
| diverge_max_reject_ratio | Maximum outlier rejection ratio to be considered as diverging | | 0.5 |
| diverge_min_num_continuous_reject | Minimum number of continuous large amount rejection to be considered as divergence | | 10 |

### Ambiguity Resolution Options

Table 2.20 is the ambiguity resolution options.

Table 2.20: Ambiguity Resolution Options

| Option | Description | Unit | Default |
|---|---|---|---|
| system_exclude | Usage of satellite systems. Currently we do not support GLONASS ambiguity resolution | | [R] |
| satellite_exclude | Usage of specific satellite. In default, we use all satellites | | |
| phase_exclude | Usage of phase types. In default, we use all phase types | | |
| min_elevation | Minimum elevation angle | deg | 15 |
| min_percentage_fixation_nl | Percentage of narrow lane ambiguity fixation to consider as fixed solution | | 0.9 |
| min_percentage_fixation_wl | Percentage of wide lane ambiguity fixation to consider as succeed | | 0.9 |
| min_percentage_fixation_uwl | Percentage of ultra wide lane ambiguity fixation to consider as succeed | | 1 |
| min_num_satellite_pairs_fixation | Minimum number of satellite pairs for valid ambiguity resolution | | 6 |
| ratio | Ambiguity fixation ratio for LAMBDA | | 3 |

### DGNSS (RTD) Estimator Options

Table 2.21 is the DGNSS estimator options.

Table 2.21: DGNSS Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| estimate_velocity | Estimate velocity or not | | true |
| max_age | Maximum age to apply difference | | 20 |

### SDGNSS Estimator Options

Table 2.22 is the SDGNSS estimator options.

Table 2.22: SDGNSS Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| estimate_velocity | Estimate velocity or not | | true |
| max_age | Maximum age to apply difference | | 20 |

### SPP Estimator Options

Table 2.23 is the single point positioning estimator options.

Table 2.23: SPP Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| estimate_velocity | Estimate velocity or not | | true |

### PPP Estimator Options

Table 2.24 is the PPP estimator options.

Table 2.24: PPP Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_window_length | Max window length | | 10 |
| use_ambiguity_resolution | Use ambiguity resolution | | false |
| estimate_velocity | Estimate velocity or not | | true |

#### RTK Estimator Options

Table 2.25 is the RTK estimator options.

Table 2.25: RTK Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_window_length | Max window length | | 3 |
| use_ambiguity_resolution | Use ambiguity resolution | | true |
| estimate_velocity | Estimate velocity or not | | true |
| max_age | Maximum age to apply difference | | 20 |

#### IMU Parameters

Table 2.26 is the IMU parameters saved for imu base estimator, which will be used in INS-related estimators.

Table 2.26: IMU Parameters

| Option | Description | Unit | Default |
|---|---|---|---|
| a_max | Accelerometer saturation | m/$s^2$ | 150 |
| g_max | Gyroscope saturation | rad/s | 7.8 |
| sigma_g_c | Gyroscope noise density | rad/s $\times$ 1/sqrt(Hz) | $1 \times 10^{-4}$ |
| sigma_bg | Initial gyroscope bias uncertainty | rad/s $\times$ 1/sqrt(Hz) | 0.01 |
| sigma_a_c | Accelerometer noise density | m/$s^2$ $\times$ 1/sqrt(Hz) | $2 \times 10^{-3}$ |
| sigma_ba | Initial accelerometer bias uncertainty | m/$s^2$ $\times$ 1/sqrt(Hz) | 0.1 |
| sigma_gw_c | Gyroscope drift noise density | rad/$s^2$ $\times$ 1/sqrt(Hz) | $2.1 \times 10^{-5}$ |
| sigma_aw_c | Accelerometer drift noise density | m/$s^3$ $\times$ 1/sqrt(Hz) | $8.4 \times 10^{-4}$ |
| g | Earth acceleration | m/$s^2$ | 9.8 |
| rate | IMU rate | Hz | 400 |
| delay_imu_cam | Camera-IMU delay: delay_imu_cam = cam_timestamp - imu_timestamp | s | 0.0 |

### INS Estimator Options

Table 2.27 is the INS estimator options, which will be used in INS-related estimators.

Table 2.27: INS Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| ImuParameters | See Table 2.26 | | |
| body_to_imu_rotation | IMU to car rotation | deg | $\mathbf{0}_{3\times1}$ |
| body_to_imu_rotation_std | STD of IMU to body rotation | deg | 3.0 |
| car_motion | If car motion | | false |
| car_motion_min_velocity | Minimum velocity to apply car motion constraints | m/s | 3.0 |
| car_motion_max_anguler_velocity | Maximum angular rate to apply car motion constraints | deg/s | 5.0 |

## Visual Estimator Base Options

Table 2.28 is the visual estimator base options, which will be used in visual-related estimators.

Table 2.28: Visual Estimator Base Options

| Option | Description | Unit | Default |
|---|---|---|---|
| feature_error_std | Feature error STD | pixel | 2 |
| landmark_outlier_rejection_threshold | Landmark outliter rejection threshold (pixel) | | 2 |
| max_frequency | Maximum frequency of visual backend processing | Hz | 10 |
| diverge_max_reject_ratio | Maximum outlier rejection ratio to be considered as diverging | | 0.5 |
| diverge_min_num_continuous_reject | Minimum number of continuous large amount rejection to be considered as divergence | | 10 |

### GNSS/INS Loosely Coupled (LC) Estimator Options

Table 2.29 is the GNSS/INS loosely-coupled estimator options

Table 2.29: GNSS/INS Loosely Coupled (LC) Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_window_length | Max window length | | 10 |

### GNSS/INS Initializer Options

Table 2.30 is the GNSS/IMU initializer, which will be performed when initialize estimator that contains GNSS and IMU.

Table 2.30: GNSS/INS Initializer Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_iteration | Max iteration number for ceres optimization | | 30 |
| num_threads | Number of threads used for ceres optimization | | 4 |
| max_solver_time | Maximum time in second for which the optimizer should run for | | 0.5 |
| time_window_length_slow_motion | We should keep stady during this period to initialize roll, pitch and angular rate bias | | 0.1 |
| time_window_length_dynamic_motion | We should do a dynamic motion after the slow motion initilization has finished, the time window will be used for bundle adjustment. | | 0.5 |
| min_acceleration | At least one state should have horizontal acceleration larger than this in the dynamic motion window, we need a relatively large acceleration to ensure the observability of yaw attitude | | 0.5 |
| gnss_extrinsics | Relative position from IMU to GNSS in IMU frame | | $\mathbf{0}_{3\times1}$ |
| gnss_extrinsics_initial_std | GNSS extrinsics initial variance | | $\mathbf{0}_{3\times1}$ |

### SPP/INS Tightly Coupled (TC) Estimator Options

Table 2.31 is the SPP/INS Tightly Coupled (TC) Estimator options.

Table 2.31: SPP/INS Tightly Coupled Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_window_length | Max window length | | 10 |

### RTK/INS Tightly Coupled (TC) Estimator Options

Table 2.32 is the RTK/INS Tightly Coupled (TC) estimator options.

Table 2.32: RTK/INS Tightly Coupled (TC) Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_window_length | Max window length | | 10 |

### Feature Detector Options

Table 2.33 is the feature detector options inherited from [3].

Table 2.33: Feature Detector Options

| Option | Description | Unit | Default |
|---|---|---|---|
| cell_size | Maximum one feature per bucked with cell_size width and height | | 30 |
| max_level | Extract features on pyramid | | 2 |
| min_level | minimum pyramid level at which features should be selected | | 0 |
| border | no feature should be within border. | | 8 |
| detector_type | Choose between FAST and FAST_GRAD, FAST_GRAD will use Edgelets. | | kFast |
| threshold_secondary | Secondary detector threshold. Used if the detector uses two different detectors. E.g. in the case of FAST_GRAD, it is the gradient detector threshold. | | 100 |
| sampling_level | Level where features are initialized. Only for detectors supporting specific feature levels like the AllPixelDetector. | | 0 |
| sec_grid_fineness | fineness level of the secondary grid (used for extra shi tomasi features when loop closing is enabled) | | 1 |
| threshold_shitomasi | Corner Strength Thrshold for shitomasi features (used only when loop closing is enabled) | | 100 |

### Feature Tracker Options

Table 2.34 is the feature tracker options also inherited from [3].

Table 2.34: Feature Tracker Options

| Option | Description | Unit | Default |
|---|---|---|---|
| window_size | Size of the search window at each pyramid level | | [21, 21] |
| max_level | 0-based maximal pyramid level number; if set to 0, pyramids are not used (single level), if set to 1, two levels are used, and so on | | 3 |
| max_count | The maximum number of iterations or elements to compute | | 30 |
| epsilon | The desired accuracy or change in parameters at which the iterative algorithm stops | | 0.01 |
| use_relative_rotation | Use relative rotation to set initial pixel of LK optical flow iteration when it is avaible | | true |
| ransac_threshold | Threshold of ransac to reject tracking outliers | pixel | 1 |
| ransac_confidence | specifies a desirable level of confidence (probability) that the estimated matrix is correct | | 0.99 |

### Visual Initialization Options

Table 2.35 is the visual initialization options, which could be used for initializing visual-related estimators.

Table 2.35: Visual Initialization Options

| Option | Description | Unit | Default |
|---|---|---|---|
| init_type | kHomography for estimating a plane from the first two views, kFundamental for estimating fundamental matrix from the first two views | | kFunda-mental |
| init_min _disparity | Minimum disparity (length of feature tracks) required to select the second frame. After that we triangulate the first pointcloud. For easy VisualInitialization you want to make this small (minimum 20px) but actually it is much better to have more disparity to ensure the initial pointcloud is good. | | 50 |
| init_disparity_ pivot_ratio | When checking whether the disparity of the tracked features are large enough, we check that a certain percentage of tracked features have disparities large than init_min_disparity. The default percentage is 0.5, which means the median is checked. For example, if this parameter is set to 0.25, it means we go for realtive pose estimation only when at least 25% of the tracked features have disparities large than init_min_disparity | | 0.5 |
| init_min_features | If less features than init_min_features can be extracted at the first frame, the first frame is not accepted and we check the next frame. | | 30 |
| init_min_inliers | At the end of VisualInitialization, we triangulate the first pointcloud and check the quality of the triangulation by evaluating the reprojection errors. All points that have more reprojection error than reproj_error_thresh are considered outliers. Only return SUCCESS if we have more inliers than init_min_inliers. | | 20 |
| init_map_scale | Initial map average depth only used for homography init | | 100 |
| reproj_error_ thresh | Reprojection threshold | pixel | 2 |

### Feature Handler Options

Table 2.36 is the feature detector options used for feature-related applications.

Table 2.36: Feature Handler Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_n_kfs | Max number of keyframes to keep | | 10 |
| max_features_per_frame | Max features per frame | | 120 |
| kfselect_min_numkfs | If we have less than this amount of features we always select a new keyframe. | | 60 |
| kfselect_min_disparity | Minimum disparity to select a new keyframe | | 10 |
| kfselect_min_dist_metric | Minimum distance in meters before a new keyframe is selected. | | 1.5 |
| kfselect_min_angle | Minimum angle in degrees to closest KF | | 10 |
| kfselect_min_dt | Minimum time duration in seconds to forcely select a new keyframe. This is used to control the long duration IMU drift under a slow or static motion. | | 2 |
| max_pyramid_level | Image max pyramid level | | 3 |
| min_disparity_init_landmark | Minimum disparity to triangulate a landmark | | 5 |
| detector | Feature detector options in Table 2.33 | | |
| tracker | Feature tracker options in Table 2.34 | | |
| initialization | Initialization options in Table 2.35 | | |
| cameras | Camera model, can be ATAN, Pinhole or Ocam (see vikit) | | |

### GNSS/INS/Camera SRR Estimator Options

Table 2.37 is the GNSS/INS/Camera SRR estimator options.

Table 2.37: GNSS/INS/Camera SRR Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_keyframes | Frame state window length. We only keep GNSS measurements near to keyframes (one-to-one) and throw the others away after one optimization, because the GNSS measurement errors, especially for the multipath, are highly correlated between epochs when we have a slow or zero motion | | 5 |
| max_gnss_window_length_minor | GNSS state window length before visual has been initialized | | 10 |
| min_yaw_std_init_visual | Maximum yaw STD to start visual initialization | deg | 0.5 |

### SPP/INS/Camera RRR Estimator Options

Table 2.38 is the SPP/INS/Camera RRR estimator options.

Table 2.38: SPP/INS/Camera RRR Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_keyframes | Frame state window length | | 5 |
| max_gnss_window_length_minor | GNSS state window length before visual has been initialized | | 3 |
| min_yaw_std_init_visual | Maximum yaw STD to start visual initialization | deg | 0.5 |

**RTK/INS/Camera RRR Estimator Options**

Table 2.39 is the RTK/INS/Camera RRR estimator options.

Table 2.39: RTK/INS/Camera RRR Estimator Options

| Option | Description | Unit | Default |
|---|---|---|---|
| max_keyframes | Frame state window length | | 5 |
| max_gnss_window_length_minor | GNSS state window length before visual has been initialized | | 3 |
| min_yaw_std_init_visual | Maximum yaw STD to start visual initialization | deg | 0.5 |

### 2.2.4 Logging Node

The definitions in the logging node is very simple. You should just specify the following 4 options:

```
logging:
    # Enable of disable logging, should be true or false.
    enable: true

    # Minimum logging level.
    min_log_level: 0

    # Whether we should print the logging streams to stderr.
    log_to_stderr: true

    # Directory you want to generate the log files.
    file_directory: ...
```

The "min_log_level" should be 0 - 3. Where 0 - 3 stands for INFO, WARNING, ERROR, FATAL, respectively.

if set "log_to_stderr" to false, we will create log files in "file_directory". If set as true, the logs will be printed to stderr, and the files will not be created.

## 2.3 Typical Usage Scenarios

In GICI, there are three typical usage scenarios: "Stream Transfer and Format Conversion", "Real-Time Estimation" and "Offline Pseudo-Real-Time Estimation". Stream threads are utilized in the first scenarios, so only the "stream node" should be set in the configuration file, which is described in subsection 2.2.2. As the name implies, the estimator stream will be additionally used in the other two scenarios, so the "estimate node" should be additionally set, which is described in subsection 2.2.3. If you want to apply them to your project, you can include the corresponding node in your configuration file. In addition, we also provide some configuration file examples and corresponding datasets for everyone to verify and simulate.

### 2.3.1 Stream Transfer and Format Conversion

There are five sub-scenarios about stream transfer and format conversion in GICI.

#### Data Storage

You can transfer the stream from "serial", "TCP client", "Ntrip client", "V4L2" into "file" in real time , as shown in Figure 2.2. The words in the box of the figure correspond exactly to the "type" in the stream node. Please refer to Table 2.1 through Table 2.6 or "option/data_storgae.yaml" for guidance on how to set them in configuration file.

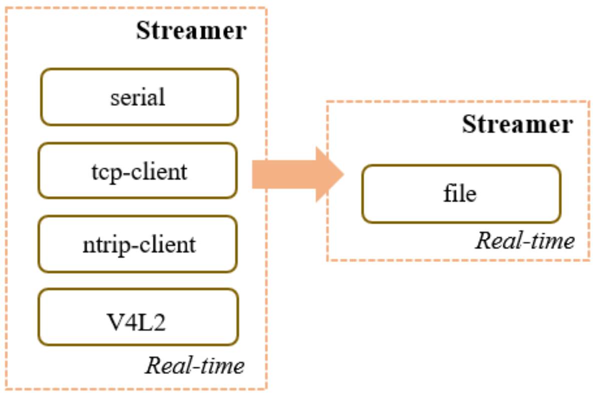


Figure 2.2: The real-time data storage. The "serial", "tcp-client", "ntrip-client", "V4L2" and "file" are the types of the streamer node.

This feature is commonly used for dataset collection. The stored files can be used in subsection 2.3.3 for algorithm development.

### Data Broadcast

You can broadcast the GNSS data stream from "serial", "TCP client" and "Ntrip client" in real time to "serial", "TCP server" and "Ntrip server", as shown in Figure 2.3. Please refer to Table 2.1 through Table 2.7 pr "option/data_broadcast.yaml" for guidance on how to set them in configuration file.

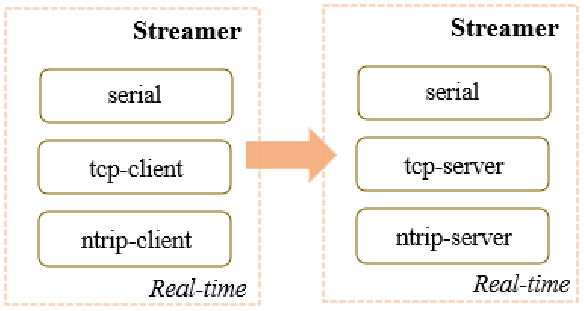


Figure 2.3: The real-time broadcast of stream

This feature is commonly used for broadcasting GNSS correction messages to users, such as the Observation State Representation (OSR) message used for RTK, and the State Space Representation (SSR) message used for PPP.

### Format Conversion and Storage

If you want to convert the data format before saving them to files, you can use the format conversion and storage feature. Figure 2.4 shows the data streaming process from real-time data input to decoding and encoding, and finally storage to file in real-time. The "Formator" decodes and converts the format of the information from multiple inputs, and then outputs it to a file in real-time. For example, you can convert various formats of GNSS, and for more conversion between formats, please refer to "src/stream/formator.cpp". The words in the box of the figure correspond exactly to the "type" in the stream node. Please refer to Table 2.1 through Table 2.14 or "option/format_conversion_and_storage.yaml" for guidance on how to set them in configuration file.

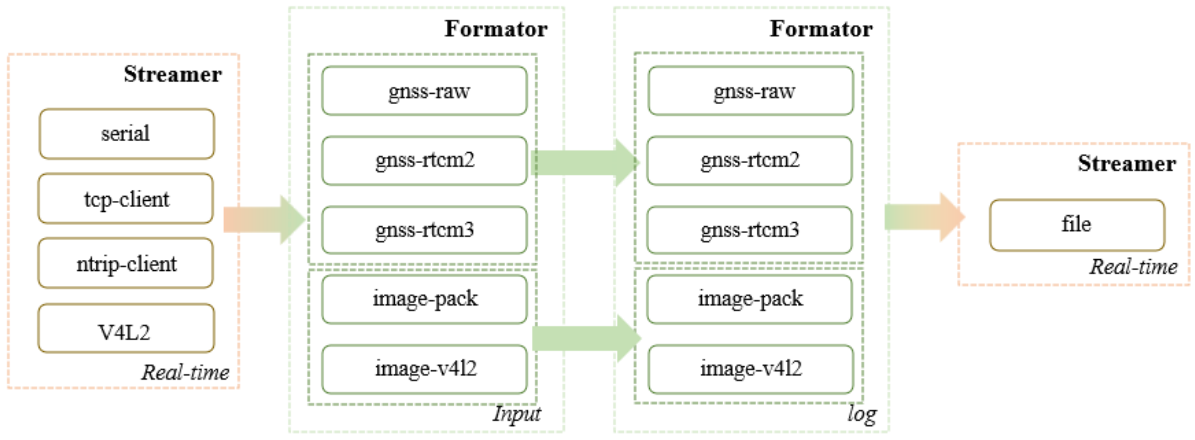


Figure 2.4: Format conversion and storage

### Format Conversion and Broadcast

If you want to convert the data format before broadcasting them to other streams, you can use the format conversion and broadcast feature. Figure 2.5 shows the GNSS data streaming process from real-time data input to decoding and encoding, and finally broadcast to serial or ICP/Ntrip in real-time. Please refer to "option/format_conversion_and_broadcast .yaml" for guidance on how to set them in configuration file.

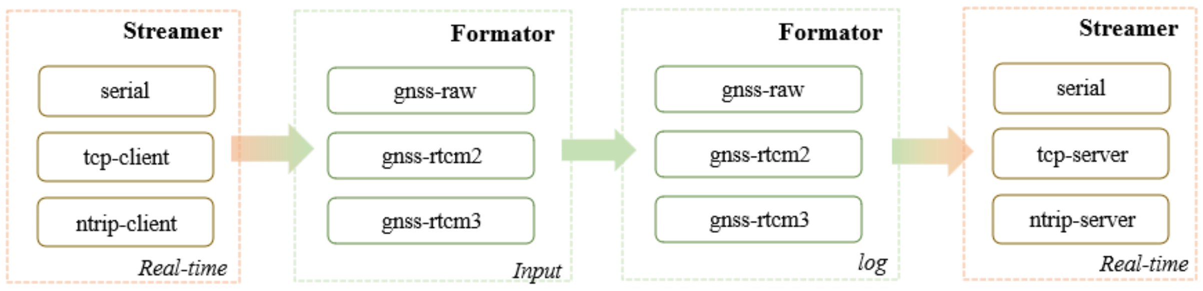


Figure 2.5: Format conversion and broadcast

### Publish Data to ROS Topics

If you build GICI in ROS wrapper, you can publish the data from interface to ros topics. Figure 2.6 shows the data streaming process from real-time data input to decoding, and finally publish to ROS topics in real-time. Please refer to Table 2.9 for guidance on how to set ROS streamer node in configuration file. It's worth noting that if ROS is chosen as the input or output, there is no need for a corresponding "formator" to perform decoding or encoding. Please refer to Table 2.10 through Table 2.14 or "ros_wrapper/src/gici/option/piblish_data_to_ros_topics.yaml" for guidance on how to set them in configuration file.

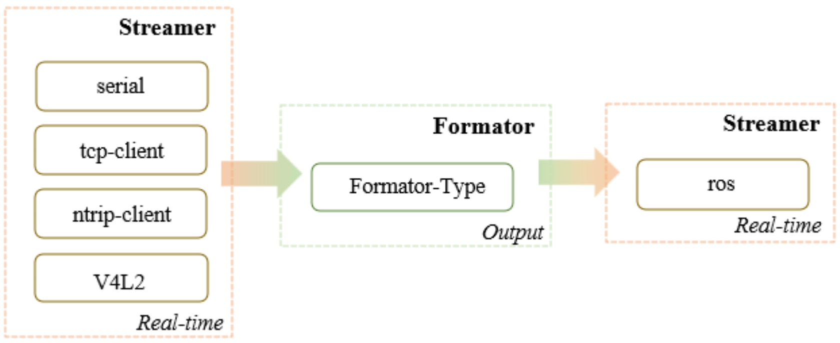


Figure 2.6: Publish data to ROS topics. The "Formotor-Type" in figure could be "gnss-rtcm-2", "gnss-rtcm-3", "gnss-raw", "image-v4l2", "image-pack" and "imu-pack", which are correspond to the "type" of "formator" in configuration file.

### 2.3.2 Real-Time Estimation

One of the most important features of GICI is to apply multi-sensor estimation, so real-time processing and estimation is the main scenario of GICI. As shown in the Figure 2.7, the data input from various interfaces of the "streamer" is first decoded by "formator" and then fed into the "estimator" in real-time. The output of the estimator can be 1) encoded and output in real-time, 2) published through ROS topics for real-time output, and 3) output to other "estimator", "formator" or "streamer(ros)".

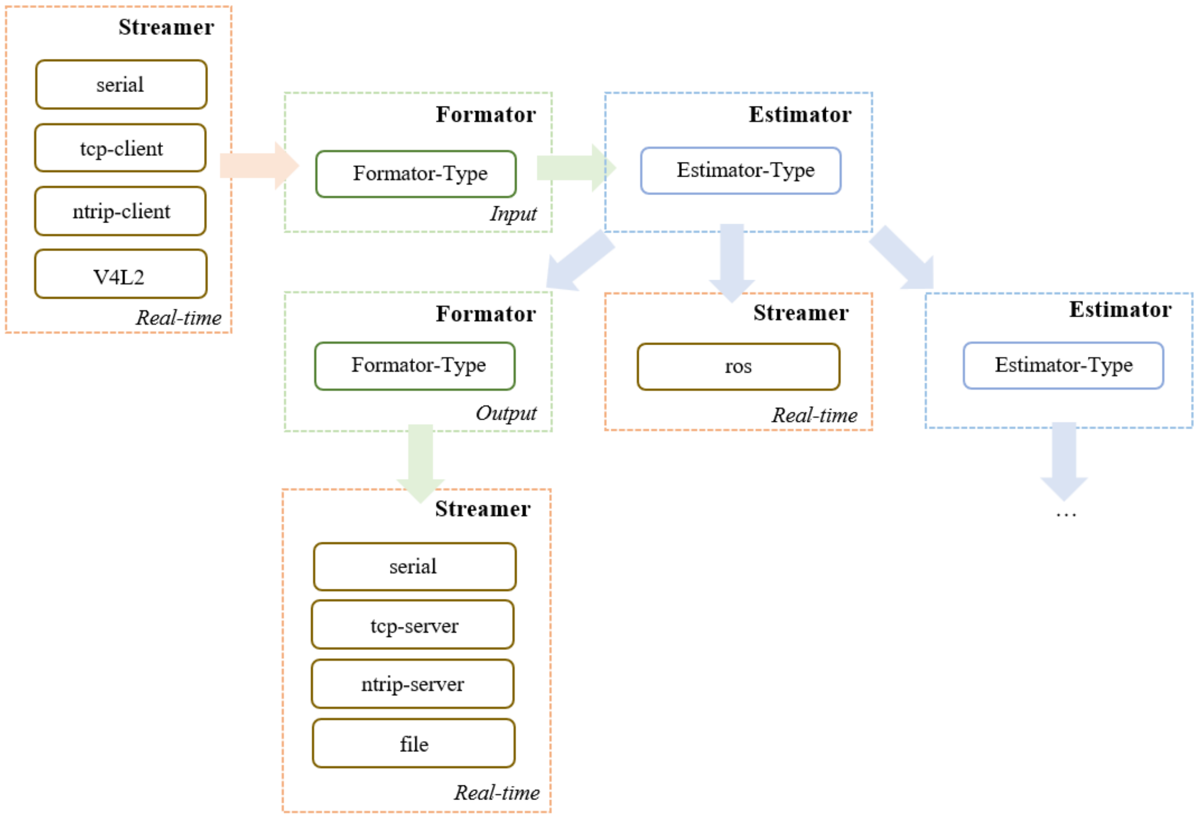


Figure 2.7: The structure of real-time estimation. The "..." could be formator, streamer(ros) or other estimator.

The "Formotor-Type" in figure could be "gnss-rtcm-2", "gnss-rtcm-3", "gnss-raw", "image-v4l2", "image-pack", "imu-pack", "nmea", "dcb-file", and "atx-file". Please refer to Table 2.1 through Table 2.14 for guidance on how to set them in configuration file.

The "Estimator-Type" in figure could be "spp", "sdgnss", "dgnss", "rtk", "ppp", "gnss_imu_lc", "spp_imu_tc", "rtk_imu_tc", "gnss_imu_camera_srr", "spp_imu_camera_rrr", and "rtk_imu_camera_rrr". For example, the first estimator type can be "rtk", and the second estimator type whose the first outputs to can be "gnss_imu_camera_srr". Please refer to subsection 2.2.3 or "option/real_time_estimation.yaml" for guidance on how to set them in configuration file.

In the example configuration file "option/real_time_estimation.yaml", we demonstrated how to configure each node of the "gnss_imu_camera_srr" type for real-time estimation. The data input from each interface is decoded and then sent to the RTK estimator and the GNSS/IMU/Camera SRR estimator. The estimated pose results are then encoded and output in NMEA format. Of course, you can also add other estimators or outputs (such as outputting to ROS topics), which can refer to the related configuration file of pseudo real-time estimation in subsection 2.3.3.

In addition, you can perform real-time processing and estimation by accessing ROS topics, as is shown in Figure 2.8. In the example configuration file "ros_wrapper/src/gici/option/ros_real_time_estimation_xxx.yaml", we subscribed to real-time ROS topics as inputs and published ROS topics about the output pose and image containing feature points.

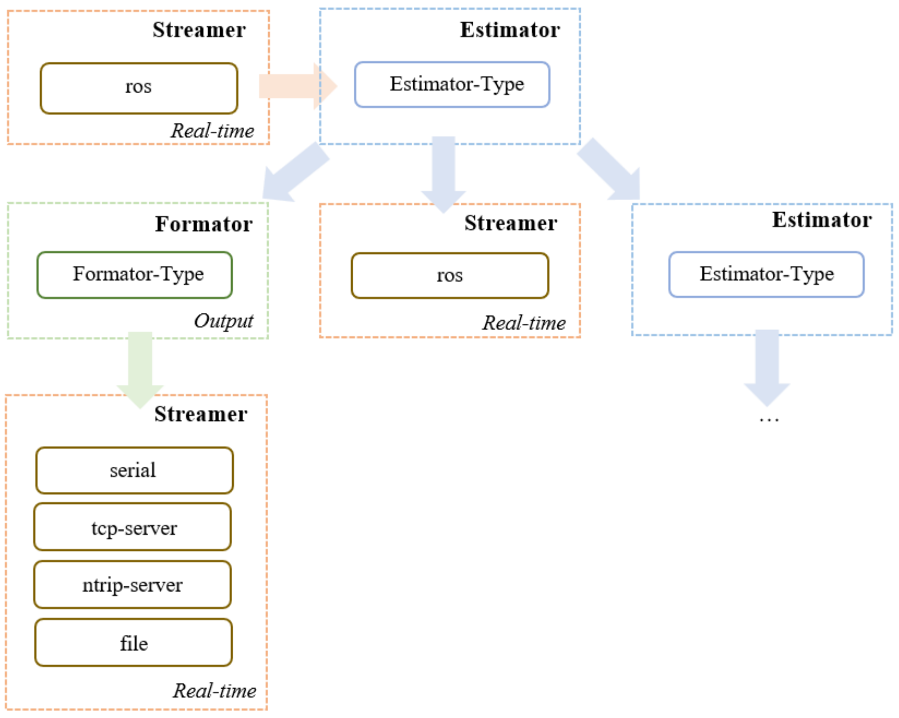


Figure 2.8: The structure of real-time estimation through ROS. The "..." could be formator, streamer(ros) or other estimator.

## 2.3.3 Offline Pseudo-Real-Time Estimation

GICI supports offline pseudo-real-time processing, which enables algorithm performance testing and scientific research by replaying previously stored files or rosbags. Compared with pure post-processing, pseudo-real-time processing can maximize the restoration of real-time running conditions, including sensor bus delay, thread synchronization, and thread blocking.

You can perform pseudo-real-time processing by replaying previously stored files or rosbags, as shown in Figure 2.9 and Figure 2.10. The encoded file data or ROS topics can be sent to estimator for corresponding estimation, and afterwards can be sent to different "streamer", "formator" or "estimator" nodes. For the convenience of readers in handling the provided datasets or datasets recorded by themselves, we provide multiple sample configuration files named "option/pseudo_real_time_estimation_xxx.yaml" for various types of pseudo-real-time estimation. If you build GICI in ROS wrapper, you can also use ROS topics as the input and output of pseudo-real-time estimation, see "ros_wrapper/src/gici /option/ros_real_time_estimation_xxx.yaml".

You can also use the visualization tool RVIZ in ROS to subscribe to related topics and observe the results of GICI estimation. We also provide an RVIZ configuration file "ros_wrapper/src/gici/rviz/gici_gic.rviz" that is suitable for observing the estimation results of our datasets. It is worth noting that we only provide the "xxx.bin" file datasets that we recorded. If you want to use our datasets as an input in rosbags, we suggest that you use the "tools/ros/gici_files_to_rosbag" tool to convert files to rosbags before data input.

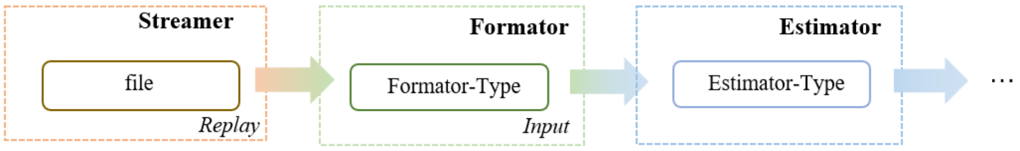


Figure 2.9: The structure of pseudo-real-time estimation

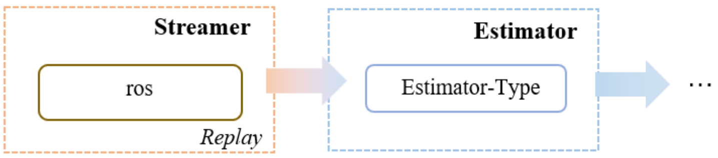


Figure 2.10: The structure of pseudo-real-time estimation through ROS

In addition to the above usage scenarios, you can also set configuration files refer to the source code and your own needs.

## 2.4 Hardware Configuration

In this section, we will introduce how to collect valid data that can be used by GICI. It should be noted that developing a well-synchronized GNSS/INS/Camera system is not that easy. We strongly suppose you use our dataset if you do not have sufficient hardware experiments.

We briefly introduce the principles here. subsection 2.4.1 introduces how to configure a sensor to output valid data. Then subsection 2.4.2 introduces how to use ROS topics as sensor data source. Finally, subsection 2.4.3 illustrates how to apply hardware time synchronization between sensors, which is essential for multi-sensor fusion algorithms achieving prospective performance.

### 2.4.1 Stream Input and Data Format

We introduce three types of input data: GNSS, IMU, and camera.

#### GNSS Data

Commonly, GNSS data is transferred via a serial port from the receiver (either integrated or chipset). Thereby, you should configure an input serial streamer to capture the stream

```
- streamer:
    tag: str_gnss_rov
    output_tags: [fmt_gnss_rov]
    type: serial
    port: ttyUSB0
    baudrate: 115200
```

The formator node is defined as

```
- formator:
    io: input
    tag: fmt_gnss_rov
    type: gnss-raw
    sub_type: tersus
```

The "type" could be either gnss-raw, gnss-rtcm-2, or gnss-rtcm-3. After defining the above nodes, the decoded data can be fed to any nodes you want.

Since we need the GNSS data format to decode the messages, you should configure your GNSS receiver to output data in the corresponding formats that you specified in the formator node.

For the gnss-raw format, we currently support ublox, septentrio, novatel, and tersus formats, as described in Table 2.12. You should find a receiver with the corresponding brand and configure it by serial via string commands. We inherit the decoders from RTKLIB [1], you can find the commands in the "data/cmd" folder of their source code folder `https://github.com/tomojitakasu/RTKLIB/tree/rtklib_2.4.3`.

If you are not using the support brands of receivers, you can first try to find out if they support the above raw message types. If your receiver do not support the messages, you can use the gnss-rtcm-2 or gnss-rtcm-3 messages. Since the RTCM messages are designed for broadcasting the differential measurements, the raw measurement output capacity may not be fully supported by your receiver. In this case, the minimum requirement is to make sure that the pseudorange and carrier phase are outputted. If possible, output the doppler measurement together.

No matter which message type you are using, a high-frequency output is preferred. Commonly, the output frequency could be 1 Hz $\sim$ 100 Hz, try to output frequency as higher as possible within the acceptable consumption of the computational load.

### IMU Data

Commonly, IMU data is transferred via SPI, IIC, or serial from the IMU sensor. In GICI, we do not support SPI or IIC, because the data acquisition job is commonly done by an MCU, and then transferred to a Linux CPU via an asynchronous bus, such as openAMP or serial. An example IMU stream handle node is

```
- streamer:
    tag: str_imu
    output_tags: [fmt_imu]
    type: serial
    port: ttyUSB0
    baudrate: 115200
```

The formator node is defined as

```
- formator:
    io: input
    tag: fmt_imu
    type: imu-pack
```

Table 2.40: Message format of imu-pack

| Preamble | Length | Data | Parity |
|---|---|---|---|
| 0xFECB | 8 bit | Length × 8 bit | 8 bit |

Table 2.41: Data format of imu-pack

| ID | Field | Description | Type | Bits | Unit |
|---|---|---|---|---|---|
| 1 | sec | Second part of timestamp. | uint | 32 | s |
| 2 | nsec | Nano-second part of timestamp. | uint | 32 | ns |
| 3 | acc-x | Accelaration in the x-axis. | int | 20 | $\frac{2^{20}}{9.8\times 24}$ m/s$^2$ |
| 4 | acc-y | Accelaration in the y-axis. | int | 20 | $\frac{2^{20}}{9.8\times 24}$ m/s$^2$ |
| 5 | acc-z | Accelaration in the z-axis. | int | 20 | $\frac{2^{20}}{9.8\times 24}$ m/s$^2$ |
| 6 | gyro-x | Angular velocity in the x-axis. | int | 20 | $\frac{2^{20}\times 180}{4000\times \pi}$ rad/s |
| 7 | gyro-y | Angular velocity in the y-axis. | int | 20 | $\frac{2^{20}\times 180}{4000\times \pi}$ rad/s |
| 8 | gyro-z | Angular velocity in the z-axis. | int | 20 | $\frac{2^{20}\times 180}{4000\times \pi}$ rad/s |

We only support imu-pack for IMU messages. The imu-pack format is

We provide the decoder and encoder for the imu-pack type, see format_imu.c. You can also instantiate your own decoder in formator.cpp if you want to use another message type.

### Camera Data

Commonly, camera data is transferred via a camera bus, such as MIPI and DCMI. Linux users can read data from these buses via a V4L2 driver if the corresponding kernel driver is instantiated.

Although these interfaces have been largely unified by V4L2, the operations still vary between sensors. Hence, if you want to use our V4L2 interface, you should modify the corresponding codes to make it fit with your sensor. An example configuration for V4L2 streamer is

```
- streamer:
    tag: str_camera
    output_tags: [fmt_camera]
    type: v4l2
    dev: video0
    height: 480
    width: 752
    buffer_count: 361472  # 752 * 480 + 512
```

The formator node is defined as

```
- formator:
    io: input
    tag: fmt_camera
    type: image-v4l2
    height: 480
    width: 752
    step: 1
```

Using the V4L2 port is a professional task. An alternative way is to find a camera module that transfers data via normal buses, such as the LAN port or USB port. Note that the module should also support time synchronization. Currently, we do not support USB ports. For LAN, we support TCP client and TCP server. The following node is an example from which we receive image messages from our GICI-board:

```
- streamer:
    tag: str_camera
    output_tags: [fmt_camera]
    type: tcp-client
    ip: 192.168.1.101
    port: 9021
    buffer_length: 361472  # 752 * 480 + 512
```

The formator node is defined as

```
- formator:
    io: input
    tag: fmt_camera
    width: 752
    height: 480
    type: image-pack
```

We implemented this pipeline for the convenience of debugging. After GICI-board acquires images from V4L2, we pack the image into the image-pack format and publish it to a TCP server. Then we use the above nodes to get images from GICI-board in a computer. The image-pack format is

Table 2.42: Message format of image-pack

| Preamble | Length | Data | Tail |
|---|---|---|---|
| 0xFECA00FF00FF | 24 bit | Length $\times$ 8 bit | 0xCCFEFF00FF00 |

We provide the decoder and encoder for the image-pack type, see format_image.c. You can also instantiate your own decoder in formator.cpp if you want to use another message type.

### 2.4.2 ROS Stream

As introduced in subsection 2.4.1, if you want to obtain data directly from the hardware, some development work needs to be conducted. If you do not want to do so, an alternative way is to use our ROS interface.

For IMU and camera, there are typical messages defined in ROS, i.e. sensor_msgs::Imu and sensor_msgs::Image. And you can also find modules that support outputting the corresponding ROS messages. Note that you should choose a module that supports hardware synchronization, and synchronize the IMUs and cameras to the GNSS time system.

Table 2.43: Data format of image-pack

| ID | Field | Description | Type | Bits | Unit |
|---|---|---|---|---|---|
| 1 | sec | Second part of timestamp. | uint | 32 | s |
| 2 | nsec | Nano-second part of timestamp. | uint | 32 | ns |
| 3 | width | Image width. | uint | 16 | |
| 4 | height | Image height. | uint | 16 | |
| 5 | step | Image step. | uint | 8 | |
| 6 | data | Image data buffer. | uint | width $\times$ height $\times$ step $\times$ 8 | |

For GNSS, there is no valid message type defined for raw GNSS measurement data in ROS. Hence, we define the messages in GICI. You can find the message files in ros_wrapper/src/gici/msg. We conclude the message definitions below:

Table 2.44: GnssEphemeris.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| prn | Satellite PRN number | string | |
| week | GPS week | uint16 | |
| sva | SV accuracy (URA index) | uint8 | m |
| code | GPS: code on L2, GAL/BDS: data source | uint16 | |
| iode | Issue of data, ephemeris | uint8 | |
| iodc | Issue of data, clock | uint16 | |
| svh | SV health | uint8 | |

Continued on next page

Table 2.44 continued from previous page

| Variable | Description | Type | Unit |
|---|---|---|---|
| toc | Time of clock | float64 | s |
| idot | SV orbit parameter | float64 | rad/s |
| crs | SV orbit parameter | float64 | m |
| deln | SV orbit parameter | float64 | rad/s |
| M0 | SV orbit parameter | float64 | rad |
| cuc | SV orbit parameter | float64 | rad |
| e | SV orbit parameter | float64 | |
| cus | SV orbit parameter | float64 | rad |
| A | SV orbit parameter | float64 | m |
| toes | SV orbit parameter | float64 | s |
| cic | SV orbit parameter | float64 | rad |
| OMG0 | SV orbit parameter | float64 | rad |
| cis | SV orbit parameter | float64 | rad |
| i0 | SV orbit parameter | float64 | rad |
| crc | SV orbit parameter | float64 | m |
| omg | SV orbit parameter | float64 | rad |
| OMGd | SV orbit parameter | float64 | rad/s |
| tgd | Group delay parameters | float64[] | s |
| f2 | SV clock parameter | float64 | $s/s^2$ |
| f1 | SV clock parameter | float64 | s/s |
| f0 | SV clock parameter | float64 | s |

Table 2.45: GlonassEphemeris.msg

| Variable | Discription | Type | Unit |
|---|---|---|---|
| prn | Satellite PRN number | string | |
| week | GPS week | uint16 | |
| frq | Satellite frequency number | int8 | |
| iode | Issue of data, ephemeris | uint8 | |
| svh | SV health | uint8 | |
| age | Age of operation | uint8 | days |
| toe | Epoch of epherides | float64 | s |
| tof | Message frame time | float64 | s |
| pos | Satellite position | float64[] | km |
| vel | Satellite velocity | float64[] | km/s |
| acc | Satellite acceleration | float64[] | km/s$^2$ |
| taun | SV clock bias | float64 | s |
| gamn | SV relative frequency bias | float64 | |
| dtaun | Delay between L1 and L2 | float64 | s |

Table 2.46: GnssEphemerides.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| header | Header of the message | std_msgs/Header | |
| ephemerides | GPS, BDS, Galileo ephemeris of each satellites | GnssEphemeris[] | |
| glonass_ephemerides | Glonass ephemeris of each satellites | GlonassEphemeris[] | |

Table 2.47: GnssAntennaPosition.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| header | Header of the message | std_msgs/Header | |
| pos | Station position in ECEF | float64[] | |

Table 2.48: GnssIonosphereParameters.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| header | Header of the message | std_msgs/Header | |
| type | Parameter type (0:GPS,1:BDS,2:Galileo) | uint8 | |
| parameters | Parameters | float64[] | |

Table 2.49: GnssObservation.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| prn | Satellite PRN number | string | |
| week | GPS week | uint16 | |
| tow | GPS time of week | float64 | s |
| SNR | Signal strengths | uint16[] | $10^{-3}$ dBHz |
| LLI | Loss of lock indicators | uint8[] | |
| code | Code indicators | string[] | |
| L | Carrier phase cycles | float64[] | cycle |
| P | Pseudoranges | float64[] | m |
| D | Dopplers | float64[] | Hz |

Table 2.50: GnssObservations.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| header | Header of the message | std_msgs/Header | |
| observations | Observations of each satellite | GnssObservation[] | |

Table 2.51: GnssSsrCodeBias.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| prn | Satellite PRN number | string | |
| week | GPS week, SSR message | uint16 | |
| tow | GPS time of week, SSR message | float64 | |
| iod | Issue of data, SSR message | uint32 | |
| udi | SSR update interval | float64 | |
| isdcb | If differenced | uint8 | |
| code | Code indicators | string[] | |
| bias | Code biases | float64[] | m |

Table 2.52: GnssSsrCodeBiases.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| header | Header of the message | std_msgs/Header | |
| biases | SSR code biases | GnssSsrCodeBias[] | |

Table 2.53: GnssSsrEphemeris.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| prn | Satellite PRN number | string | |
| week | GPS week, SSR message | uint16 | |
| tow | GPS time of week, SSR message | float64 | |
| iod | Issue of data, SSR message | uint32 | |
| udi | SSR update interval | float64 | |
| iode | Issue of data, ephemeris | uint8 | |
| iodcrc | Issue of data crc for BDS | uint32 | |
| refd | Sat ref datum (0:ITRF, 1:regional) | uint8 | |
| deph | Delta orbit (radial, along, cross) | float64[] | m |
| ddeph | Dot delta orbit (radial, along, cross) | float64[] | m/s |
| dclk | Delta clock (c0, c1, c2) | float64[] | s |

Table 2.54: GnssSsrEphemerides.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| header | Header of the message | std_msgs/Header | |
| corrections | SSR ephemeris corrections | GnssSsrEphemeris[] | |

Table 2.55: GnssSsrPhaseBias.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| prn | Satellite PRN number | string | |
| week | GPS week, SSR message | uint16 | |
| tow | GPS time of week, SSR message | float64 | |
| iod | Issue of data, SSR message | uint32 | |
| isdpb | If differenced | uint8 | |
| udi | SSR update interval | float64 | |
| phase | Phase indicators | string[] | |
| bias | Phase biases | float64[] | m |

Table 2.56: GnssSsrPhaseBiases.msg

| Variable | Description | Type | Unit |
|---|---|---|---|
| header | Header of the message | std_msgs/Header | |
| biases | SSR phase biases | GnssSsrPhaseBias[] | |

To use our GNSS messages, you should use GICI to collect your GNSS data using the ”Publish Data to ROS Topic” feature described in subsection 2.3.1.

### 2.4.3 Hardware Time Synchronization

Hardware Time Synchronization is to ensure the sensor measurements are stamped under a unified hardware time system. Since GICI does not support time system offset estimation, we strongly recommend you apply hardware time synchronization, or you may get an unsatisfactory performance. Based on our experience, the precision of time synchronization should be better than 5 ms for normal mobilities, such as car, Unmanned Aerial Vehicle (UAV), and handed motion.

To apply hardware time synchronization, the first thing is to appoint a time base hardware. Commonly, it is a CPU driven by a crystal oscillator. A simple hardware time synchronization system is to use the CPU to trigger all the sensors, record the trigger time, and stamp the time to the corresponding arrived measurement data. However, the CPUs for receiving the measurement data may be distributed because of the requirements of processing speed and memory space. For example, we commonly use an MCU to trigger the IMU and camera sensors because of its strong real-time performance, but acquiring image data, especially for high-resolution data, is almost impossible for an MCU. Hence, we use an MPU to acquire and handle the image data. In such cases, the different CPUs should be synchronized together before they handle the sensors.

To synchronize multiple CPUs together, the time base CPU should generate synchronization signals to the other CPUs, and the other CPUs receive the signal and tune their local clocks. There are many kinds of synchronization signals, such as Pulse Per-Second (PPS) and Precision Time Protocol (PTP). Let's take PPS as an example to illustrate. On the time base CPU side, a rising (or descending) edge is generated on an I/O port every time when the time reaches integer seconds. And a package of data containing the current time in seconds is transferred through a serial port after the edge signal has been generated. On the slave CPU side, it captures the edge signal and serial message, and adjusts its decimal time counter and integer time counter by a control algorithm.

After triggering a sensor, it starts measuring, and then sending the data to your processor after the measurement has been finished. The intermediate will cause a delay in time. This delay should be carefully calibrated because it may be hard for you to distinguish which data is corresponding to the trigger time.

There are some sensors that do not support external triggering, such as the BMI088 IMU on our GICI-board. For such sensors, you can find the data delay in their manual or ask the manufacturer for this parameter. Then you can stamp the data as the local CPU time when you receive the data interrupt minus the delay.

For the GNSS receiver, it cannot be triggered for measuring because it is itself a timing system. It can output the PPS signal to users, which delivers a nano-second-level timing solution. Hence, for the systems that contain GNSS receivers, we appoint the receivers as time bases (different GNSS receivers are naturally synchronized together), and let their PPS signals synchronize all the CPUs. Then, all the sensor timestamps are synchronized to the GNSS time.

# Chapter 3

# Theories

## 3.1 Factor Graph Optimization

### 3.1.1 Describing Least-Squares Problem in Factor Graph

The multi-sensor integrated estimation problem can be described as a Least-Squares (LSQ) problem by forming residuals from each measurement and state constraint. Optimization is to solve the LSQ problem. And factor graph is an intuitive way to describe the LSQ problem.

Given an LSQ problem:

$$\min_{\chi} \frac{1}{2} \sum_{i} \rho_i \left( \|z_i - f_i(\chi)\|^2_{R_i} \right) \tag{3.1}$$

where $\chi$ is the parameters to be estimated, $z$ is a measurement or constraint (for simplicity, we will use "measurement" to represent the "measurement or constraint" in the rest of this manual), $f$ is a non-linear or linear model, $R$ is the covariance of $z$, $\rho$ is a loss function, $i$ is the index of measurements.

We can use a graph to describe the problem, see Figure 3.1. There are two kinds of elements in a graph: nodes and edges. The nodes represent the estimated parameters and measurements. The edges form connections between the parameter nodes and measurement nodes, which are actually, the residuals.

Normally, each measurement $z$ does not connect to every parameter given in $\chi$. For example, in a 15-parameter GNSS/INS loosely integration problem, the parameters are

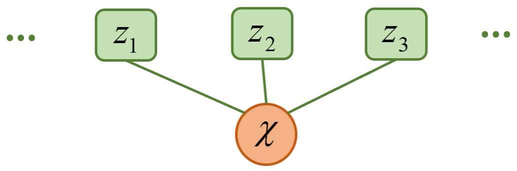


Figure 3.1: LSQ problem described by a graph.

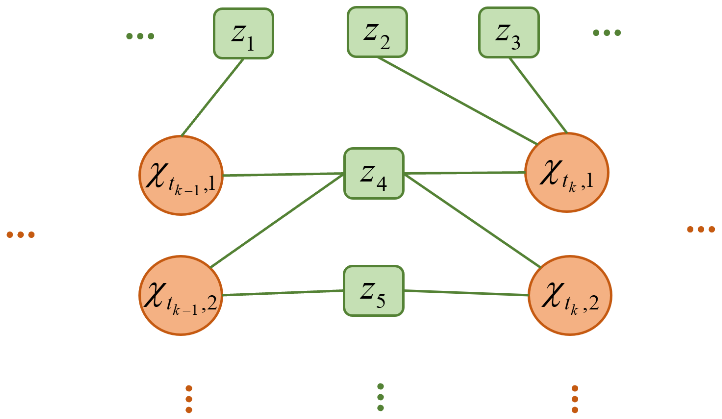


Figure 3.2: LSQ problem described by a graph in a more general way.

positions, attitudes, speeds, accelerometer bias, and gyroscope bias. The GNSS position measurement only connects to the position parameter in the corresponding epoch. The INS integration error connects to all the parameters in the two adjacent epochs.

Hence, we can draw the graph in a more general way, see Figure 3.2. It clearly shows the relationship between measurements and parameters. One can add or delete items by editing the nodes and edges. One can also analyze the sparsity of the whole problem to optimize the solving speed. It is more effective compared with directly dealing with the residual and Jacobian equations.

The LSQ problem can be solved by many optimization methods. In GICI, we use Ceres-Solver [4] to solve the LSQ problems. See `http://ceres-solver.org/tutorial.html` for details.

### 3.1.2 Marginalization

If we consider state propagation between epochs, the dimension of the LSQ problem increases over time. We cannot let it increase endlessly in real-time processing. Commonly, we define the length of a sliding window to constrain the maximum dimension. In this case, marginalization converts the old parameters and residuals as prior information, rather than throwing them, to ensure optimality.

We follow the theories on [2] and describe the concept of marginalization in brief. Given a linearized (or linear) LSQ problem $\boldsymbol{z} = \boldsymbol{J}\delta\boldsymbol{\chi}$. We form a Gauss-Newton equation $\boldsymbol{H}\delta\boldsymbol{\chi} = \boldsymbol{b}$, where $\boldsymbol{H} = \boldsymbol{J}^T\boldsymbol{J}$ is the Hessian matrix, $\boldsymbol{b} = \boldsymbol{z} - h(\boldsymbol{\chi})$ is the residual vector. Let us consider a set of states to be marginalized out, $\boldsymbol{\chi}_\mu$, the set of all states related to those by residual terms, $\boldsymbol{\chi}_\lambda$, and the set of remaining states, $\boldsymbol{\chi}_\rho$. Due to conditional independence, we can simplify the marginalization step by ignoring the states $\boldsymbol{\chi}_\rho$ and only apply it to a sub-problem:

$$\begin{bmatrix} \boldsymbol{H}_{\mu\mu} & \boldsymbol{H}_{\mu\lambda_1} \\ \boldsymbol{H}_{\lambda_1\mu} & \boldsymbol{H}_{\lambda_1\lambda_1} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\chi}_\mu \\ \delta\boldsymbol{\chi}_\lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_\mu \\ \boldsymbol{b}_{\lambda_1} \end{bmatrix} \tag{3.2}$$

Application of the Schur complement operation yields:

$$\boldsymbol{H}^*_{\lambda_1\lambda_1}\delta\boldsymbol{\chi}_\lambda = \boldsymbol{b}^*_{\lambda_1} \tag{3.3a}$$

$$\boldsymbol{H}^*_{\lambda_1\lambda_1} := \boldsymbol{H}_{\lambda_1\lambda_1} - \boldsymbol{H}_{\lambda_1\mu}\boldsymbol{H}^{-1}_{\mu\mu}\boldsymbol{H}_{\mu\lambda_1} \tag{3.3b}$$

$$\boldsymbol{b}^*_{\lambda_1} := \boldsymbol{b}_{\lambda_1} - \boldsymbol{H}_{\lambda_1\mu}\boldsymbol{H}^{-1}_{\mu\mu}\boldsymbol{b}_\mu \tag{3.3c}$$

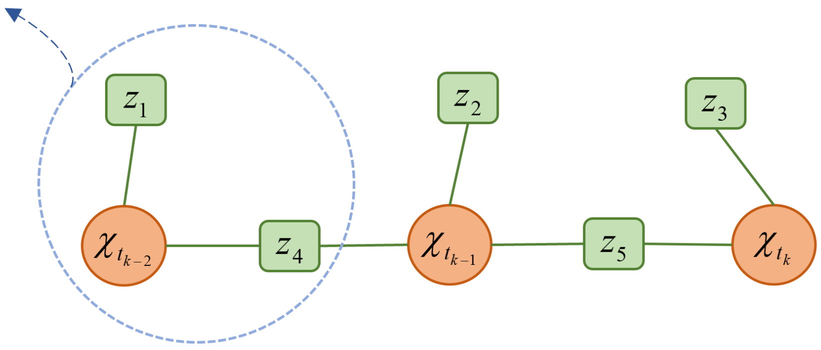
where $\boldsymbol{b}^*_{\lambda_1}$ and $\boldsymbol{H}^*_{\lambda_1\lambda_1}$ are nonlinear functions of $\delta\boldsymbol{\chi}_\mu$ and $\delta\boldsymbol{\chi}_\lambda$.

Equation 3.3a puts a new residual item into the LSQ problem, which represents all the prior information that has been marginalized. We can describe this process in a graph, see Figure 3.3. The old parameters and residuals are converted to a residual item through the marginalization process. This process is applied repeatedly as time goes on.

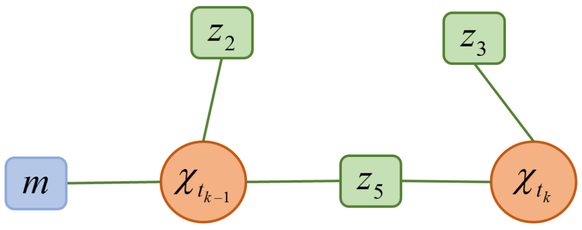
## 3.2 FGO structure in GICI

### 3.2.1 Graph Structure

There are two kinds of graph structures used in GICI, the single-epoch structure, and the sliding-window structure. If we do not need to consider the time propagation between epochs, we will use the single-epoch structure. This structure is used in the GNSS SPP and the GNSS RTD. If the time propagation constraints, e.g. the INS pre-integration error,



(a) Graph before marginalization.



(b) Graph after marginalization.

Figure 3.3: Marginalization process described in graphs.

the velocity-position model, and the constant variable constraint, should be applied, we will use the sliding-window structure. This structure is used in the rest of the estimators.

The single-epoch structure can be described by Figure 3.1. We solve the LSQ problem for each epoch of measurements separately since we assume that there is no time propagation between epochs. We also do not need to apply marginalization because no prior information can be connected to a new epoch.

The sliding-window structure can be described by Figure 3.3(b). We maintain a window of epochs and solve the whole graph every time when a new epoch has been formed. The oldest epochs will be marginalized if the window is full.

### 3.2.2 Wrapping

No matter which kind of graph structure is used, we need to define the residual and Jacobian for each measurement, and the ways to edit the graph.

We call these definitions factors. The factors are used to form edges between parameter nodes and measurement nodes. We implement a lot of factors for different sensors working in different scenarios. We will introduce them in the following sections. Note that there are many frame transformation operations in the formulations, please see Appendix A for the definitions.

Moreover, to make developers easy to form their graphs, we implement base classes for each kind of sensor, which contain the adding, erasing, marginalizing, rejecting, and accessing functions for parameter and measurement nodes. The multi-sensor fusion problem can be implemented by simply inheriting the base classes and calling their functions. You can refer to xxx_estimator_base.h files for details.

## 3.3 GNSS Loose Integration Error Factors

Generally, a GNSS receiver can provide three levels of measurement: 1) Solutions, including position and velocity in a global frame. 2) Raw measurements, including pseudorange, doppler frequency, and carrier phase. 3) Base-band data, i.e., the intermediate frequency signal. Thereby, there are three kinds of integration formulation: loose integration, tight integration, and deep integration, correspondingly.

In GICI, we implement loose integration and tight integration formulations. We will introduce the former in the reset of this section, and the latter will be introduced in section 3.4.

### 3.3.1 Position Error Factor

The position solution provided by a GNSS receiver is in the global frame ${}^{W_G}\mathcal{F}$. We implement two parameterization methods to handle the GNSS measurements. One forms the parameters in a GNSS-receiver-centered global frame ${}^{W_G}\mathcal{F}$, and the other forms in an INS-centered local frame ${}^{W}\mathcal{F}$. We will introduce them separately.

#### GNSS-receiver-centered global frame formulation

This formulation can be used when we do not need to estimate the orientation.

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T\right]^T \tag{3.4}$$

Given a GNSS position measurement ${}^{W_G}\hat{\boldsymbol{p}}_k^T$ at epoch $k$, The residual is

$$\boldsymbol{r}_k := {}^{W_G}\hat{\boldsymbol{p}}_k - {}^{W_G}\boldsymbol{p}_k \tag{3.5}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := -\boldsymbol{I} \tag{3.6}$$

The covariance matrix is a $3 \times 3$ matrix generated by the GNSS position algorithm.

#### INS-centered local frame formulation

If one wants to estimate the orientation, the estimation center is commonly defined as the center of an IMU sensor. So we define the INS-centered formulation here to handle this scenario.

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, \boldsymbol{q}_{B,k}^{W}{}^T, {}^{B}\boldsymbol{t}_r^T\right]^T \tag{3.7}$$

where ${}^{B}\boldsymbol{t}_r$ is the GNSS receiver position in the body frame, i.e. the lever-arm.

The residual is

$$\boldsymbol{r}_k := {}^{W_G}\hat{\boldsymbol{p}}_k - (\boldsymbol{R}_W^{W_G}({}^W\boldsymbol{p}_k + \boldsymbol{R}_{B,k}^{W}{}^B\boldsymbol{t}_r) + {}^{W_G}t_W) \tag{3.8}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := -\boldsymbol{R}_W^{W_G}\begin{bmatrix} \boldsymbol{I} & -\lfloor \boldsymbol{R}_{B,k}^{W}{}^B\boldsymbol{t}_r \times \rfloor & \boldsymbol{R}_{B,k}^{W} \end{bmatrix} \tag{3.9}$$

The covariance matrix is a $3 \times 3$ matrix generated by the GNSS position algorithm.

### 3.3.2 Velocity Error Factor

Similarly, the velocity provided by a GNSS receiver is also in the global frame ${}^{W_G}\mathcal{F}$. We implement two parameterization methods to handle the GNSS measurements.

#### GNSS-receiver-centered global frame formulation

This formulation can be used when we do not need to estimate the orientation.

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{v}_k^T\right]^T \tag{3.10}$$

Given a GNSS velocity measurement ${}^{W_G}\boldsymbol{z}_k^T$ at epoch $k$, the residual is

$$\boldsymbol{r}_k := {}^{W_G}\boldsymbol{z}_k - {}^{W_G}\boldsymbol{v}_k \tag{3.11}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := -\boldsymbol{I} \tag{3.12}$$

The covariance matrix is a $3 \times 3$ matrix generated by the GNSS positioning algorithm.

#### INS-centered local frame formulation

The INS-centered formulation is applied to handle scenarios containing an IMU.

The related parameters are

$$\boldsymbol{\chi}_k := \left[\boldsymbol{q}_{B,k}^{W}{}^T, {}^W\boldsymbol{v}_k^T, {}^B\boldsymbol{t}_r^T\right]^T \tag{3.13}$$

The residual is

$$\boldsymbol{r}_k := {}^{W_G}\boldsymbol{z}_k - R_W^{W_G}({}^{W}\boldsymbol{v}_k + \lfloor \omega_k \times \rfloor \boldsymbol{R}_{B,k}^{W}\,{}^{B}\boldsymbol{t}_r) \tag{3.14}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := -R_W^{W_G} \begin{bmatrix} \boldsymbol{0} & -\lfloor \omega_k \times \rfloor \lfloor \boldsymbol{R}_{B,k}^{W}\,{}^{B}\boldsymbol{t}_r \times \rfloor & \boldsymbol{I} & \lfloor \omega_k \times \rfloor \boldsymbol{R}_{B,k}^{W} \end{bmatrix} \tag{3.15}$$

where $\omega_k$ is the angular velocity of the interval $k$ in the ${}^{W}\mathcal{F}$.

The covariance matrix is a $3 \times 3$ matrix generated by the GNSS positioning algorithm.

## 3.4 GNSS Tight Integration Error Factors

The GNSS signal structure can be simply described by Figure 3.4. The GNSS signal is generally composed of the multiplication of the carrier frequency (Carrier), the spreading code (Code), and the navigation data (Data). The spreading codes are also called PRN (pseudo-random noise) codes. The pseudorange measurement is obtained by decoding the spreading code, and the carrier phase measurement is obtained by stripping the code and data from the carrier. There is also a doppler shift caused by the relative motivation between the satellite and the user. The doppler measurement is obtained by measuring the frequency shift of the carrier.

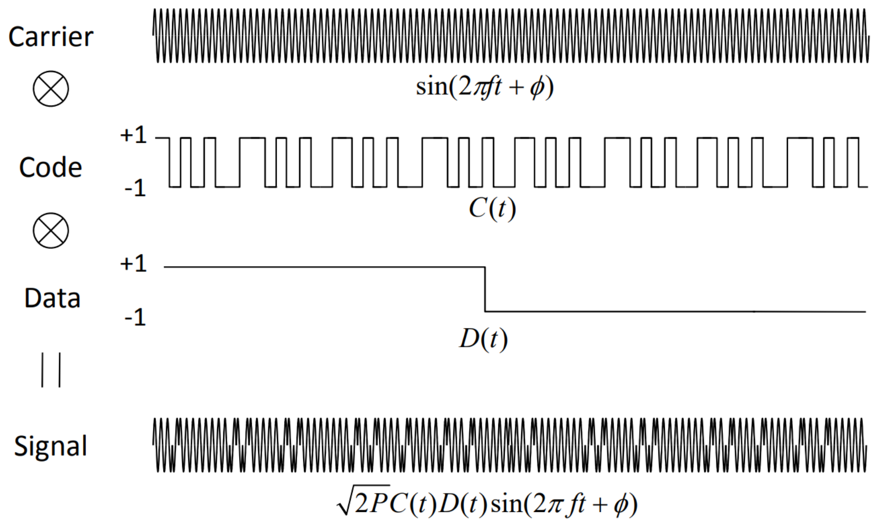


Figure 3.4: GNSS signal structure [1].

Generally, the accuracy of pseudorange, doppler-velocity, and phase-range are meter-level, decimeter-per-second-level, and millimeter-level, respectively. Hence, for standard precision positioning (meter- to decimeter-level) formulations, e.g. SPP and RTD, we just use pseudorange and doppler. For precise positioning (centimeter- to millimeter-level) formulations, e.g. PPP and RTK, we additionally use the carrier phase. We will describe the measurements in the rest of this section, and introduce the formulations in section 3.8.

### 3.4.1 Pseudorange Error Factors

The pseudorange measurement can be described as

$$P_{r,i}^s = \rho_r^s + c\left(dt_r - dt^s\right) + I_{r,i}^s + T_r^s + d_{r,i} - d_i^s + \varepsilon_P \tag{3.16}$$

where indices $s$, $r$, and $i$ refer to the satellite, receiver, and carrier frequency band, respectively. $\rho_r^s = \left\|p_r - p^s\right\|^2$is the geometry distance between the receiver antenna phase center and satellite phase center. $c$ is the speed of light in the vacuum. $dt_r$ and $dt^s$ represent the receiver clock offset at the signal receiving time and the satellite clock offset at the signal transmitting time respectively. $I_{r,i}^s$ is the ionospheric delay along the signal propagation path at the $i$-th frequency. $T_r^s$ is the tropospheric delay of the signal path. $d_{r,i}$ and $d_i^s$ are code biases for receiver and satellite. $\varepsilon_P$ is the un-modeled errors, mainly containing the multipath and random noise.

There are several methods to handle these items. In GICI, we implement 12 formulations, which are the permutations of 2 frame definitions, 3 GNSS linear combination types, and 2 ways to handle atmosphere delays. The frame definitions are the GNSS-receiver-centered global (GRCG) frame and the INS-centered local (ICL) frame. The GNSS linear combination types are undifferenced and uncombined (UDUC), single-difference (SD), and double-difference (DD). The 2 ways to handle atmosphere delays are correcting by model and estimating. The 12 formulations are listed in Table 3.1. We will illustrate them in detail.

#### Common Theories

Here we illustrate some common procedures for handling some variables:

1. Satellite position $p^s$ and satellite clock offset $dt^s$.

Table 3.1: Formulations of pseudorange estimation.

| | Frame | Combination | Atmosphere | Usage in GICI |
|---|---|---|---|---|
| Formulation 1 | GRC | UDUC | Correct | SPP |
| Formulation 2 | GRC | UDUC | Estimate | PPP |
| Formulation 3 | ICL | UDUC | Correct | SPP-based tightly integration |
| Formulation 4 | ICL | UDUC | Estimate | PPP-based tightly integration |
| Formulation 5 | GRC | SD | Eliminated | SDGNSS |
| Formulation 6 | GRC | SD | Estimate | |
| Formulation 7 | ICL | SD | Eliminated | |
| Formulation 8 | ICL | SD | Estimate | |
| Formulation 9 | GRC | DD | Eliminated | RTD and RTK |
| Formulation 10 | GRC | DD | Estimate | |
| Formulation 11 | ICL | DD | Eliminated | RTD- and RTK-based tightly integration |
| Formulation 12 | ICL | DD | Estimate | |

In SPP, RTD, and short-baseline RTK, the two variables are computed by ephemeris, which is broadcasted by the navigation satellites. In PPP and long-baseline RTK, they are computed by precise ephemeris, which, in real-time cases, is broadcasted by some special navigation satellites (like BDS B2b), commercial satellites, or the internet.

2. Ionosphere delay $I_{r,i}^s$.

In SPP, the ionosphere delay is computed by models

$$I_{r,i}^s = \mathcal{I}(\boldsymbol{\iota}) \tag{3.17}$$

where $\iota$ is a vector of parameters broadcasted by navigation satellites. $\mathcal{I}$ is the ionosphere model. We use the Klobuchar [5] model in GICI.

In RTD and short-baseline RTK, the ionosphere delay is eliminated by the double-difference combination.

In PPP and long-baseline RTK, the ionosphere delay at the base frequency for each satellite $I_{r,1}^s$ is estimated. For the other frequencies, we only consider the first-order ionosphere delay, i.e.

$$I_{r,i}^s = \gamma_i I_{r,1}^s, \quad \gamma_i = \frac{f_1^2}{f_i^2} \tag{3.18}$$

where $f_i$ is the signal frequency at the signal channel $i$.

The ionosphere delay can also be computed by other models, such as the Satellite Based Augmentation System (SBAS) model, the Global Ionosphere Map (GIM) model, and the PPP-RTK ionosphere grid model. These models are not implemented in GICI, so we do not discuss them in this manual.

3. Troposphere delay $T_r^s$.

In GICI, the troposphere delay is modeled as two parts, the hydro-static delay $T_{r,h}^s$, and the wet delay $T_{r,w}^s$, i.e.

$$T_r^s = T_{r,h}^s + T_{r,w}^h \tag{3.19}$$

The hydro-static delay in the zenith direction, calling zenith total delay (ZTD), is computed by the Saastamoinen model [6]. And the ZTD is mapped to satellites according to the hydro-static part of the Global Mapping Function (GMF) [7]:

$$T_{r,h}^s = \mathcal{G}_h(\alpha_e) T_{Z,h}, \quad T_{Z,h} = \mathcal{S}({}^{W_G}p) \tag{3.20}$$

where $\alpha_e$ is the elevation angle of the satellite.

In SPP, we only consider the hydro-static delay.

In RTD and short-baseline RTK, the troposphere delay is eliminated by the double-difference combination.

In PPP and long-baseline RTK, we estimate the wet delay in the zenith direction $T_{Z,w}$, and map $T_{Z,w}$ to satellites according to the GMF wet model:

$$T^s_{r,w} = \mathcal{G}_w(\alpha_e)T_{Z,w} \tag{3.21}$$

There are also other models dealing with the troposphere delay, we do not discuss them in this manual.

4. Satellite code bias $d^s_i$.

If the between-receiver difference (SD or DD) is applied, this item is eliminated and does not need to be handled separately. Otherwise, it should be treated carefully. We will discuss this situation in the following.

The absolute values of code biases are unobservable because they are coupled with clock biases. So we can only work with the relative values. To correct the satellite code biases, we must first define the base frequency of the satellite clock, and all the corrections should be relative to the base frequency. For the broadcast ephemeris, the base frequencies for GPS, GLONASS, BDS, and Galileo are L1/L2 IF combination (see Appendix B), G1/G2 IF combination, B3, and E1/E5a IF combination, respectively. These definitions vary for the precise ephemeris products, so we do not introduce them here. By defining the base frequency, all the satellite code biases correction operations aim to arrange the measurement to the base frequency. Take the GPS system as an example, the broadcasted satellite clock offset $d\bar{t}^s$ contains the true clock offset $dt^s$ and the code bias in the base frequency:

$$d\bar{t}^s = dt^s - (\frac{\gamma_2}{\gamma_2 - 1}d^s_1 - \frac{1}{\gamma_2 - 1}d^s_2) = dt^s - d^s_{IF} \tag{3.22}$$

Since the satellite clock offset is unique for all satellites (in one navigation system) and all frequencies, one must arrange the pseudorange measurements to the base frequency to make them consistent.

There are three kinds of products to correct satellite code biases, Total Group Delay (TGD), Differential Code Bias (DCB), and Zero-Differenced Code Bias (ZDCB).

The TGD, together with the Inter-System Corrections (ISC), is contained in the broadcast ephemeris, which describes the satellite code bias between frequencies. For GPS, the

TGD is defined as

$$d^s_{TGD} := \frac{1}{1-\gamma_2}(d^s_1 - d^s_2) = d^s_{IF} - d^s_1 = \frac{1}{\gamma_2}(d^s_{IF} - d^s_2) \tag{3.23}$$

which describes the satellite code bias between the frequencies L1 and L2. The ISCs, similarly, describe the biases between other frequencies.

The DCB is commonly provided in text files by the International GNSS Service (IGS) analysis centers or other institutions. The definition is

$$d^s_{DCB,i,j} := d^s_i - d^s_j \tag{3.24}$$

which is similar to the TGD and ISCs. The difference is that the DCB products are more complete. The TGD and ISCs only consider the code biases between frequencies, but omit the code biases between codes within one frequency. For example, there are L1 C/A codes and L1 P codes in the GPS L1 frequency, the TGD and ISCs consider the code bias between the two frequencies as zero, while the DCB products publish this bias. This inner bias is relatively small (decimeter-level), so it can be omitted in the standard precision formulations, while it should be considered in high precision formulations.

The ZDCB is defined the same as Equation 3.23, i.e. be relative to the base frequency. The difference is that it provides complete code biases like the DCB and its base frequency varies depending on the product provider. It is commonly contained in the State-Space Representation (SSR) message and broadcasted by specific satellites or the internet.

5. Receiver code bias $d_{r,i}$.

If the between-satellite difference (BSD or DD) is applied, this item is eliminated. Otherwise, it should be handled. We will discuss this situation in the following.

The receiver code biases are also unobservable. It will be absorbed into the receiver clock offset parameter and the ionosphere delay parameter during estimation.

For single-frequency processes, the receiver code bias is directly absorbed into the receiver clock offset parameter, i.e.

$$d\bar{t}_{r,k} = dt_{r,k} + d_{r,1} \tag{3.25}$$

For double-frequency processes, the receiver code bias is absorbed into both the receiver clock offset parameter and the ionosphere delay parameter, i.e.

$$d\bar{t}_{r,k} = dt_{r,k} + d_{r,IF} \tag{3.26a}$$

$$\overline{I}^s_{r,1} = I^s_{r,1} - \frac{f_j^2}{f_i^2 - f_j^2}(d_{r,1} - d_{r,2}) \tag{3.26b}$$

For multi-frequency processes, the receiver code bias cannot be fully absorbed by the two parameters. So we should additionally estimate a bias for each frequency other than the first two frequencies. This bias is called the Inter-Frequency Bias (IFB), the definition is

$$d_{r,IFB_i} = d_{r,IF} - \frac{\gamma_i f_2^2}{f_1^2 - f_2^2}(d_{r,1} - d_{r,2}) + d_{r,i} \tag{3.27}$$

Rigorously, the above theories do not hold for GLONASS. Because the signal of GLONASS is Frequency Division Multiple Access (FDMA), the frequency differs between satellites, which leads that there are IFBs between satellites. Ideally, we can estimate the IFBs for each satellite, but it is time-consuming. In GICI, we omit the IFBs between the GLONASS satellites and give GLONASS pseudorange a relatively larger STD prior because the un-modeled IFBs will be left in the residuals.

6. Receiver clock offset $dt_{r,k}$.

The receiver clock is unique for each system, but differs from systems. That is to say, if we want to use GPS, GLONASS, BDS, and Galileo together, we should estimate four receiver clocks $dt^G_{r,k}$, $dt^R_{r,k}$, $dt^C_{r,k}$, and $dt^E_{r,k}$. To simplify, we use $dt_{r,k}$ to represent all the necessarily estimated receiver clocks in the rest of this manual.

**Formulation 1**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, dt_{r,k}\right]^T \tag{3.28}$$

Given a pseudorange measurement $P^s_{r,i,k}$ at a frequency $i$ of a satellite $s$ at epoch $k$, after correcting all the necessary errors, the residual is

$$r^s_{r,i,k} := P^s_{r,i,k} - \left\|{}^{W_G}t_k - p^s_k\right\|^2 - c\bar{dt}_{r,k} \tag{3.29}$$

Note that, according to Equation 3.25, the observable receiver clock offset is $\bar{dt}_{r,k}$, but not the true value $dt_{r,k}$, which means that the estimated receiver clock offset is biased.

The Jacobian matrix is

$$\boldsymbol{J}_k := \left[\begin{array}{cc} \boldsymbol{e}^s_{r,k} & -1 \end{array}\right] \tag{3.30}$$

where $\boldsymbol{e}^s_{r,k}$ is a normalized translation vector from the GNSS receiver to the satellite.

The covariance matrix is a diagonal matrix with the dimension of the number of pseudorange measurements. For each diagonal element, the variance is computed by [1]

$$\sigma_P^2 = F^{s2} R_r^2 \left(a_\sigma^2 + b_\sigma^2 \sin(\alpha_e)\right) + \sigma_{eph}^2 + \sigma_{ion}^2 + \sigma_{trop}^2 + \sigma_{bias}^2 \tag{3.31}$$

where $F^s$ is the satellite system error factor. $R_r$ is the code to carrier-phase error ratio. $a_\sigma$ and $b_\sigma$ are the carrier-phase error factors. The above 4 parameters can be modified by users via the "gnss_error_parameter" options. $\sigma_{eph}$, $\sigma_{ion}$, $\sigma_{trop}$, and $\sigma_{bias}$ are the STD of ephemeris error, ionosphere correction model error, troposphere correction model error, and code bias error, respectively. They are set as fixed values.

### Formulation 2

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, dt_{r,k}, I_{r,1,k}^s, T_{Z,w,k}, d_{r,IFB_i}\right]^T \tag{3.32}$$

where $d_{r,IFB_i}$ should be estimated only when the number of frequencies we are using is larger than two.

The residual is

$$r_{r,i,k}^s := P_{r,i,k}^s - \left\|{}^{W_G}t_k - p_k^s\right\|^2 - cd\bar{t}_{r,k} - \overline{I}_{r,i,k}^s - \mathcal{G}_w(\alpha_c)T_{Z,w,k} - d_{r,IFB_i} \tag{3.33}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \left[\begin{array}{ccccc} \boldsymbol{e}_{r,k}^s & -1 & -\gamma_i & -\mathcal{G}_w(\alpha_c) & -1 \end{array}\right] \tag{3.34}$$

The covariance computation is the same as formulation 1. $\sigma_{ion}$ and $\sigma_{trop}$ are set as zeros.

### Formulation 3

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^W}^T, {}^{B}\boldsymbol{t}_r^T, dt_{r,k}\right]^T \tag{3.35}$$

The residual is the same as Equation 3.29, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by

$${}^{W_G}\boldsymbol{p}_k = R_W^{W_G}({}^{W}\boldsymbol{p}_k + \boldsymbol{R}_{B,k}^W\,{}^{B}\boldsymbol{t}_r) + {}^{W_G}t_W \tag{3.36}$$

Renaming Equation 3.30 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is:

$$\boldsymbol{J}_k := \boldsymbol{J}_0 \begin{bmatrix} \boldsymbol{J}_1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.37}$$

The covariance computation is the same as formulation 1.

### Formulation 4

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^{W}}^T, {}^{B}\boldsymbol{t}_r^T, dt_{r,k}, I_{r,1,k}^s, T_{Z,w,k}, d_{r,IFB_i}\right]^T \tag{3.38}$$

The residual is the same as Equation 3.33, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.34 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is:

$$\boldsymbol{J}_k := \boldsymbol{J}_0 \begin{bmatrix} \boldsymbol{J}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3.39}$$

The covariance computation is the same as formulation 2.

### Formulation 5

After applying the SD (see section B.2), there are some differences compared with the UDUC formulations: 1) The measurements and some parameters become the SD form. 2) Satellite code bias is fully eliminated, so we do not need to correct TGDs, DCBs, or ZDCBs. 3) We think most of the ephemeris errors are eliminated. 4) We think most of the atmosphere errors are eliminated, so we do not need to apply the model correction.

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, dt_{rr_b,k}\right]^T \tag{3.40}$$

Given a SD pseudorange measurement $P_{rr_b,i,k}^s$, the residual is

$$r_{rr_b,i,k}^s := P_{rr_b,i,k}^s - \left\|{}^{W_G}t_k - p_k^s\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p_k^s\right\|^2 - c d\bar{t}_{rr_b,k} \tag{3.41}$$

where ${}^{W_G}t_{r_b,k}$ is a constant value provided by the base station.

The Jacobian matrix is the same as Equation 3.30.

The covariance matrix is a diagonal matrix with the dimension of the number of SD pseudorange measurements. For each diagonal element, the variance is computed by

$$\sigma^2_{P_{SD}} = 2F^{s2}R_r^2\left(a_\sigma^2 + b_\sigma^2 \sin\left(\alpha_e\right)\right) \tag{3.42}$$

**Formulation 6**

We estimate the remaining atmosphere errors in this formulation.

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, dt_{rr_b,k}, I^s_{rr_b,1,k}, T_{Z,w,k}, T_{Z,w,r_b,k}, d_{rr_b,IFB_i}\right]^T \tag{3.43}$$

The residual is

$$\begin{aligned} r^s_{rr_b,i,k} :=& P^s_{rr_b,i,k} - \left\|{}^{W_G}t_k - p^s_k\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p^s_k\right\|^2 - \\ & c\bar{dt}_{rr_b,k} - \bar{I}^s_{rr_b,1,k} - \mathcal{G}_w(\alpha_c)T_{Z,w,k} + \mathcal{G}_w(\alpha_c)T_{Z,w,r_b,k} - d_{rr_b,IFB_i} \end{aligned} \tag{3.44}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} \boldsymbol{e}^s_{r,k} & -1 & -\gamma_i & -\mathcal{G}_w\left(\alpha_c\right) & \mathcal{G}_w\left(\alpha_c\right) & -1 \end{bmatrix} \tag{3.45}$$

The covariance computation is the same as formulation 5.

**Formulation 7**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, \boldsymbol{q}^{W}_{B,k}{}^T, {}^{B}\boldsymbol{t}_r^T, dt_{rr_b,k}\right]^T \tag{3.46}$$

The residual is the same as Equation 3.41, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

The Jacobian matrix is the same as Equation 3.37.

The covariance computation is the same as formulation 5.

## Formulation 8

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^{W}}^T, {}^{B}\boldsymbol{t}_r^T, dt_{rr_b,k}, I_{rr_b,1,k}^s, T_{Z,w,k}, T_{Z,w,r_b,k}, d_{rr_b,IFB_i}\right]^T \tag{3.47}$$

The residual is the same as Equation 3.44, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.45 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is the same as Equation 3.39.

The covariance computation is the same as formulation 5.

## Formulation 9

Given SD measurements, the DD measurements are generated by further applying differential between satellites (see section B.2). The difference between the DD and SD measurements is: The receiver clock offset and code biases are fully eliminated (Not hold for GLONASS because the receiver code biases are not consistent between satellites, so we amplify the covariance of the DD GLONASS pseudorange).

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T\right]^T \tag{3.48}$$

Given a DD pseudorange measurement $P_{rr_b,i,k}^{ss_b}$, the residual is

$$r_{rr_b,i,k}^{ss_b} := P_{rr_b,i,k}^{ss_b} - \left\|{}^{W_G}t_k - p_k^s\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p_k^s\right\|^2 + \left\|{}^{W_G}t_k - p_k^{s_b}\right\|^2 - \left\|{}^{W_G}t_{r_b,k} - p_k^{s_b}\right\|^2 \tag{3.49}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \left[\ \boldsymbol{e}_{r,k}^s - \boldsymbol{e}_{r,k}^{s_b}\ \right] \tag{3.50}$$

The covariance matrix is a diagonal matrix with the dimension of the number of DD pseudorange measurements. For each diagonal element, the variance is computed by

$$\sigma_{P_{DD}}^2 = 4F^{s2}R_r^2\left(a_\sigma^2 + b_\sigma^2 \sin\left(\alpha_e\right)\right) \tag{3.51}$$

## Formulation 10

We estimate the remaining atmosphere errors in this formulation.

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, I_{rr_b,1,k}^{s}, I_{rr_b,1,k}^{s_b}, T_{Z,w,k}, T_{Z,w,r_b,k}\right]^T \tag{3.52}$$

The residual is

$$\begin{aligned} r_{rr_b,i,k}^{s} :=& P_{rr_b,i,k}^{ss_b} - \left\|{}^{W_G}t_k - p_k^s\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p_k^s\right\|^2 + \left\|{}^{W_G}t_k - p_k^{s_b}\right\|^2 - \left\|{}^{W_G}t_{r_b,k} - p_k^{s_b}\right\|^2 - \\ & \overline{I}_{rr_b,1,k}^{s} + \overline{I}_{rr_b,1,k}^{s_b} - \mathcal{G}_w(\alpha_c)T_{Z,w,k} + \mathcal{G}_w(\alpha_c)T_{Z,w,r_b,k} \end{aligned} \tag{3.53}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \left[\begin{array}{cccccc} \boldsymbol{e}_{r,k}^{s} - \boldsymbol{e}_{r,k}^{s_b} & -1 & 1 & -\gamma_i & -\mathcal{G}_w\left(\alpha_c\right) & \mathcal{G}_w\left(\alpha_c\right) \end{array}\right] \tag{3.54}$$

The covariance computation is the same as formulation 9.

### Formulation 11

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^{W}}^T, {}^{B}\boldsymbol{t}_r^T\right]^T \tag{3.55}$$

The residual is the same as Equation 3.49, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.50 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is the same as Equation 3.37.

The covariance computation is the same as formulation 9.

### Formulation 12

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^{W}}^T, {}^{B}\boldsymbol{t}_r^T, I_{rr_b,1,k}^{s}, I_{rr_b,1,k}^{s_b}, T_{Z,w,k}, T_{Z,w,r_b,k}\right]^T \tag{3.56}$$

The residual is the same as Equation 3.53, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.54 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is the same as Equation 3.39.

The covariance computation is the same as formulation 9.

### 3.4.2 Carrier Phase Error Factors

The carrier phase measurement can be described as

$$L_{r,i}^{s} = \rho_{r}^{s} + c\left(dt_{r} - dt^{s}\right) + I_{r,i}^{s} + T_{r}^{s} + b_{r,i} - b_{i}^{s} + N_{r,i}^{s} + \varepsilon_{L} \tag{3.57}$$

where $b_{r,i}$ and $b_i^s$ are phase biases for receiver and satellite. $N_{r,i}^s$ is the phase ambiguity, which has an integer nature.

Similar to the pseudorange measurement, there are also 12 formulations categorized by frame definitions, GNSS linear combination types, and ways to handle atmosphere delays. The 12 formulations are listed in Table 3.2.

Table 3.2: Formulations of carrier phase estimation.

| | Frame | Combination | Atmosphere | Usage in GICI |
|---|---|---|---|---|
| Formulation 1 | GRC | UDUC | Correct | |
| Formulation 2 | GRC | UDUC | Estimate | PPP |
| Formulation 3 | ICL | UDUC | Correct | |
| Formulation 4 | ICL | UDUC | Estimate | PPP-based tightly integration |
| Formulation 5 | GRC | SD | Eliminated | |
| Formulation 6 | GRC | SD | Estimate | |
| Formulation 7 | ICL | SD | Eliminated | |
| Formulation 8 | ICL | SD | Estimate | |
| Formulation 9 | GRC | DD | Eliminated | RTK |
| Formulation 10 | GRC | DD | Estimate | |
| Formulation 11 | ICL | DD | Eliminated | RTK-based tightly integration |
| Formulation 12 | ICL | DD | Estimate | |

#### Common Theories

Except for the common theories described in subsection 3.4.1, there are still some other error items that should be handled for the carrier phase.

1. Satellite phase bias $b_i^s$.

If the between-receiver difference (SD or DD) is applied, this item is eliminated. Otherwise, it should be handled. We will discuss this situation in the following.

Similar to the code bias $d_i^s$, the absolute value of phase bias $b_i^s$ is also unobservable. During estimation, it is absorbed into the ambiguity parameter $N_{r,i}^s$. If we do not want to resolve the integer ambiguity value, we do not need to care about $b_i^s$ because it just affects $N_{r,i}^s$, but not the position parameter. However, if we want to resolve the integer ambiguity, $b_i^s$ should be corrected to recover the integer nature of $N_{r,i}^s$.

There are also services for $b_i^s$ corrections (Note that the biases are also relative to base frequencies). The services can be gotten by either internet, commercial satellites or file centers.

After applying the satellite phase bias corrections, the integer ambiguity is still unrecovered. Because the corrections are relative to base frequencies and there is still a non-integer value unsolved. So a BSD should be formulated to eliminate this float value. Then, the BSD ambiguity can be used for ambiguity resolution.

2. Receiver phase bias $b_{r,i}$.

If no between-satellite difference (BSD or DD) is applied, the receiver phase bias $b_{r,i}$ will be absorbed into the ambiguity parameter $N_{r,i}^s$ during estimation, which will also break the integer nature of ambiguity. Unfortunately, $b_{r,i}$ cannot be compensated by any estimation or correction technologies. Hence, if one wants to resolve the integer ambiguity, a between-satellite difference must be applied.

3. Phase wind-up, Phase Center Offset (PCO), Phase Center Variation (PCV), earth tide.

These error items do not appear in the measurement equations because they can are not critical errors (but still affects the performance of precise GNSS algorithms) and can be corrected by models or products completely. The theories have been described in the appendix of the RTKLIB manual [1] in detail, so we will not elaborate further on these matters.

**Formulation 1**

We do not support this formulation because the residuals after correcting the atmospheric delays are far larger (meter-level) than the typical precision of carrier phase measurement (millimeter-level). Hence, the carrier phase measurement will not be effective in this situation.

#### Formulation 2

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, dt_{r,k}, I_{r,1,k}^s, T_{Z,w,k}, N_{r,i,k}^s\right]^T \tag{3.58}$$

The residual is

$$r_{r,i,k}^s := L_{r,i,k}^s - \left\|{}^{W_G}t_k - p_k^s\right\|^2 - cd\bar{t}_{r,k} - \overline{I}_{r,i,k}^s - \mathcal{G}_w(\alpha_c)T_{Z,w,k} - N_{r,i,k}^s \tag{3.59}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} \boldsymbol{e}_{r,k}^s & -1 & -\gamma_i & -\mathcal{G}_w(\alpha_c) & -1 \end{bmatrix} \tag{3.60}$$

The covariance matrix is a diagonal matrix with the dimension of the number of carrier phase measurements. For each diagonal element, the variance is computed by

$$\sigma_L^2 = F^{s2}\left(a_\sigma^2 + b_\sigma^2 \sin(\alpha_e)\right) + \sigma_{eph}^2 \tag{3.61}$$

#### Formulation 3

We do not support this formulation for the same reason as formulation 1.

#### Formulation 4

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, \boldsymbol{q}_{B,k}^{W\ T}, {}^{B}\boldsymbol{t}_r^T, dt_{r,k}, I_{r,1,k}^s, T_{Z,w,k}, N_{r,i,k}^s\right]^T \tag{3.62}$$

The residual is the same as Equation 3.59, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.60 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is:

$$\boldsymbol{J}_k := \boldsymbol{J}_0 \begin{bmatrix} \boldsymbol{J}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3.63}$$

The covariance computation is the same as formulation 2.

**Formulation 5**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, dt_{rr_b,k}, N^s_{rr_b,i,k}\right]^T \tag{3.64}$$

where $N^s_{rr_b,i,k}$ is the SD ambiguity.

The residual is

$$r^s_{rr_b,i,k} := L^s_{rr_b,i,k} - \left\|{}^{W_G}t_k - p^s_k\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p^s_k\right\|^2 - cd\bar{t}_{rr_b,k} - N^s_{rr_b,i,k} \tag{3.65}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} \boldsymbol{e}^s_{r,k} & -1 & -1 \end{bmatrix} \tag{3.66}$$

The covariance matrix is a diagonal matrix with the dimension of the number of SD carrier phase measurements. For each diagonal element, the variance is computed by

$$\sigma^2_{L_{SD}} = 2F^{s2}\left(a^2_\sigma + b^2_\sigma \sin\left(\alpha_e\right)\right) \tag{3.67}$$

**Formulation 6**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, dt_{rr_b,k}, I^s_{rr_b,1,k}, T_{Z,w,k}, T_{Z,w,r_b,k}, N^s_{rr_b,i,k}\right]^T \tag{3.68}$$

The residual is

$$\begin{aligned} r^s_{rr_b,i,k} :=& L^s_{rr_b,i,k} - \left\|{}^{W_G}t_k - p^s_k\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p^s_k\right\|^2 - \\ & cd\bar{t}_{rr_b,k} - \bar{I}^s_{rr_b,1,k} - \mathcal{G}_w(\alpha_c)T_{Z,w,k} + \mathcal{G}_w(\alpha_c)T_{Z,w,r_b,k} - N^s_{rr_b,i,k} \end{aligned} \tag{3.69}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} \boldsymbol{e}^s_{r,k} & -1 & -\gamma_i & -\mathcal{G}_w\left(\alpha_c\right) & \mathcal{G}_w\left(\alpha_c\right) & -1 \end{bmatrix} \tag{3.70}$$

The covariance computation is the same as formulation 5.

**Formulation 7**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^W\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^W}^T, {}^B\boldsymbol{t}_r^T, dt_{rr_b,k}, N_{rr_b,i,k}^s\right]^T \tag{3.71}$$

The residual is the same as Equation 3.65, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.66 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is:

$$\boldsymbol{J}_k := \boldsymbol{J}_0 \begin{bmatrix} \boldsymbol{J}_1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.72}$$

The covariance computation is the same as formulation 5.

**Formulation 8**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^W\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^W}^T, {}^B\boldsymbol{t}_r^T, dt_{rr_b,k}, I_{rr_b,1,k}^s, T_{Z,w,k}, T_{Z,w,r_b,k}, N_{rr_b,i,k}^s\right]^T \tag{3.73}$$

The residual is the same as Equation 3.69, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.70 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is the same as Equation 3.63.

The covariance computation is the same as formulation 5.

**Formulation 9**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, N_{rr_b,i,k}^s, N_{rr_b,i,k}^{s_b}\right]^T \tag{3.74}$$

where $N_{rr_b,i,k}^{s_b}$ is the SD ambiguity of the base satellite. We estimate the SD ambiguities instead of the DD ambiguities to avoid having to frequently handle base satellite switching due to signal disruption from shadowing or dropouts.

The residual is

$$r^{ss_b}_{rr_b,i,k} := L^{ss_b}_{rr_b,i,k} - \left\|{}^{W_G}t_k - p^s_k\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p^s_k\right\|^2 + \left\|{}^{W_G}t_k - p^{s_b}_k\right\|^2 - \left\|{}^{W_G}t_{r_b,k} - p^{s_b}_k\right\|^2 - N^s_{rr_b,i,k} + N^{s_b}_{rr_b,i,k} \tag{3.75}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \left[\begin{array}{c} \boldsymbol{e}^s_{r,k} - \boldsymbol{e}^{s_b}_{r,k}, -1, 1 \end{array}\right] \tag{3.76}$$

The covariance matrix is a diagonal matrix with the dimension of the number of DD carrier phase measurements. For each diagonal element, the variance is computed by

$$\sigma^2_{L_{DD}} = 4F^{s2}\left(a^2_\sigma + b^2_\sigma \sin(\alpha_e)\right) \tag{3.77}$$

**Formulation 10**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}^T_k, I^s_{rr_b,1,k}, I^{s_b}_{rr_b,1,k}, T_{Z,w,k}, T_{Z,w,r_b,k}, N^s_{rr_b,i,k}, N^{s_b}_{rr_b,i,k}\right]^T \tag{3.78}$$

The residual is

$$r^s_{rr_b,i,k} := P^{ss_b}_{rr_b,i,k} - \left\|{}^{W_G}t_k - p^s_k\right\|^2 + \left\|{}^{W_G}t_{r_b,k} - p^s_k\right\|^2 + \left\|{}^{W_G}t_k - p^{s_b}_k\right\|^2 - \left\|{}^{W_G}t_{r_b,k} - p^{s_b}_k\right\|^2 - \overline{I}^s_{rr_b,1,k} + \overline{I}^{s_b}_{rr_b,1,k} - \mathcal{G}_w(\alpha_c)T_{Z,w,k} + \mathcal{G}_w(\alpha_c)T_{Z,w,r_b,k} - N^s_{rr_b,i,k} + N^{s_b}_{rr_b,i,k} \tag{3.79}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \left[\begin{array}{ccccccccc} \boldsymbol{e}^s_{r,k} - \boldsymbol{e}^{s_b}_{r,k} & -1 & 1 & -\gamma_i & -\mathcal{G}_w(\alpha_c) & \mathcal{G}_w(\alpha_c) & -1 & 1 \end{array}\right] \tag{3.80}$$

The covariance computation is the same as formulation 9.

**Formulation 11**

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}^T_k, \boldsymbol{q}^{W}_{B,k}{}^T, {}^{B}\boldsymbol{t}^T_r, N^s_{rr_b,i,k}, N^{s_b}_{rr_b,i,k}\right]^T \tag{3.81}$$

The residual is the same as Equation 3.75, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.76 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is the same as Equation 3.72.

The covariance computation is the same as formulation 9.

#### Formulation 12

The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, \boldsymbol{q}_{B,k}^{W}{}^T, {}^{B}\boldsymbol{t}_r^T, I_{rr_b,1,k}^{s}, I_{rr_b,1,k}^{s_b}, T_{Z,w,k}, T_{Z,w,r_b,k}, N_{rr_b,i,k}^{s}, N_{rr_b,i,k}^{s_b}\right]^T \tag{3.82}$$

The residual is the same as Equation 3.79, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36.

Renaming Equation 3.80 as $\boldsymbol{J}_0$ and Equation 3.9 as $\boldsymbol{J}_1$, the Jacobian matrix is the same as Equation 3.39.

The covariance computation is the same as formulation 9.

### 3.4.3 Doppler Error Factors

The doppler error can be described as

$$D_{r,i}^{s} = \dot{\rho}_r^{s} + c\left(df_{r,k} - df^{s}\right) + \varepsilon_D \tag{3.83}$$

where $df_{r,k}$ and $df^{s}$ are the receiver and satellite clock frequency offsets.

Since the doppler is less affected by errors, there are only two typical formulations, categorized by their frames of reference. The doppler measurements are used in all the algorithms listed in Table 3.1 and Table 3.2.

#### Formulation 1

This formulation is formed under the GRC frame. The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{p}_k^T, {}^{W_G}\boldsymbol{v}_k^T, df_{r,k}\right]^T \tag{3.84}$$

Given a doppler measurement $D^s_{r,i,k}$ at a frequency $i$ of a satellite $s$ at epoch $k$, the residual is

$$r^s_{r,i,k} := D^s_{r,i,k} - \left(v^s_k - {}^{W_G}v_k\right) \cdot \left(p^s_k - {}^{W_G}t_k\right) - c\dot{df}_{r,k} \tag{3.85}$$

where $v$ represents velocity, $\cdot$ is dot product.

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} \mathbf{0} & \boldsymbol{e}^s_{r,k} & -1 \end{bmatrix} \tag{3.86}$$

The covariance matrix is a diagonal matrix with the dimension of the number of doppler measurements. For each diagonal element, the variance is computed by

$$\sigma^2_D = F^{s2}\sigma^2_D \tag{3.87}$$

where $\sigma_D$ is doppler error factor, which can be modified via the "gnss_error_parameter" options.

### Formulation 2

This formulation is formed under the ICL frame. The related parameters are

$$\boldsymbol{\chi}_k := \left[{}^W\boldsymbol{p}^T_k, {}^W\boldsymbol{v}^T_k, {\boldsymbol{q}^W_{B,k}}^T, {}^B\boldsymbol{t}^T_r, df_{r,k}\right]^T \tag{3.88}$$

The residual is the same as Equation 3.85, except that ${}^{W_G}\boldsymbol{p}_k$ should be replaced by Equation 3.36 and ${}^{W_G}\boldsymbol{v}_k$ should be replaced by

$${}^{W_G}\boldsymbol{v}_k = R^{W_G}_W\left({}^W\boldsymbol{v}_k + \lfloor\boldsymbol{\omega}_k\times\rfloor\boldsymbol{R}^W_{B,k}{}^B\boldsymbol{t}_r\right) \tag{3.89}$$

where $\boldsymbol{\omega}_k$ is the body anguler velocity.

Renaming Equation 3.86 as $\boldsymbol{J}_0$ and Equation 3.15 as $\boldsymbol{J}_1$, the Jacobian matrix is:

$$\boldsymbol{J}_k := \boldsymbol{J}_0\begin{bmatrix} \boldsymbol{J}_1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.90}$$

The covariance computation is the same as formulation 1.

## 3.5 Camera Error Factors

A feature-based visual odometry problem uses tracked features between two frames to define constraints, namely reprojection error. In GICI, we use point features to apply visual estimation. subsection 3.5.1 will introduce how the features are detected and tracked. Then, subsection 3.5.2 will illustrate how the features are used in the estimator.

### 3.5.1 Feature Detection and Tracking

We use the Features from Accelerated Segment Test (FAST) [8] method for feature detection and the Lucas-Kanade (LK) optical flow [9] method for feature tracking. Both of the two methods are famous, so we do not elaborate on them further.

After that, we have a set of tracked features. Then, we should apply triangulation to get the initial depth of the feature, and thereby the initial position of the corresponding landmarks (features expressed in the world frame). Now we can feed the landmarks and the corresponding features into the estimator.

### 3.5.2 Reprojection Error Factor

The reprojection error uses tracked features to estimate the camera pose and landmark positions. The related parameters are

$$\boldsymbol{\chi}_{c,k} := \left[ {}^{W}\boldsymbol{p}_k, \boldsymbol{q}_k^W, {}^{B}\boldsymbol{t}_c, \boldsymbol{q}_C^B, {}^{W}\boldsymbol{p_l} \right]^T \tag{3.91}$$

where ${}^{B}\boldsymbol{t}_c$ and $\boldsymbol{q}_C^B$ are the translation and orientation parts of camera extrinsic. ${}^{W}\boldsymbol{p_l}$ is a vector of landmark positions ${}^{W}\boldsymbol{p}_l$.

Given a feature ${}^{C}\tilde{\boldsymbol{p}}_l$ ($2 \times 1$ dimension), the residual can be written as

$$\boldsymbol{r}_{C,k} := {}^{C}\tilde{\boldsymbol{p}}_l - \pi_C \left( \boldsymbol{R}_B^C \left( \boldsymbol{R}_W^B \left( {}^{W}\boldsymbol{p_l} - {}^{W}\boldsymbol{p}_k \right) - {}^{B}\boldsymbol{t}_c \right) \right) \tag{3.92}$$

where $\pi_C(\cdot)$ denotes the camera projection model (which may include distortion).

The Jacobian matrix is

$$\boldsymbol{J}_k := \boldsymbol{J}_\pi \left[ \begin{array}{ccccc} \boldsymbol{R}_B^C \boldsymbol{R}_W^B & \boldsymbol{J}_{\theta_k}^p & \boldsymbol{R}_B^C & \boldsymbol{J}_{\theta_C}^p & -\boldsymbol{R}_B^C \boldsymbol{R}_W^B \end{array} \right] \tag{3.93}$$

with

$$\boldsymbol{J}_{\theta_k}^p = -\boldsymbol{R}_B^C \boldsymbol{R}_W^B \lfloor \left( {}^{W}\boldsymbol{p}_l - {}^{W}\boldsymbol{p}_k \right) \times \rfloor \tag{3.94}$$

$$\boldsymbol{J}_{\theta_C}^{p} = -\boldsymbol{R}_B^C \lfloor \left( \boldsymbol{R}_W^B \left( {}^W\boldsymbol{p}_l - {}^W\boldsymbol{p}_k \right) - {}^B\boldsymbol{t}_c \right) \times \rfloor \tag{3.95}$$

where $\boldsymbol{J}_\pi$ is the Jacobian matrix of $\pi_C(\cdot)$. It varies according to the projection model and distortion model employed. So we do not concretize it here.

The covariance is a 2-dimensional diagonal matrix. Each element is the STD of tracking error in pixel, which is defined in the "feature_error_std" option.

## 3.6 INS Error Factors

An Inertial Measurement Unit (IMU) can provide linear acceleration and angular velocity measurements, and is often used for time-propagation between states. By employing motion integration mechanics, the system is known as Inertial Navigation System (INS).

Directly applying the INS mechanics into FGO is time-consuming. This is because the formulations are dependent on estimated parameters, whose values constantly change during iteration, making it necessary to repeatedly redo the integration. In response, we use INS pre-integration [10] to isolate the integration procedure from most of the estimated states. This algorithm will be introduced in subsection 3.6.1.

A sole INS estimator suffers from drift over time due to the multi-order integration of bias and random noises. Integrating INS with other sensors is a sufficient way to mitigate the drift. But even so, the estimation of INS-relevant parameters is driven by user motion. Accuracy degradation happens if a certain axis has not been excited for long. This is not the case for free motion equipment such as Unmanned Aerial Vehicles (UAV) and handheld devices, whereas it is commonplace among constrained motion equipment like car navigators. More constraints should be added. In response, we implemented Zero Motion Update (ZUPT), Heading Measurement Constraint (HMC), and Non-Holonomic Constraint (NHC) to constrain some of the possible scenarios. These constraints will be introduced in subsection 3.6.2, subsection 3.6.3, and subsection 3.6.4, respectively.

### 3.6.1 Pre-integration Factors

The measurements of angular velocity and acceleration from the raw gyroscope and accelerometer are defined as

$$\hat{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_k + \boldsymbol{b}_{g,k} - \boldsymbol{\varepsilon}_{g,k} \tag{3.96}$$

$$\hat{\boldsymbol{a}}_k = \boldsymbol{a}_k + \boldsymbol{R}_W^B {}^W\boldsymbol{g} + \boldsymbol{b}_{a,k} - \boldsymbol{\varepsilon}_{a,k} \tag{3.97}$$

where $\hat{\boldsymbol{\omega}}_k$ and $\hat{\boldsymbol{a}}_k$ are the raw IMU measurements at epoch $k$ in $B$ frame. $\boldsymbol{b}_{g,k}$, $\boldsymbol{b}_{a,k}$ are the biases of the accelerometer and gyroscope, which are commonly modeled as random walks. $\boldsymbol{\varepsilon}_{g,k}$ and $\boldsymbol{\varepsilon}_{a,k}$ are random noises, which are commonly modeled as white noises.

The relevant estimated parameters are

$$\boldsymbol{\chi}_k := \left[{}^W\boldsymbol{p}_k^T, \boldsymbol{q}_{B,k}^{W\ T}, {}^W\boldsymbol{v}_k^T, \boldsymbol{b}_{a,k}, \boldsymbol{b}_{g,k}\right]^T \tag{3.98}$$

Ignoring the effects of the Earth rotation, the INS mechanics could be written as

$$\begin{aligned}
{}^W\dot{\boldsymbol{p}} &= {}^W\boldsymbol{v} \\
{}^W\dot{\boldsymbol{v}} &= \boldsymbol{R}_B^W(\hat{\boldsymbol{a}} - \boldsymbol{b}_a + \boldsymbol{\varepsilon}_a) - {}^W\boldsymbol{g} \\
\dot{\boldsymbol{q}}_B^W &= \frac{1}{2}\boldsymbol{q}_B^W \otimes \exp\left(\begin{bmatrix} \hat{\boldsymbol{\omega}} - \boldsymbol{b}_g + \boldsymbol{\varepsilon}_g \\ 0 \end{bmatrix}\right) \\
\dot{\boldsymbol{b}}_a &= \boldsymbol{\varepsilon}_a \\
\dot{\boldsymbol{b}}_g &= \boldsymbol{\varepsilon}_g
\end{aligned} \tag{3.99}$$

We can find that the derivatives of position ${}^W\dot{\boldsymbol{r}}$, velocity ${}^W\dot{\boldsymbol{v}}$, and quaternion $\dot{\boldsymbol{q}}_B^W$ depend on the estimated parameters ${}^W\boldsymbol{v}$, $\boldsymbol{q}_B^W$, $\boldsymbol{b}_a$, and $\boldsymbol{b}_g$. The motion parameters ${}^W\boldsymbol{v}$ and $\boldsymbol{q}_B^W$ are frequently updated with comparatively large amounts because of iteration. We must recompute the entire integration between two epochs $k$ and $k+1$ whenever there is a change. To overcome this, we transfer the integration from $W$ frame to $B_k$, the body frame at epoch $k$. Then the mechanics become

$$\begin{aligned}
{}^{B_k}\dot{\boldsymbol{p}} &= {}^{B_k}\boldsymbol{v} \\
{}^{B_k}\dot{\boldsymbol{v}} &= \boldsymbol{R}_B^{B_k}(\hat{\boldsymbol{a}}_k - \boldsymbol{b}_{a,k} + \boldsymbol{\varepsilon}_{a,k}) - {}^W\boldsymbol{g} \\
\dot{\boldsymbol{q}}_B^{B_k} &= \frac{1}{2}\boldsymbol{q}_B^{B_k} \otimes \exp\left(\begin{bmatrix} \hat{\boldsymbol{\omega}}_k - \boldsymbol{b}_{g,k} + \boldsymbol{\varepsilon}_{g,k} \\ 0 \end{bmatrix}\right) \\
\dot{\boldsymbol{b}}_{a,k} &= \boldsymbol{\varepsilon}_{a,k} \\
\dot{\boldsymbol{b}}_{g,k} &= \boldsymbol{\varepsilon}_{g,k}
\end{aligned} \tag{3.100}$$

For convenience, we use $x$ to represent body frame $B_x$ if it is written at the position of frames. Note that the position, velocity, and quaternion in $B_k$ frame, i.e. ${}^k\boldsymbol{p}$ and ${}^k\boldsymbol{v}$, and $\boldsymbol{q}_B^k$, are actually incremental variables, we rewrite them as delta values $\delta\boldsymbol{p}_B^k$, $\delta\boldsymbol{v}_B^k$, and

$\delta \boldsymbol{q}_B^k$. Then we discretize the equations

$$\begin{aligned}
\delta \hat{\boldsymbol{p}}_{k+1}^k &= \sum_{i=k}^{k+1} \left( \sum_{i=k}^{k+1} \left( \overline{\boldsymbol{R}}_i^k \left( \overline{\boldsymbol{a}}_i - \boldsymbol{b}_{a,k} + \boldsymbol{\varepsilon}_{a,k} \right) \delta t \right) \delta t \right) \\
\delta \hat{\boldsymbol{v}}_{k+1}^k &= \sum_{i=k}^{k+1} \left( \overline{\boldsymbol{R}}_i^k \left( \overline{\boldsymbol{a}}_i - \boldsymbol{b}_{a,k} + \boldsymbol{\varepsilon}_{a,k} \right) \delta t \right) \\
\delta \hat{\boldsymbol{q}}_{k+1}^k &= \prod_{i=k}^{k+1} \left( \left( \begin{bmatrix} \overline{\boldsymbol{\omega}}_i - \boldsymbol{b}_{g,k} + \boldsymbol{\varepsilon}_{g,k} \\ 0 \end{bmatrix} \right) \delta t \right)
\end{aligned} \tag{3.101}$$

where $\delta t$ is the time interval between $i$ and $i+1$. Overline $\overline{x}_i$ means an approximate describing the discrete value between $i$ and $i+1$, which depends on the numerical integration methods. We use the first-order (midpoint) method and hence $\overline{x}_i = (x_i + x_{i+1})/2$.

Now the pre-integrated measurements $\delta \hat{\boldsymbol{p}}_i^k$, $\delta \hat{\boldsymbol{v}}_i^k$, and $\delta \hat{\boldsymbol{\theta}}_i^k$ are independent from the estimated parameters ${}^W\boldsymbol{v}$ and $\boldsymbol{q}_B^W$. Note that they still depend on the bias parameters, which also change during iteration. But the amplitude is small so we can use the first-order approximation to update the measurements when they have changed. We will discuss it later.

Given a batch of IMU measurements $\boldsymbol{a}_i$ and $\boldsymbol{\omega}_i$ between two epochs $i \subseteq [k, k+1]$, we can compute Equation 3.101 recursively by

$$\begin{aligned}
\delta \hat{\boldsymbol{p}}_{i+1}^k &= \delta \hat{\boldsymbol{p}}_i^k + \delta \hat{\boldsymbol{v}}_i^k \delta t + \frac{1}{2} \overline{\boldsymbol{R}}_i^k \left( \overline{\boldsymbol{a}}_i - \boldsymbol{b}_{a,k} + \boldsymbol{\varepsilon}_{a,k} \right) \delta t^2 \\
\delta \hat{\boldsymbol{v}}_{i+1}^k &= \delta \hat{\boldsymbol{v}}_i^k + \overline{\boldsymbol{R}}_i^k \left( \overline{\boldsymbol{a}}_i - \boldsymbol{b}_{a,k} + \boldsymbol{\varepsilon}_{a,k} \right) \delta t \\
\delta \hat{\boldsymbol{q}}_{i+1}^k &= \delta \hat{\boldsymbol{q}}_i^k \otimes \frac{1}{2} \exp \left( \begin{bmatrix} \overline{\boldsymbol{\omega}}_i - \boldsymbol{b}_{g,k} + \boldsymbol{\varepsilon}_{g,k} \\ 0 \end{bmatrix} \delta t \right)
\end{aligned} \tag{3.102}$$

Now we discuss the covariance propagation. We linearize Equation 3.100 to get the

$\dot{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}\boldsymbol{\varepsilon}$ form

$$\begin{aligned}\begin{bmatrix} \delta\dot{\boldsymbol{p}}^k \\ \delta\dot{\boldsymbol{v}}^k \\ \delta\dot{\boldsymbol{\theta}}^k \\ \delta\dot{\boldsymbol{b}}_{a,k} \\ \delta\dot{\boldsymbol{b}}_{g,k} \end{bmatrix} &= \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\lfloor \boldsymbol{R}^k(\hat{\boldsymbol{a}} - \boldsymbol{b}_a)\times\rfloor & -\boldsymbol{R}^k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\lfloor(\hat{\boldsymbol{\omega}} - \boldsymbol{b}_g)\times\rfloor & \mathbf{0} & -\boldsymbol{R}^k \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{p}^k \\ \delta\boldsymbol{v}^k \\ \delta\boldsymbol{\theta}^k \\ \delta\boldsymbol{b}_{a,k} \\ \delta\boldsymbol{b}_{g,k} \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{R}^k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varepsilon}_{a,k} \\ \boldsymbol{\varepsilon}_{g,k} \\ \boldsymbol{\varepsilon}_{b_a,k} \\ \boldsymbol{\varepsilon}_{b_g,k} \end{bmatrix}\end{aligned} \tag{3.103}$$

We discretize the above continuous equation by first-order approximation, i.e. $\exp(\boldsymbol{F}\delta t) = \boldsymbol{I} + \boldsymbol{F}\delta t$. Then we can get a recursive equation to update the covariance

$$\boldsymbol{P}^k_{i+1} = (\boldsymbol{I} + \boldsymbol{F}_i\delta t)\left(\boldsymbol{P}^k_i + \boldsymbol{G}_i\boldsymbol{Q_i}\boldsymbol{G}_i^T\right)(\boldsymbol{I} + \boldsymbol{F}_i\delta t)^T \tag{3.104}$$

where $\boldsymbol{Q_i}$ is the covariance of $\boldsymbol{\varepsilon}_i$.

The covariance matrix is computed recursively during integration and the final result $\boldsymbol{P}^k_{k+1}$ will be used as measurement covariance when add the pre-integration measurement into the graph.

As mentioned before, the pre-integration measurements are still dependent on the bias parameters. If the biases have been updated, the pre-integration measurements should be updated correspondingly via

$$\begin{aligned} \delta\hat{\boldsymbol{p}}^{k\prime}_{k+1} &\approx \delta\hat{\boldsymbol{p}}^k_{k+1} + \boldsymbol{J}^p_{b_a}\delta\boldsymbol{b}_{a,k} + \boldsymbol{J}^p_{b_g}\delta\boldsymbol{b}_{g,k} \\ \delta\hat{\boldsymbol{v}}^{k\prime}_{k+1} &\approx \delta\hat{\boldsymbol{v}}^k_{k+1} + \boldsymbol{J}^v_{b_a}\delta\boldsymbol{b}_{a,k} + \boldsymbol{J}^v_{b_g}\delta\boldsymbol{b}_{g,k} \\ \delta\hat{\boldsymbol{q}}^{k\prime}_{k+1} &\approx \delta\hat{\boldsymbol{q}}^k_{k+1} \otimes \frac{1}{2}\exp\left(\begin{bmatrix} \boldsymbol{J}^\theta_{b_g}\delta\boldsymbol{b}_{g,k} \\ 0 \end{bmatrix}\right) \end{aligned} \tag{3.105}$$

where $\boldsymbol{J}^a_b$ is the Jacobian matrix of $a$ over $b$.

A straightforward method to compute the above Jacobians is to apply the recursive equation $\boldsymbol{\Phi}_{i+1}^{k} = (\boldsymbol{I} + \boldsymbol{F}_i)\,\boldsymbol{\Phi}_{i}^{k}$ to get $\boldsymbol{\Phi}_{k+1}^{k}$. Then the Jacobians can be gotten from the corresponding matrix blocks. However, to alleviate the computational load, we follow [10] to compute the Jacobians in a more analytical way. Taking partial derivation of Equation 3.101, the Jacobians can be computed as

$$\begin{aligned}
\boldsymbol{J}_{b_g,k}^{\theta} &= -\sum_{i=k}^{k+1}\left(\boldsymbol{R}_i^{k+1^T}\boldsymbol{J}_{r,i}\delta t\right)\\
\boldsymbol{J}_{b_g,k}^{v} &= -\sum_{i=k}^{k+1}\left(\overline{\boldsymbol{R}}_i^{k}\lfloor(\overline{\boldsymbol{a}}_i - \boldsymbol{b}_{a,k})\times\rfloor\boldsymbol{J}_{b_g,i}^{\theta}\delta t\right)\\
\boldsymbol{J}_{b_g,k}^{p} &= -\sum_{i=k}^{k+1}\left(\sum_{i=k}^{k+1}\left(\overline{\boldsymbol{R}}_i^{k}\lfloor(\overline{\boldsymbol{a}}_i - \boldsymbol{b}_{a,k})\times\rfloor\boldsymbol{J}_{b_g,i}^{\theta}\delta t\right)\delta t\right)\\
\boldsymbol{J}_{b_a,k}^{v} &= -\sum_{i=k}^{k+1}\left(\overline{\boldsymbol{R}}_i^{k}\delta t\right)\\
\boldsymbol{J}_{b_a,k}^{p} &= -\sum_{i=k}^{k+1}\left(\sum_{i=k}^{k+1}\left(\overline{\boldsymbol{R}}_i^{k}\delta t\right)\delta t\right)
\end{aligned} \tag{3.106}$$

where $\boldsymbol{J}_{r,i} = \boldsymbol{J}_r\left(\overline{\boldsymbol{\omega}}_i - \boldsymbol{b}_{g,k}\right)$, $\boldsymbol{J}_r(\cdot)$ is the right Jacobian of SO(3),

$$\boldsymbol{J}_r(\boldsymbol{\theta}) = \boldsymbol{I} - \frac{1-\cos(\|\boldsymbol{\theta}\|)}{\|\boldsymbol{\theta}\|^2}\lfloor\boldsymbol{\theta}\times\rfloor + \frac{\|\boldsymbol{\theta}\| - \sin(\|\boldsymbol{\theta}\|)}{\|\boldsymbol{\theta}^3\|}\lfloor\boldsymbol{\theta}\times\rfloor^2 \tag{3.107}$$

Now we get the pre-integrated measurements. The residual can be written as

$$\boldsymbol{r}^k_{I,k+1} := \begin{bmatrix} \delta\hat{\boldsymbol{p}}^k_{k+1} \\ \delta\hat{\boldsymbol{v}}^k_{k+1} \\ \delta\hat{\boldsymbol{q}}^k_{k+1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \boxminus \begin{bmatrix} \boldsymbol{R}^{W^T}_k \left( {}^W\boldsymbol{p}_{k+1} - {}^W\boldsymbol{p}_k - {}^W\boldsymbol{v}_k\delta t + \frac{1}{2}{}^W\boldsymbol{g}\delta t^2 - \boldsymbol{J}^p_{b_a}\delta\boldsymbol{b}_{a,k} - \boldsymbol{J}^p_{b_g}\delta\boldsymbol{b}_{g,k} \right) \\ \boldsymbol{R}^{W^T}_k \left( {}^W\boldsymbol{v}_{k+1} - {}^W\boldsymbol{v}_k + {}^W\boldsymbol{g}\delta t - \boldsymbol{J}^v_{b_a}\delta\boldsymbol{b}_{a,k} - \boldsymbol{J}^v_{b_g}\delta\boldsymbol{b}_{g,k} \right) \\ \boldsymbol{q}^{W^{-1}}_k \otimes \boldsymbol{q}^W_{k+1} \otimes \frac{1}{2}\exp\left( \begin{bmatrix} \boldsymbol{J}^\theta_{b_g}\delta\boldsymbol{b}_{g,k} \\ 0 \end{bmatrix} \right)^{-1} \\ \boldsymbol{b}_{a,k+1} - \boldsymbol{b}_{a,k} \\ \boldsymbol{b}_{g,k+1} - \boldsymbol{b}_{g,k} \end{bmatrix} \tag{3.108}$$

The Jacobian matrix over the parameters at epoch $k$ is

$$\boldsymbol{J}_k := \begin{bmatrix} \boldsymbol{R}^{W^T}_k & \boldsymbol{R}^{W^T}_k\delta t & \boldsymbol{J}^p_{\theta,k} & \boldsymbol{J}^p_{b_a,k} & \boldsymbol{J}^p_{b_g,k} \\ \mathbf{0} & \boldsymbol{R}^{W^T}_k & \boldsymbol{J}^v_{\theta,k} & \boldsymbol{J}^v_{b_a,k} & \boldsymbol{J}^v_{b_g,k} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{J}^\theta_{\theta,k} & \mathbf{0} & \boldsymbol{J}^{\theta\,\prime}_{b_g,k} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} \end{bmatrix} \tag{3.109}$$

with

$$\begin{aligned} \boldsymbol{J}^p_{\theta,k} &= \lfloor \boldsymbol{R}^{W^T}_k \left( {}^W\boldsymbol{p}_{k+1} - {}^W\boldsymbol{p}_k - {}^W\boldsymbol{v}_k\delta t + \frac{1}{2}{}^W\boldsymbol{g}\delta t^2 \right) \times \rfloor \\ \boldsymbol{J}^v_{\theta,k} &= \lfloor \boldsymbol{R}^{W^T}_k \left( {}^W\boldsymbol{v}_{k+1} - {}^W\boldsymbol{v}_k + {}^W\boldsymbol{g}\delta t \right) \times \rfloor \\ \boldsymbol{J}^\theta_{\theta,k} &= \left( \boldsymbol{Q}^+ \left( \delta\hat{\boldsymbol{q}}^k_{k+1} \otimes \boldsymbol{q}^{W\ -1}_{k+1} \right) \boldsymbol{Q}^- \left( \boldsymbol{q}^W_k \right) \right)_{3\times 3} \\ \boldsymbol{J}^{\theta\,\prime}_{b_g,k} &= \left( \boldsymbol{Q}^+ \left( \delta\hat{\boldsymbol{q}}^k_{k+1} \right) \boldsymbol{Q}^- \left( \boldsymbol{q}^{W\ -1}_{k+1} \otimes \boldsymbol{q}^W_k \right) \right)_{3\times 3} \boldsymbol{J}^\theta_{b_g,k} \end{aligned} \tag{3.110}$$

where the subscript $3 \times 3$ represents the corresponding matrix block at the top left.

The Jacobian matrix over the parameters at epoch $k+1$ is

$$\boldsymbol{J}_{k+1} := \begin{bmatrix} -\boldsymbol{R}_k^{W^T} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{R}_k^{W^T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{J}_{\theta,k+1}^{\theta} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{I} \end{bmatrix} \tag{3.111}$$

with

$$\boldsymbol{J}_{\theta,k+1}^{\theta} = -\left(\boldsymbol{Q}^{+}\left(\delta\hat{\boldsymbol{q}}_{k+1}^{k} \otimes \boldsymbol{q}_{k+1}^{W^{-1}}\right)\boldsymbol{Q}^{-}\left(\boldsymbol{q}_k^W\right)\right)_{3\times 3} \tag{3.112}$$

## 3.6.2 Zero Motion Update Factor

The ZUPT detects user static motion and adds a pseudo-measurement to constrain the zero motion.

There are several ways to detect static motions. A straightforward way is to examine a window of IMU measurements, if there is no acceleration or angular velocity larger than a threshold, we think we are under a static motion. In the multi-sensor fusion system, we can further use the information from the other sensors to make the judgment more reliable. For example, we can use GNSS velocity and position and camera feature disparity to examine the motion. In GICI, we only used the IMU measurement examination for static motion detection.

Once a static motion is detected, the ZUPT constraint can be added by a pseudo-measurement

$$\boldsymbol{r}_k := {}^{W}\boldsymbol{v}_k \tag{3.113}$$

Hence, the related parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W}\boldsymbol{v}_k\right]^T \tag{3.114}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \boldsymbol{I} \tag{3.115}$$

The covariance matrix is a diagonal matrix with comparably small variance values.

### 3.6.3 Heading Measurement Constraint Factor

If there is a heading measurement, it can be constrained to the estimator to improve the accuracy and stability of the yaw estimation. For the GICI system, the heading can be gotten from a dual-antenna GNSS measurement or computed from GNSS velocity. Currently, we do not support dual-antenna GNSS. For the GNSS velocity-derived heading measurement, we set an option to enable it, i.e. "car_motion". Note that computing heading angle from GNSS velocity just valid on the fix-direction-movement vehicles, such as cars and fixed-wing aircraft. We just implemented the HMC for car motion. Rather than using the GNSS velocity to compute the heading, we use the estimated velocity, which makes the constraint more tight.

The HMC residual can be computed as

$$\boldsymbol{r}_k := \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \cdot 2\log\left(\hat{\boldsymbol{q}}_k^{W-1} \otimes \boldsymbol{q}_k^W\right) \tag{3.116}$$

with

$$\hat{\boldsymbol{q}}_k^W = \exp\begin{pmatrix} \frac{1}{2}\boldsymbol{\theta} \\ 0 \end{pmatrix} \tag{3.117}$$

where $\boldsymbol{\theta} = [0, 0, \theta_z]$, $\theta_z$ is the heading measurement.

with

$$\theta_z = \arctan\left(\frac{{}^W v_{x,k}}{{}^W v_{y,k}}\right) \tag{3.118}$$

Hence, the related parameters are

$$\boldsymbol{\chi}_k := \left[\boldsymbol{q}_k^W, {}^W\boldsymbol{v}_k\right]^T \tag{3.119}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} \boldsymbol{J}_\theta^{\theta_z} & \boldsymbol{J}_v^{\theta_z} \end{bmatrix} \tag{3.120}$$

with

$$\boldsymbol{J}_\theta^{\theta_z} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \cdot \boldsymbol{Q}^+\left(2\log\left(\hat{\boldsymbol{q}}_k^{W-1} \otimes \boldsymbol{q}_k^W\right)\right) \tag{3.121}$$

$$\boldsymbol{J}_v^{\theta_z} = \frac{1}{\sqrt{{}^W v_{x,k}^2 + {}^W v_{y,k}^2}} \begin{bmatrix} -{}^W v_{y,k} & {}^W v_{x,k} & 0 \end{bmatrix} \tag{3.122}$$

The covariance matrix is a scalar value, which is given by

$$\sigma_H^2 = \sigma_h^2 + \sigma_a^2 \tag{3.123}$$

where $\sigma_h$ is the STD of heading measurement error. $\sigma_a$ is the STD of installation angle error.

### 3.6.4 Non-holonomic Constraint Factor

The NHC assumes that the vehicle moves along one body axis and the velocity on the other two body axes are zeros. This feature is also configured by the "car_motion" option in GICI.

By assuming that the motion is along the y-axis, the NHC residual can be written as

$$\boldsymbol{r}_k := \boldsymbol{J}_l \boldsymbol{R}_k^{W^T W} \boldsymbol{v}_k \tag{3.124}$$

with

$$\boldsymbol{J}_l = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.125}$$

Hence, the related parameters are

$$\boldsymbol{\chi}_k := \left[\boldsymbol{q}_k^W, {}^W\boldsymbol{v}_k\right]^T \tag{3.126}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \boldsymbol{J}_l \begin{bmatrix} \boldsymbol{R}_k^{W^T} \lfloor {}^W\boldsymbol{v}_k \times \rfloor & \boldsymbol{R}_k^{W^T} \end{bmatrix} \tag{3.127}$$

The covariance matrix is a 2-dimensional diagonal matrix. Each element is given as

$$\sigma_N^2 = \sigma_a^2 \tag{3.128}$$

where $\sigma_a$ is the STD of installation angle error.

## 3.7 Common Factors

### 3.7.1 Parameter Error Factors

The parameter error factor constrains the parameter values directly. It is often used to give an initial guess of a parameter, such as the initial GNSS ambiguity, ionosphere delay, extrinsic, etc, or an external constraint from outer solvers, such as the fixed GNSS ambiguities.

For an estimated parameter $\boldsymbol{\chi}_k$, the parameter error residual is defined as

$$\boldsymbol{r}_k := \hat{\boldsymbol{\chi}}_k - \boldsymbol{\chi}_k \tag{3.129}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \boldsymbol{I} \tag{3.130}$$

The covariance matrix is set according to the covariance of the confidence of the initial guess.

### 3.7.2 Relative Constant Error Factors

The relative constant error factor constrains the parameter values between epochs. It is used to connect the non-INS-propagated parameters between epochs, such as the GNSS ambiguity, ionosphere delay, etc.

For an estimated parameter at two epochs $\boldsymbol{\chi}_k$ and $\boldsymbol{\chi}_{k+1}$, the relative const error residual is defined as

$$\boldsymbol{r}_k := \boldsymbol{\chi}_{k+1} - \boldsymbol{\chi}_k \tag{3.131}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \tag{3.132}$$

The covariance matrix is a diagonal matrix with the dimension of the number of parameters. For each diagonal element, the variance is computed by

$$\sigma_\delta = q^2 \delta t \tag{3.133}$$

where $q$ is the Power Spectral Density (PSD) of the corresponding parameter. $\delta t$ is the time step between the two epochs.

### 3.7.3 Relative Integral Error Factors

The relative integral factor constrains the integral parameters values between epochs. It is used to connect the non-INS-propagated integral parameters between epochs, such as the position-velocity pair, clock-frequency pair, etc.

Taking the velocity and position model as an example. For estimated parameters at two epochs $\boldsymbol{p}_k$, $\boldsymbol{v}_k$, $\boldsymbol{p}_{k+1}$, and $\boldsymbol{v}_{k+1}$, the relative integral error residual is defined as

$$\boldsymbol{r}_k := \begin{bmatrix} \boldsymbol{p}_{k+1} - \boldsymbol{p}_k - \boldsymbol{v}_k \delta t \\ \boldsymbol{v}_{k+1} - \boldsymbol{v}_k \end{bmatrix} \tag{3.134}$$

The Jacobian matrix is

$$\boldsymbol{J}_k := \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{I} & -\boldsymbol{I}\delta t & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \tag{3.135}$$

The covariance matrix is

$$\boldsymbol{Q}_k := \begin{bmatrix} \frac{1}{3}\boldsymbol{I} q_v \delta t^3 & \frac{1}{2}\boldsymbol{I} q_v \delta t^2 \\ \frac{1}{2}\boldsymbol{I} q_v \delta t^2 & \boldsymbol{I} q_v \delta t \end{bmatrix} \tag{3.136}$$

## 3.8 GNSS Estimators

### 3.8.1 Single Point Positioning

The SPP algorithm utilizes one epoch ZD pseudorange and doppler measurements to solve the receiver position, velocity, and clock. The graph structure is shown in Figure 3.5.

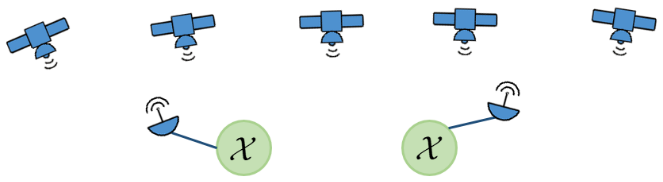


Figure 3.5: FGO structure of GNSS SPP.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, {}^{W_G}\boldsymbol{v}_k^T, d\boldsymbol{t}_{r,k}, d\boldsymbol{f}_{r,k}\right]^T \tag{3.137}$$

where the boldsymbols $d\boldsymbol{t}_{r,k}$ and $d\boldsymbol{f}_{r,k}$ means that we estimate clocks and frequencies for each satellite systems.

There are two typical SPP algorithms: single frequency SPP and ionosphere combination SPP. We implemented the former one in GICI. The absent of multi-frequency SPP is because one have to further estimate the IFB items (see subsection 3.4.1) and hence multi-frequency cannot bring comparably better performance for SPP.

The edges in the graph contain multiple residuals of pseudorange and doppler measurements, which correspond to formulation 1 in subsection 3.4.1 and formulation 1 in subsection 3.4.3.

### 3.8.2 Real-Time Differential

The RTD algorithm utilizes one epoch DD pseudornage and ZD doppler measurements to solve the receiver position and velocity. The graph structure is the same as Figure 3.5.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, {}^{W_G}\boldsymbol{v}_k^T, d\boldsymbol{f}_{r,k}\right]^T \tag{3.138}$$

Unlike SPP, the multi-frequency measurements can be used in RTD because the IFB items are eliminated by the DD operation. So users can use any number of frequencies to run RTD by controlling the input stream of GICI estimator node.

The edges contain DD pseudorange and ZD doppler measurements, which correspond to formulation 9 in subsection 3.4.1 and formulation 1 in subsection 3.4.3.

### 3.8.3 Real-Time Kinematic

The RTK algorithm utilizes multi-epoch DD pseudorange, ZD doppler, and DD carrier phase measurements to solve the receiver position and velocity. The graph structure is shown in Figure 3.6.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, {}^{W_G}\boldsymbol{v}_k^T, d\boldsymbol{f}_{r,k}, \boldsymbol{N}_{rr_b,i,k}^s\right]^T \tag{3.139}$$

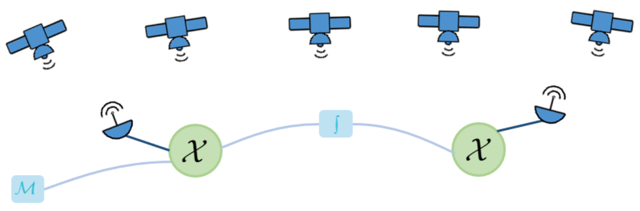


Figure 3.6: FGO structure of GNSS RTK.

where $\boldsymbol{N}^{s}_{rr_b,i,k}$ is a vector of SD ambiguities $N^{s}_{rr_b,i,k}$. If there are $n$ matched satellites between the rover and base receivers, and $m$ matched frequencies for each satellite, there will be $m \times n$ SD ambiguities $N^{s}_{rr_b,i,k}$ should be estimated.

Now we illustrate the edges defined in the graph.

For the receiver edge, the containers are:

1) DD pseudorange measurements that correspond to formulation 9 in subsection 3.4.1.

2) DD carrier phase measurements that correspond to formulation 9 in subsection 3.4.2.

3) ZD doppler measurements that correspond to formulation 1 in subsection 3.4.3.

4) Initial ambiguity guesses that correspond to subsection 3.7.1.

5) Fixed ambiguity constraints that correspond to subsection 3.7.1.

For the epoch inter-connection edge, the containers are:

1) Relative position and velocity error that corresponds to subsection 3.7.3.

2) Relative clock frequency errors that correspond to subsection 3.7.2.

3) Relative ambiguity errors that correspond to subsection 3.7.2.

For the marginalization edge, the corresponding residuals and Jacobians are defined in subsection 3.1.2.

Despiting the FGO, there is also an extra process for RTK: Ambiguity Resolution (AR). AR is a major procedure for high-precision GNSS algorithms to achieve centimeter-level solutions.

The AR starts with the estimated float ambiguities $\boldsymbol{N}^{s}_{rr_b,i,k}$ and their covariance matrix $\boldsymbol{P}_{N_{SD}}$. A Between-Satellite-Difference (BSD) is first applied by selecting base satellites for each satellite constellation to eliminate the receiver phase biases $b_{r,i}$ absorbed by float ambiguities during estimation. Herein we get the DD ambiguities $\boldsymbol{N}^{ss_b}_{rr_b,i,k}$ and covariance matrix $\boldsymbol{P}_{N_{DD}}$. For convenience, we represent the DD ambiguities and their covariance as $\boldsymbol{N}$ and $\boldsymbol{P}$. Then the problem becomes solving an integer LSQ problem

$$\hat{\boldsymbol{N}} = \arg\min_{\hat{\boldsymbol{N}}} \|\hat{\boldsymbol{N}} - \boldsymbol{N}\|_P^2 \tag{3.140}$$

where $\hat{\boldsymbol{N}}$ is the integer ambiguities to be solved.

The problem can be solved by two steps: decorrelation and search. We use the MLAMBDA [11] algorithm to conduct the resolution. After the integer ambiguities have been solved, they will be constrained to FGO using the parameters error factors with low variances.

To further improve the fixation rate and reliability, we use partial ambiguity resolution and (ultra-) wide-lane combination technologies.

The wide-line combination is shown in section B.1. This combination can widen the wavelength of the measurements from the raw wavelength $\sim$ 0.2 m to about 0.7 $\sim$ 6 m. The wider the wavelength, the lower the noise amplitude mapped on the cycle of the ambiguity parameters, and hence the easier the integer ambiguity can be solved. In GICI, we define the wide-lane ambiguity as the combined ambiguities with the wavelength within 0.3 $\sim$ 2 m, the ultra-wide-lane as $>$ 2 m, and the narrow-lane as $<$ 0.3 m. Combining the ambiguities with similar frequencies yields ultra-wide-lane ambiguities, such as combining GPS L2 (1227.6 MHz) and L5 (1176.45 MHz) yields $\sim$ 5 m wavelength. And combining the ambiguities with farther frequencies yields wide-lane ambiguities, such as combining GPS L1 (1572.42 MHz) and L2 yields $\sim$ 0.8 m wavelength. Before applying the MLAMBDA algorithm, we combine the ambiguities according to the frequencies of candidates. After that, we prioritize solving the ultra-wide-lane ambiguities, then the wide-lane ambiguities, and finally the narrow-lane ambiguities. Every time when a batch of ultra-wide-lane or wide-lane ambiguities is solved, their integer value will be constrained into the float ambiguities can their covariance matrix by a Kalman update

$$\boldsymbol{N}^{'} = \boldsymbol{N} + \boldsymbol{K}_N \left(\hat{\boldsymbol{N}} - \boldsymbol{J}_N \boldsymbol{N}\right) \tag{3.141}$$

$$\boldsymbol{P}^{'} = \left(\boldsymbol{I} - \boldsymbol{K}_N \boldsymbol{J}_N\right) \boldsymbol{P} \tag{3.142}$$

with

$$\boldsymbol{K}_N = \boldsymbol{P}\boldsymbol{J}^T \left(\boldsymbol{J}_N \boldsymbol{P} \boldsymbol{J}^T + \boldsymbol{P}_{\hat{N}}\right)^{-1} \tag{3.143}$$

where $\boldsymbol{J}_N$ is the Jacobian matrix that contains the combination coefficients. $\boldsymbol{P}_{\hat{N}}$ is the covariance of the fixed ambiguities, which is a diagonal matrix with the diagonal element set as a very small value (0.001 cycles in default).

After the final narrow-lane ambiguities are solved, we try to add the ambiguity constraints into the graph. If we find that the total cost decreases, we think the solution has better optimality, and we adopt this ambiguity resolution. Or we will reject the current resolution and remain with the float solution.

For each lane type, the partial ambiguity resolution strategy is applied. The partial ambiguity resolution strategy is to solve a subset of integer ambiguities, instead of the whole. This is more practical because some of the float ambiguities may be affected by errors and probability cannot be solved. We first arrange the float ambiguities according to the satellite elevation, parameter variance, and amplitude of the fractional part, respectively. Then we try to solve the subset of ambiguities for each sequence by erasing the ambiguity at the back recursively until the integer ambiguity is solved or we reach the minimum percentage limit.

### 3.8.4 Precise Point Positioning

The PPP algorithm utilizes multi-epoch ZD pseudorange, ZD doppler, and ZD carrier phase measurements to solve the receiver position, velocity, clock, and atmospheric delays. The graph structure is the same as Figure 3.6.

Note that there are two typical PPP algorithms: ionosphere-free PPP and undifferenced and uncombined PPP. The performance of the two algorithms is similar. We choose to use the latter one because it is more flexible to handle multi-frequency measurements.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := \left[{}^{W_G}\boldsymbol{p}_k^T, {}^{W_G}\boldsymbol{v}_k^T, d\boldsymbol{t}_{r,k}, d\boldsymbol{f}_{r,k}, \boldsymbol{N}_{r,i,k}^s, T_{Z,w,k}, \boldsymbol{I}_{r,1,k}^s, \boldsymbol{d}_{r,IFB_i}\right]^T \tag{3.144}$$

where $d\boldsymbol{t}_{r,k}$ and $d\boldsymbol{f}_{r,k}$ are vectors of receiver clocks and frequencies $dt_{r,k}$ and $df_{r,k}$. $\boldsymbol{N}_{r,i,k}^s$ is a vector of ZD ambiguities $N_{r,i,k}^s$. $\boldsymbol{I}_{r,1,k}^s$ is a vector of ionosphere delays $I_{r,1,k}^s$. $\boldsymbol{d}_{r,IFB_i}$ is a vector of inter-frequency biases $d_{r,IFB_i}$. If there are $n_S$ satellite systems, containing $n$ satellites, and for each satellite, there are $m$ frequencies. There will be $n_S$ receiver clocks and frequencies $dt_{r,k}$ and $df_{r,k}$, $m \times n$ ZD ambiguities $N_{r,i,k}^s$, $n$ ionosphere delays $I_{r,1,k}^s$, and $n_S(m-2)$ inter-frequency biases $d_{r,IFB_i}$ should be estimated.

Now we illustrate the edges defined in the graph.

For the receiver edge, the containers are:

1) ZD pseudorange measurements that correspond to formulation 2 in subsection 3.4.1.

2) ZD carrier phase measurements that correspond to formulation 2 in subsection 3.4.2.

3) ZD doppler measurements that correspond to formulation 1 in subsection 3.4.3.

4) Initial ambiguity guesses that correspond to subsection 3.7.1.

5) Initial troposphere wet delay that correspond to subsection 3.7.1.

6) Initial ionosphere delay that correspond to subsection 3.7.1.

7) Initial IFB that correspond to subsection 3.7.1.

8)$^*$ Fixed ambiguity constraints that correspond to subsection 3.7.1. This is an extra constraint for PPP with Ambiguity Resolution (PPP-AR)

9)$^*$ Troposphere wet delay correction that correspond to subsection 3.7.1. This is an extra constraint for PPP with reference network corrections (PPP-RTK)

10)$^*$ Ionosphere delay correction that correspond to subsection 3.7.1. This is an extra constraint for PPP-RTK.

where $^*$ means we do not support the corresponding features currently because there are no stable standard open services for these corrections yet.

For the epoch inter-connection edge, the containers are:

1) Relative position and velocity error that corresponds to subsection 3.7.3.

2) Relative clock and frequency error that corresponds to subsection 3.7.3.

3) Relative ambiguity errors that correspond to subsection 3.7.2.

4) Relative troposphere wet delay error that corresponds to subsection 3.7.2.

5) Relative ionosphere delay errors that correspond to subsection 3.7.2.

For the marginalization edge, the corresponding residuals and Jacobians are defined in subsection 3.1.2.

We implemented the PPP-AR. But this feature has not been fully tested. To conduct the PPP-AR, one must access a satellite phase bias service and correct it. Then, by extracting the ZD float ambiguities from the PPP estimator, one should first apply BSD to eliminate the receiver phase biases and the unobservable part of the satellite phase biases. Then almost the same AR algorithm as RTK can be applied to solve integer ambiguities and constrain them back into the PPP estimator. The difference is that the PPP float ambiguities absorb more noise and hence the strategy should be slightly modified to ensure reliable and fast AR. Here we will not elaborate further.

### 3.8.5 Global Frame Initialization

As described above, we conduct the GNSS-only estimators in the ECEF global frame. However, we realize multi-sensor fusion algorithms in the ENU. frame. Hence, global-to-local frame conversions should be applied frequently and the datum should be defined initially.

The global frame initialization defines a reference point to convert states from ECEF global frame to ENU local frame. The reference point is also used to compute the local gravity, which is used by INS mechanics. There are two ways to set the reference point: 1) We run an SPP estimator and set the reference point as the first valid solution of the estimator. 2) We load the point coordinate from the configuration file via the "initial_global_position" option if the "force_initial_global_position" option is set as true.

## 3.9 GNSS/INS Integrated Estimators

### 3.9.1 Loosely Integration

The LC estimator utilizes solution from GNSS (position and velocity), and raw measurements from INS (acceleration and angular velocity), for estimation. The FGO structure of the GNSS/INS LC estimator is shown in Figure 3.7.

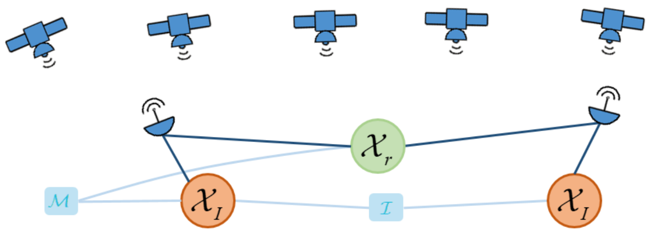


Figure 3.7: FGO structure of GNSS/INS estimator.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := [\boldsymbol{\chi}_{I,k}, \boldsymbol{\chi}_r]^T \tag{3.145}$$

with

$$\boldsymbol{\chi}_{I,k} := \left[{}^W\boldsymbol{p}_k^T, {\boldsymbol{q}_{B,k}^W}^T, {}^W\boldsymbol{v}_k^T, \boldsymbol{b}_{a,k}, \boldsymbol{b}_{g,k}\right]^T \tag{3.146}$$

$$\boldsymbol{\chi}_r := \left[{}^B\boldsymbol{t}_r^T\right]^T \tag{3.147}$$

The GNSS receiver edge contains:

1) GNSS position error that corresponds to the INS-centered local frame formulation in subsection 3.3.1.

2) GNSS velocity error that corresponds to the INS-centered local frame formulation in subsection 3.3.2.

3) Initial GNSS extrinsic error that corresponds to subsection 3.7.1.

The IMU inter-connect edge contains the INS pre-integration error that corresponds to subsection 3.6.1.

There are also some optional constraints applied on the INS parameters $\boldsymbol{\chi}_{I,k}$, containing:

1) ZUPT error in subsection 3.6.2.

2) HMC error in subsection 3.6.3. (Only for car motion.)

3) NHC error in subsection 3.6.4. (Only for car motion.)

### 3.9.2 Tightly Integration

The TC estimator utilizes GNSS raw measurements from GNSS (pseudorange, carrier phase, and doppler), and raw measurements from INS (acceleration and angular velocity), for estimation. The FGO structure of the GNSS/INS TC estimator is shown in Figure 3.8.

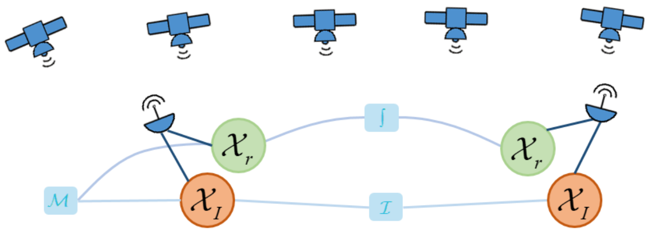


Figure 3.8: FGO structure of GNSS/INS estimator.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := \left[\boldsymbol{\chi}_{I,k}, \boldsymbol{\chi}_{r,k}\right]^T \tag{3.148}$$

with

$$\boldsymbol{\chi}_{r,k} = \left[{}^{B}\boldsymbol{t}_r^T, \tilde{\boldsymbol{\chi}}_{r,k}\right]^T \tag{3.149}$$

where $\tilde{\boldsymbol{\chi}}_{r,k}$ varies according to the formulations utilized.

The definitions of mutable parameters $\tilde{\boldsymbol{\chi}}_{r,k}$ (nodes) and the corresponding GNSS residuals (edges) can be found at subsection 3.8.1 $\sim$ subsection 3.8.4, except that an initial GNSS extrinsic error that corresponds to subsection 3.7.1 should be added.

The IMU nodes and corresponding edges are the same as subsection 3.9.1.

### 3.9.3 Initialization

The GNSS/INS initializer estimates initial poses, velocities, and biases for LC or TC estimators. Regardless of whether we use LC or TC integrations, we use the LC formulations, i.e. position and velocity, to form the initialization optimization graph for efficiency.

The pitch and roll angle is first computed by the acceleration measure. Then a batch optimization is conducted when a sufficient acceleration is detected. The optimization structure is mostly the same as GINS LC optimization. The difference is that we do not trust the position measurements during initialization because they exhibit large noise if the integer values of GNSS ambiguities are unsolved, which often leads to divergence. Instead, we use the velocity measurement to compute the increment of initial position parameters because we believe the GNSS doppler measurement (which mainly contributes to the velocity estimation) is precise and less affected by errors.

## 3.10 GNSS/INS/Camera Integrated Estimators

### 3.10.1 Solution/Raw/Raw Integration

The SRR estimator utilizes solution from GNSS (position and velocity), raw measurements from INS (acceleration and angular velocity), and raw measurements from visual (features), for estimation. The FGO structure of the GNSS/INS/Camera SRR estimator is shown in Figure 3.9.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := [\boldsymbol{\chi}_{I,k}, \boldsymbol{\chi}_r, \boldsymbol{\chi}_c]^T \tag{3.150}$$

where $\boldsymbol{\chi}_{I,k}$ and $\boldsymbol{\chi}_r$ are the same as subsection 3.9.1. $\boldsymbol{\chi}_{c,k}$ is the visual parameters, which is given as

$$\boldsymbol{\chi}_c := \left[{}^{B}\boldsymbol{t}_c, \boldsymbol{q}_C^B, {}^{W}\boldsymbol{p}_l\right]^T \tag{3.151}$$

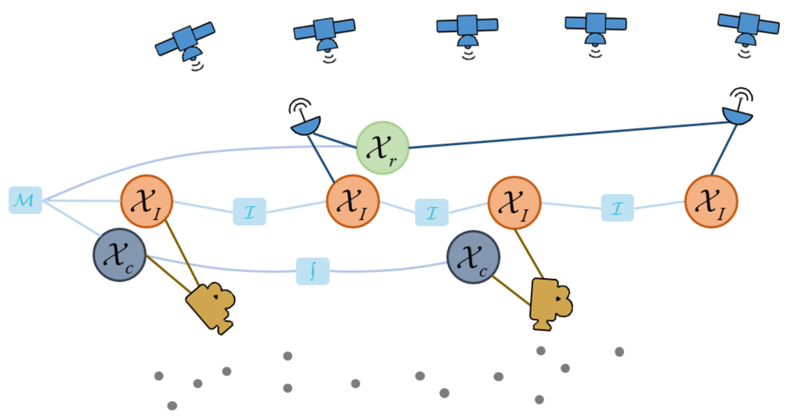


Figure 3.9: FGO structure of GNSS/INS estimator.

where ${}^{B}\boldsymbol{t}_c$ and $\boldsymbol{q}_C^B$ are the camera extrinsics. ${}^{W}\boldsymbol{p_l}$ is a vector of landmark positions ${}^{W}\boldsymbol{p}_l$.

The definitions of the GNSS and IMU edges are the same as subsection 3.9.1.

The camera edge contains:

1) Reprojection error in subsection 3.5.2.

2) Initial camera extrinsic error that corresponds to subsection 3.7.1.

There is no interconnection between camera states because the corresponding parameters are time-invariant. We keep the connection on the graph to indicate that the estimated parameters of epochs vary because of switching tracked landmarks.

### 3.10.2 Raw/Raw/Raw Integration

The RRR estimator utilizes raw measurements from GNSS (pseudorange, carrier phase, and doppler), raw measurements from INS (acceleration and angular velocity), and raw measurements from visual (features), for estimation. The FGO structure of the GNSS/INS/Camera RRR estimator is shown in Figure 3.10.

In total, the estimated parameters are

$$\boldsymbol{\chi}_k := [\boldsymbol{\chi}_{I,k}, \boldsymbol{\chi}_{r,k}, \boldsymbol{\chi}_c]^T \tag{3.152}$$

The definition of $\boldsymbol{\chi}_{I,k}$, $\boldsymbol{\chi}_{r,k}$, and the corresponding edges are the same as subsection 3.9.2. The definition of $\boldsymbol{\chi}_c$ and the corresponding edges is the same as subsec-

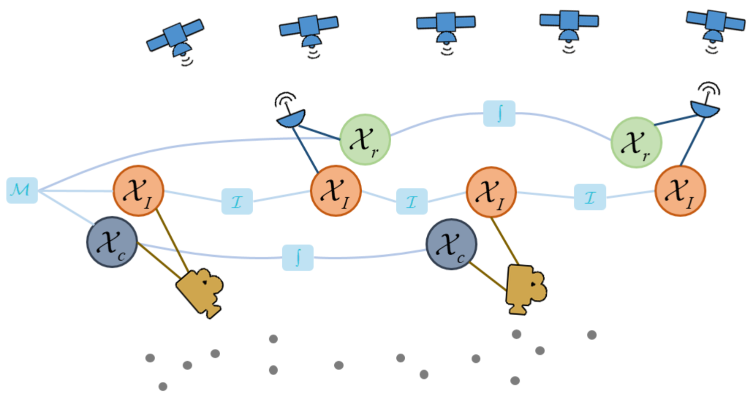


Figure 3.10: FGO structure of GNSS/INS estimator.

tion 3.10.1.

### 3.10.3 Initialization

The GNSS/INS/Camera initializer estimates initial poses, velocities, biases, and landmark positions for SRR or RRR estimators. The initialization is done in two steps: GNSS/INS initialization step and visual initialization step. The GNSS/INS initialization step has been introduced in subsection 3.9.3. After the GNSS/INS initialization, we triangulate the tracked features using the estimated poses.

### 3.10.4 GNSS measurement sparsification

It is widely agreed upon that vision-based estimation algorithms employ keyframes for sparsification to save the computational load. A similar strategy should be developed for high-precision TC GNSS formulations because the dimension of the corresponding parameters is also high.

The principle of selecting keyframes in visual estimation is to try to skip the frames with similar scenes, which contributes less to the estimator and causes a meaningless decrease in the covariance of the estimated parameters. The essence of this principle is that the correlation between unmodeled errors in adjacent frames is high, which makes it incorrect to model the errors as white noise. Sparsifying the frames whitens the error and makes it fit the basic assumption of FGO better.

This property is also evident in GNSS raw measurements. For high-precision GNSS

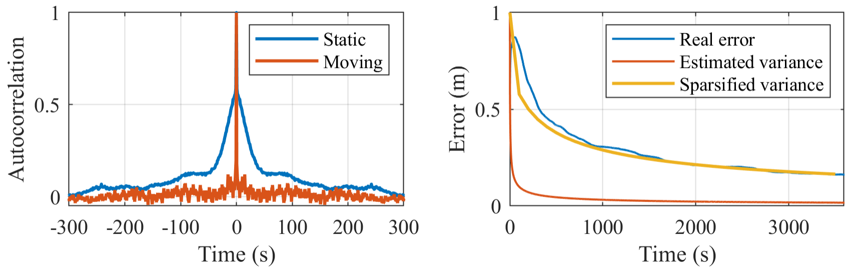


Figure 3.11: Auto-correlation of a satellite in different vehicle motion (left) and the effect of sparsification on covariance estimation (right). The right plot is a simulation of a single parameter estimation problem with a first-order Gauss–Markov error. The correlation time of the error is $\tau$ and the sparsification time step is $2\tau$.

formulations, the auto-correlation property of the unmodeled errors is mainly caused by the multipath, and the correlation time is dependent on the change rate of the scene. The GNSS measurement auto-correlation and the effect of sparsification are shown in Figure 3.11. We can see that the correlation time changes according to the motion, and the effect of correlation on the estimator can be mitigated by sparsification.

Due to the correlation time of GNSS measurement changing over time, a consistent time step for sparsification is difficult to determine. Luckily, scene change rate behaves similarly to the criteria for keyframe selection. Therefore, our GNSS sparsification strategy is designed as keeping measurements near keyframes and disregarding the rest.

# Appendices

# Appendix A

# Notations and Definitions

## A.1 Frames

Denote ${}^{A}\mathcal{F}$ the reference frame $A$, we define some commonly used frames:

${}^{W}\mathcal{F}$ The East-North-Up (ENU) world frame. The origin is the first estimated GNSS position or the value of configuration "initial_global_position".
${}^{W_G}\mathcal{F}$ The Earth-Centered Earth-Fixed (ECEF) world frame. Since the Latitude-Longitude-Height (LLA) world frame can be converted to the ECEF frame with a fixed model, we do not distinguish these two frames in this manual.
${}^{B}\mathcal{F}$ Body frame defined in Right-Forward-Up sequence.
${}^{r}\mathcal{F}$ GNSS receiver frame, rotateless.
${}^{I}\mathcal{F}$ IMU sensor frame.
${}^{C}\mathcal{F}$ Camera sensor frame.

## A.2 Transformations

The followings are some definitions of commonly used transformation:

${}^{P}\boldsymbol{T}_{Q}$ The transformation matrix that transforms a homogeneous point from ${}^{Q}\mathcal{F}$ to ${}^{P}\mathcal{F}$.
$\boldsymbol{R}_{Q}^{P}$ The rotation matrix that rotates a point from ${}^{Q}\mathcal{F}$ to ${}^{P}\mathcal{F}$.
$\boldsymbol{q}_{Q}^{P}$ The quaternion that describes the rotation $\boldsymbol{R}_{Q}^{P}$.

${}^{Q}\boldsymbol{t}_P$ The translation of the origin point of ${}^{P}\mathcal{F}$ in ${}^{Q}\mathcal{F}$ frame. If the translation item is defined at the origin point of the body frame, i.e. ${}^{Q}\boldsymbol{t}_B$, we write this translation as ${}^{Q}\boldsymbol{t}$ for simplicity.

## A.3 Rotation Definitions

Define the quaternion

$$\boldsymbol{q} := \begin{bmatrix} \boldsymbol{q}_v^T & q_w \end{bmatrix}^T = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^T \tag{A.1}$$

The multiplication matrix is defined as

$$\boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = \boldsymbol{Q}_1^{+}\boldsymbol{q}_2 = \boldsymbol{Q}_2^{-}\boldsymbol{q}_1 \tag{A.2}$$

with

$$\boldsymbol{Q}^{+} := q_w\boldsymbol{I} + \begin{bmatrix} \lfloor \boldsymbol{q}_v \times \rfloor & \boldsymbol{q}_v \\ -\boldsymbol{q}_v^T & 0 \end{bmatrix}, \quad \boldsymbol{Q}^{-} := -q_w\boldsymbol{I} + \begin{bmatrix} \lfloor \boldsymbol{q}_v \times \rfloor & \boldsymbol{q}_v \\ -\boldsymbol{q}_v^T & 0 \end{bmatrix} \tag{A.3}$$

and

$$\lfloor \boldsymbol{q}_v \times \rfloor := \begin{bmatrix} 0 & -q_z & q_y \\ q_z & 0 & -q_x \\ -q_y & q_x & 0 \end{bmatrix} \tag{A.4}$$

We estimate the quaternion of the body in the world frame $q_B^W$, and the disturbance is defined on the left side, i.e.

$$\boldsymbol{R}_B^W \boxplus \delta\boldsymbol{R} := \exp\left(\lfloor \boldsymbol{\delta\theta} \times \rfloor\right) \boldsymbol{R}_B^W \tag{A.5}$$

and

$$\boldsymbol{q}_B^W \boxplus \delta\boldsymbol{q} := \delta\boldsymbol{q} \otimes \boldsymbol{q}_B^W \tag{A.6}$$

with

$$\delta\boldsymbol{q} := \exp\left(\begin{bmatrix} \frac{1}{2}\boldsymbol{\delta\theta} \\ 0 \end{bmatrix}\right) \tag{A.7}$$

The quaternion exponential is computed as

$$\exp\left(\begin{bmatrix} \frac{1}{2}\boldsymbol{\delta\theta} \\ 0 \end{bmatrix}\right) = \begin{bmatrix} \operatorname{sinc}\left\|\frac{\boldsymbol{\delta\theta}}{2}\right\| \frac{\boldsymbol{\delta\theta}}{2} \\ \cos\left\|\frac{\boldsymbol{\delta\theta}}{2}\right\|^2 \end{bmatrix} \tag{A.8}$$

Correspondingly, the quaternion logarithmic is defined as

$$\log(\delta\boldsymbol{q}) := \begin{bmatrix} \frac{1}{2}\boldsymbol{\delta\theta} \\ 0 \end{bmatrix} \tag{A.9}$$

The rotation angle exponential is computed as

$$\exp(\lfloor\boldsymbol{\theta}\times\rfloor) = \boldsymbol{I} + \frac{\sin(\|\boldsymbol{\theta}\|)}{\|\boldsymbol{\theta}\|}\lfloor\boldsymbol{\theta}\times\rfloor + \frac{1-\cos(\|\boldsymbol{\theta}\|)}{\|\boldsymbol{\theta}\|^2}\lfloor\boldsymbol{\theta}\times\rfloor^2 = \boldsymbol{R} \tag{A.10}$$

For convenience, we define the Exp operation to directly convert the rotation vector to the rotation matrix

$$\operatorname{Exp}(\boldsymbol{\theta}) := \boldsymbol{R} \tag{A.11}$$

Correspondingly,

$$\operatorname{Log}(\boldsymbol{R}) := \boldsymbol{\theta} \tag{A.12}$$

We use the quaternion $\boldsymbol{q}$ for parameterization and $\boldsymbol{\theta}$ for error computation. So there will be an $\boldsymbol{q}$ update every time when a new $\boldsymbol{\theta}$ is computed. The update is given by Equation A.6. The corresponding Jacobian matrix is

$$\boldsymbol{J}_q^{\theta} := 2\boldsymbol{Q}^{-}\left(\boldsymbol{q}^{-1}\right)_{3\times 4} \tag{A.13}$$

where the subscript $3 \times 4$ represents the corresponding matrix block at the top left.

# Appendix B

# GNSS Linear Combinations

To simplify, we use the notation $G$ to represent the pseudorange $P$ and phase-range $L$ in this chapter.

## B.1 Combination

1. Ionosphere-Free (IF) combination.

   It uses the measurements from two frequencies to form a (first-order-) ionosphere-delay-independent measurement. It can eliminate most of the ionosphere effect.

$$G_{IF} := \frac{f_i^2}{f_i^2 - f_j^2} G_i - \frac{f_j^2}{f_i^2 - f_j^2} G_j \tag{B.1}$$

2. Geometric-Free (GF) combination.

   It cancels the geometric part of the measurement, leaving all the frequency-dependent effects besides multipath and random noise.

$$G_{GF} := G_i - G_j \tag{B.2}$$

3. Wide-lane (WL) combination.

   It is used to create a signal with a significantly wide wavelength.

$$G_{WL} := \frac{f_i}{f_i - f_j} G_i - \frac{f_j}{f_i - f_j} G_j \tag{B.3}$$

4. Narrow-lane (NL) combination.

   It is used to create a signal with a narrow wavelength.

$$G_{NL} := \frac{f_i}{f_i + f_j} G_i + \frac{f_j}{f_i + f_j} G_j \tag{B.4}$$

## B.2 Differential

1. Single-Difference (SD)

   It make difference between the measurements from two GNSS receivers to eliminate most of the satellite and propagation segment errors.

$$G^s_{rr_b,i} := G^s_{r,i} - G^s_{r_b,i} \tag{B.5}$$

2. Between-Satellite-Difference (BSD)

   It make difference between the measurements from two GNSS receivers to eliminate most of the receiver and propagation segment errors.

$$G^{ss_b}_{r,i} := G^s_{r,i} - G^{s_b}_{r,i} \tag{B.6}$$

3. Double-Difference (DD)

   It combinates the SD and BSD to eliminate most of the errors.

$$G^{ss_b}_{rr_b,i} := G^s_{r,i} - G^s_{r_b,i} - (G^{s_b}_{r,i} - G^{s_b}_{r_b,i}) \tag{B.7}$$

# References

[1] T. Takasu. *RTKLIB*. Version 2.4.3. Dec. 2020. URL: https://www.rtklib.com/rtklib.htm.

[2] S. Leutenegger et al. “Keyframe-based visual–inertial odometry using nonlinear optimization”. In: *International Journal of Robotics Research* 34.3 (2014), pp. 314–334.

[3] Christian Forster et al. “SVO: Semidirect visual odometry for monocular and multicamera systems”. In: *IEEE Transactions on Robotics* 33.2 (2016), pp. 249–265.

[4] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. *Ceres Solver*. Version 2.1. Mar. 2022. URL: https://github.com/ceres-solver/ceres-solver.

[5] John A Klobuchar. “Ionospheric time-delay algorithm for single-frequency GPS users”. In: *IEEE Transactions on aerospace and electronic systems* 3 (1987), pp. 325–331.

[6] Michael Bevis et al. “GPS meteorology: Mapping zenith wet delays onto precipitable water”. In: *Journal of Applied Meteorology (1988-2005)* (1994), pp. 379–386.

[7] Johannes Böhm et al. “Global Mapping Function (GMF): A new empirical mapping function based on numerical weather model data”. In: *Geophysical research letters* 33.7 (2006).

[8] Deepak Geetha Viswanathan. “Features from accelerated segment test (fast)”. In: *Proceedings of the 10th workshop on image analysis for multimedia interactive services, London, UK*. 2009, pp. 6–8.

[9] Bruce D Lucas and Takeo Kanade. “An iterative image registration technique with an application to stereo vision”. In: *IJCAI’81: 7th international joint conference on Artificial intelligence*. Vol. 2. 1981, pp. 674–679.

[10] Christian Forster et al. “On-manifold preintegration for real-time visual–inertial odometry”. In: *IEEE Transactions on Robotics* 33.1 (2016), pp. 1–21.

[11] X -W Chang, X Yang, and T Zhou. “MLAMBDA: A modified LAMBDA method for integer least-squares estimation”. In: *Journal of Geodesy* 79 (2005), pp. 552–565.