Robot Operating System - A Modular and Flexible Framework for Geodetic Multi-Sensor-Systems

David REJCHRT, Tomas THALMANN, Andreas ETTLINGER, Hans-Berndt NEUNER

Research Group Engineering Geodesy – TU Wien, email@email.ac.at

Zusammenfassung

Die Messdatenerfassung ist eine elementare Aufgabe in ingenieurgeodätischen Projekten. In vielen Anwendungen sind mehrere Sensoren erforderlich, um die Messaufgabe zu erfüllen. Solche Multi-Sensor-Systeme kommen vor allem in kinemtischen robotischen Anwendungen zum Einsatz um 3D-Punktwolken zu generieren. Vor allem bei Systemen wie Drohnen (Unmanned Aerial Vehicles, UAVs) oder bodengestützten Robotern (Unmanned Ground Vehicles, UGVs) erfreut sich das Robot Operating System (ROS) großer Beliebtheit. Dieses Open-Source Projekt ist ein Software-Framework, das die Datenakquise, Auswertung, Verspeicherung und Visualisierung für nahezu alle Robotik-Anwendungen ermöglicht. Der Forschungsbereich Ingenieurgeodäsie an der TU Wien nutzt und entwickelt auf dieser Basis Komponenten sowohl für die Nutzung solcher kinematischer Multi-Sensor-System, als auch für permanente statische Sensornetze. Dieser Beitrag gibt eine Einführung in das Konzept von ROS und zeigt mehrere Fallstudien, in denen dieses System bereits in Kombination mit Einplatinenrechner wie z.B. Raspberry Pi 3 eingesetzt wurde.

1 Geodetic Data Acquisition

A key competence of engineering geodesy is planning, organization and execution of measurement campaigns, generally speaking the data acquisition process in a specific application. This provides the foundation for further in-depth analysis and evaluation to generate the required information. This information is necessary to conclude the right decisions depending on the specific goal or task of the application. In most applications, multiple sensors are necessary due to the growing level of complexity. From an application point of view, a single sensor is at the retreat. According to SCHWIEGER & STERNBERG (2014), multiple sensors can be used spatially distributed, redundant and complementary.

Due to the technological development over the last years, sensors and affiliated hardware has become cheaper, smaller and less power consuming. A development, also driven by hot topics like Industry 4.0 and Internet of Things (IoT). This also leads to an increasing number of sensors involved in data acquisition tasks in engineering geodesy. Furthermore, permanently mounted sensor networks required for real-time or near-real-time applications become more feasible and economical.

In terms of data acquisition, we can distinguish between two cases: *kinematic*, where all or some of the sensors are in motion, or *static*, where all sensors remain stable during the measurement process. Nevertheless, the object or region of interest might be static or kinematic too FOPPE et al. (2004). From the application point of view, we can categorize two groups of applications:

- Permanent (long or medium term) static **Sensor Networks** (SNs), used for any kind of monitoring task. The first thing that comes to mind is classical deformation monitoring. Very often such sensor networks also serve as supplementary infrastructure, e.g. for indoor navigation or GNSS correction services. Key requirement for this group of applications are low latencies in data output, high reliability and high availability.
- Short period sensor networks for kinematic applications like **Mobile Mapping Systems** (MMSs) or other robotic applications span the second group of data acquisition tasks. The key problem here is the determination of the platform pose (consisting of position and attitude, 6DOF). These platforms might be Unmanned Ground Vehicles (UGVs), cars, trains or Unmanned Aerial Vehicles (UAVs). Some sensors used, might be the same as in SN applications (e.g. GNSS or total stations) and some might be special sensors (e.g. odometers). Usability, real-time capability and efficiency play a role in these applications.

For both categories, a variety of software solutions is available on the market. For geodetic monitoring, there are proprietary packages (Leica GeoMoS, Trimble 4D Control Software, Geodata Eupalinos and many more) as well as Open Source packages (e.g. DABAMOS, see ENGEL & SCHWEIMLER (2015)). Whereas, in the mobile mapping sector most proprietary hardware solutions come along with their own software which prevents reusability. Regarding the Open Source community, the Robot Operating System (ROS) has gained a lot of attention over the last decade. With a permanently growing community (about 150.000 unique visitors on wiki.ros.org, growing by 30% annually) of researchers, developers and application engineers it has a significant influence on the robotics market. Geodetic researchers, e.g. LINZER et al. (2018) have already shown that ROS is very suitable for geodetic MMSs.

In this paper, we want to show, that ROS (with a few extensions and improvements) is also a very powerful tool for several classical geodetic sensor networks. Requirements and components of a modern geodetic monitoring system characterized by high flexibility are (after STEMPFHUBER (2012)):

a) Configuration and Administration (ideally also during operations without any downtime)

b) Automatic controlling and reading of several sensors

c) Communications and remote control

d) Data evaluation and visualization

e) Data administration (permanent, durable and redundant storage of the data)

f) Alarming (via Signal, Email or SMS, etc.)

g) Data export and automatic report creation

Since points a) to d) are highly overlapping with MMS and more generally with robotics, these are already covered by the functionality of ROS. Points e) to g) are more specific to monitoring networks, where further development effort is necessary. In this contribution, we are going to show the progress at our research group concerning such developments. According to the Open-Source idea we are planning to publish the developed packages at our

website. In our opinion - especially in education and research - it would be a great benefit to use one single framework for both groups of data acquisition tasks. It could reduce time and cost expanses, because of shorter training periods and less license costs. The future goal of our work is to establish a reliable and adaptable Multi-Sensor-System, to solve numerous measurement tasks in engineering geodesy.

2 Robot Operating System

Robot Operating System or ROS is an open source framework for programming robotic networks. It provides an elegant, easy and robust solution for connecting many separate computing units together in a neatly structured network, thus enabling all machines on the network to communicate with each other. This framework, as the name suggests, comes from the field of robotics, but because of its flexibility, it is possible to find applications in many other fields, as this paper shows.

One of the key concepts of ROS is its modularity as the whole system consists of many different packages, which makes reusability of code simple. There are already many packages available to use. In 2018, 11.770 unique packages are available in the free to use repositories (ROS METRICS 2018). In addition, many manufacturers are aware of the impact of ROS, thus already providing ROS packages (drivers) for their hardware. A selection of such sensors used at our research group is shown in Figure 1. These packages are able to interact smoothly with each other because ROS defines the underlying interfaces and protocols.

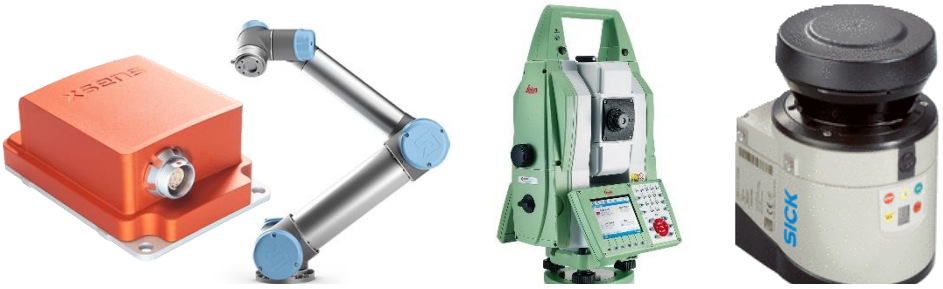

Fig. 1: Some devices compatible with ROS, from left to right: Xsens IMU, UR 5 industrial robot arm, Leica Multistation MS50, Sick LMS 111 Laser Scanner

2.1 Nodes and Topics

A ROS package consists of one or more ROS nodes, which are essential pieces of code that solve a small part of the whole task. For example, a driver node encapsulates some piece of hardware. On the other hand, an algorithm node implements a methodical processing step. For example, a total station driver node handles the measurement process and streams spherical coordinates to a subsequent algorithm node, which converts these to cartesian coordinates. Subsequently, another node transforms these coordinates to the project coordinate system. An example code of such a node in the appendix of this article indicates the simplicity of the development process. A productive data stream is created by chaining multiple nodes,

which allows solving complex tasks using divide and conquer design paradigm. A graphical representation (called ROS graph) of all ROS nodes is shown in Figure 2. An ellipse represents a node and topic is shown as a rectangle.

Data and information between those nodes are exchanged through so-called ROS topics. A ROS topic is a subject oriented abstract space or data stream. A topic is strongly typed, which means that meta-information like structure and type of the data is strictly defined in a ROS message. A ROS message is a packet of structured information through which data is exchanged in a ROS network. The structure of this message is fully customizable. ROS provides several pre-defined message types.

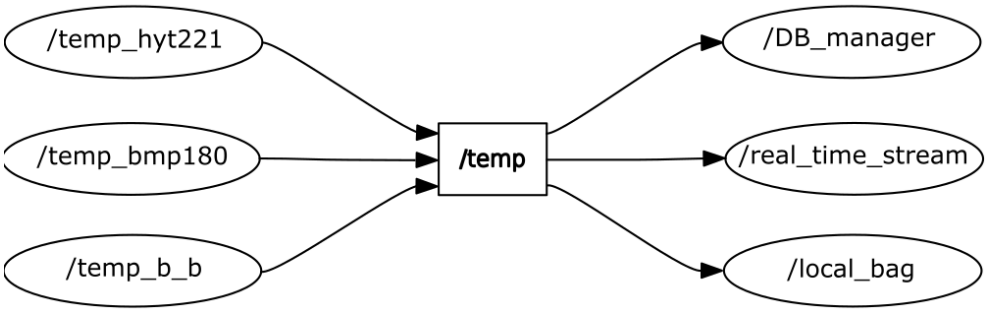

Fig. 2: Example of simple RQT graph

Whereas a message defines the kind of data, a ROS topic is a data stream through which nodes share data and information. It is possible for two or more nodes to publish to the same topic as well as an arbitrary number of nodes is able to subscribe to a topic and receive all published messages. Once a message arrives at a subscriber node, the message gets decomposed and the data is further processed according to the purpose of this specific node. ROS is completely responsible for the underlying communication, so that the developer can focus on the actual problem.

Every topic is visible in the whole ROS network. This is achieved by a central part of ROS, called the ROS Master that serves as 'broker' who keeps track about topics, subscriptions and publishers (ROS WIKI 2018).

2.2 Parameters and Launch files

A ROS node should be developed as a reusable piece of code for a recurring task or problem. No matter if it abstracts a hardware device or encapsulates a specific processing step, it is good practice to define some configuration switches or settings. For instance, one might design the measurement rate, thresholds or processing parameters like constants or filter parameters. With such an approach, it is possible to enable the program to an even broader range of application and improve its reusability.

These settings are generally called *parameters* in the ROS terminology. These parameters are stored and managed centrally, so that every node can access its own private parameters as well as global or local parameters. At execution time, the parameters are parsed by a parameter server. A parameter server is a shared, multi-variate dictionary. Nodes use this server to store and retrieve parameters at runtime. It is meant to be globally viewable so that tools can easily inspect the configuration state of the system and modify it if necessary. This makes

the key requirement for geodetic SNs of online administration and configuration very easy, since changes at this central parameter server are automatically propagated to the affected parts of the SN.

As mentioned above, one of the key concepts of ROS is its modularity. As a consequence, a number of nodes and topics will be necessary to tackle a specific task or to solve a specific problem. For different applications one might need different sets of nodes that interact in specific ways with each other. To represent/reproduce these variable applications (or use cases) ROS introduces a helpful concept called a *launch* file, which is an XML file where it is specified, which nodes should run, with which parameters and on which machine. Before launching individual nodes, it makes sure that all prerequisites (like master and parameter server are running) have been met.

3 IGROS Web application

IGROS Web application (**I**ngenieur**g**eodäsie **R**obot **O**perating **S**ystem) serves as a graphical user interface (GUI) to the sensor network represented within the ROS framework. Although ROS is easy to use (ASCII/XML knowledge suffice), the aim of the GUI is to make the usage of ROS even easier. The main functions of the interface are based on the requirements of a geodetic monitoring system:

- Managing jobs and data
- Configuring the sensor network
- Starting and stopping data acquisition sessions
- Displaying and exploring data
- Exporting data

A major component in this whole system is a relational database, which serves as central data storage. This is an integral part of a modern geodetic monitoring system, which is missing in ROS out of the box. Compared to a file storage, a database simplifies backup, maintenance, authorization and sharing of data.

3.1 Database

The database fulfils two challenges within the IGROS framework: it abstracts and represents the actual sensors and hardware in the network and all acquired data is centrally stored in a structured and accessible way.

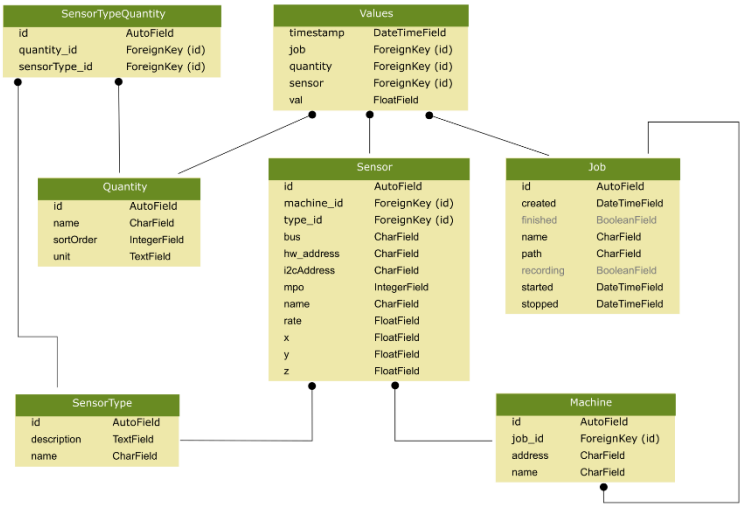

Fig. 3:Database Schema

Figure 3 shows the database scheme. Measured values are stored in the *values* table. Every measured value is associated with a *job*. Primary function of a job is to make related data easily accessible and to keep track of common metadata. For each job, one can deploy different *machines* (data logger or computing unit) with different sensors attached. Deployed components can have different settings and/or purpose in the *job*, so it is necessary to store their various configurations as well. The amount of such metadata about the SN can grow fast with the size of the SN. Therefore, the web application provides a function to clone a job, which ensures reusability of different setups.

A *sensor* has a specific *sensor type*. A sensor type is an abstraction model of a physical sensor. The *sensor type* is meant to store general information of a sensor, e.g. the configuration parameters that can be adjusted or the *quantities* a sensor observes. An example for a sensor type might be an IMU (Inertial Measurement Unit), which observes 3 acceleration and 3 angular rates (=*quantity*). The actual sensors deployed in the system (could be more than one IMU) are represented by entries in the *sensor* table.

Physical quantities that can be measured are kept in the *quantity* table, with some details, like unit and sort order, which is essential when displaying or exporting data.

3.2 User interface

The user interface is designed as a web-application, because of the platform independence and high accessibility. The user interface of a SN should be ideally accessible anytime and anywhere and with a web interface this is the case nowadays even from smartphones. We

have decided to deploy a *bootstrap* template because it ensures visually appealing and responsive design. Figure 4 shows a visual concept of two pages of the admin interface.

One key functionality is the overview and administration of all the jobs in the system, as well as the administration of supported sensor types and quantities. The job dashboard (Figure 4) shows elementary details about the job (e.g. name, date created, …) in the upper left box and a summary of configured/deployed computing units and sensors (physical SN components). The dashboard serves as a control centre for the data acquisition job and allows several operations like start/stop, configure, delete and clone. There is also a graphical representation of ROS graph visualization of all contributing ROS nodes and topics, which refer to the logical SN components.

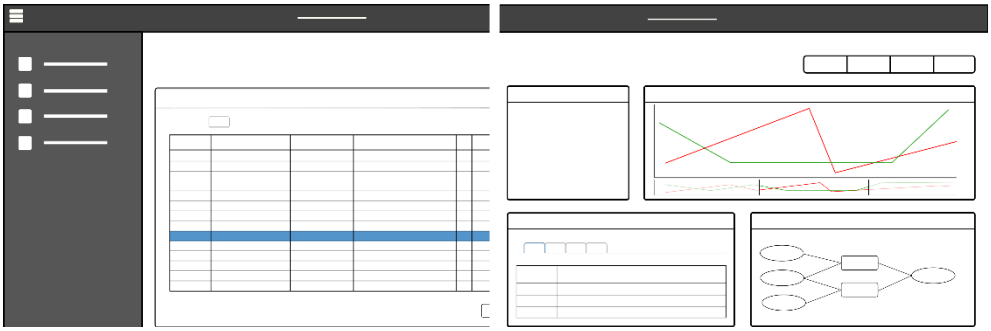

Fig. 4: (left) Home page template. (right) job dashboard page

The predominant part of the dashboard is reserved for data visualization. A number of options like time windowing and sub selection of sensors and quantities is available. A data cursor can be used to inspect the exact numerical value at every timestamp. There is also an option to switch the plot into full screen mode to allow even better inspection of the data.

In addition to visualization, there is also a module for exporting data. On the export page, the user is able to customize the details to be exported: the time window, which sensors, which quantities and the resulting file format. For example: We have a *MATLAB* program, which calculates distance correction due to refraction along a laser beam path. Therefore, we are interested only in data from sensors which measure temperature and are located near the beam path. Since we do not need the whole series of observations, we can select observations made "shortly" prior to the distance measurement and "shortly" afterwards. Since the output is going to be used in *MATLAB*, we can export the data to a *.mat* format. Other formats available are csv (various delimiters), pandas, xls/xlsx, txt.

4 Temperature Sensor Network

With increasing requirement on accurate laser distance measurement, more detailed description of the local atmosphere, through which a laser beam propagates, is desired. The atmosphere has a dispersive effect on the beam, causing its wavelength to vary, which introduces a systematic effect on the distance measurement. Three parameters have the major impact on the propagation of electro-magnetic waves through atmosphere: temperature, pressure and air humidity. Among them, temperature has by far the strongest influence. Measuring these quantities simultaneously with sufficiently high measurement rate on several query points along the beam's trajectory, one can very accurately approximate the state of the atmosphere at propagation time and thus estimate the resulting correction term for the measurement.

Fig. 6:
Measurement laboratory at the research group engineering geodesy at TU Wien

The research group engineering geodesy is planning to equip the measurement lab (see Figure 6) with a permanent sensor network for capturing meteorological parameters. Using the developed system based on IGROS and the database on a central server we are able to query and interpolate a laboratory covering temperature field to correct measurements of EDM (Electronic Distance Measurement), Laser Scanner, Laser Trackers or Interferometer. Pilot studies have been carried out concerning calibration of different temperature sensors within the IGROS framework. Figure 7 shows temperature series of two sensors in a cup of hot water cooling down.

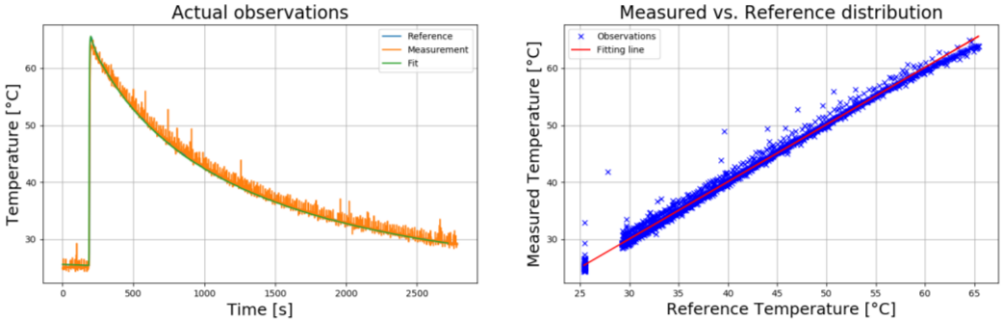

Fig. 7: Temperature sensor calibration

5 Indoor-Pedestrian Navigation with Smartphones

Indoor positioning with smartphone sensors is a challenging task, as these built-in sensors are low-cost and subject to multiple error sources leading to systematic effects in the estimated position (KELLER (2012), ETTLINGER (2017), BURGESS (2018)). To aid the positioning process on the smartphone, additional sources of information delivered by the building itself are of interest. A possibility is, to use the sensor infrastructure provided by the building, which can consist of many different sensors (e.g. motion detectors, door sensors, temperature sensors, etc...). In the measurement lab of the research group engineering geodesy of TU Wien, such a sensor infrastructure is installed. It consists of six Raspberry Pi's model 3 which are distributed in the lab, each controlling two motion-detectors and two ultrasonic distance sensors (Figure 8). The sensor events are collected by running custom ROS-nodes on the Raspberry Pi's and are sent to the ROS Master via a Wi-Fi network connection.

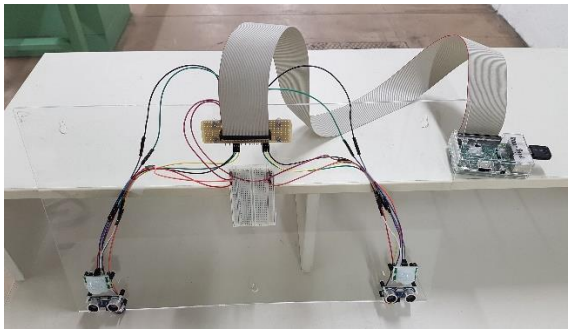

Fig. 8:
Mounting and control of motion detectors and ultrasonic distance sensors

To fuse the information from the phone and from the sensor infrastructure, all the sensors have to be available in a common spatial and temporal frame of reference. Especially the timing reference is a challenging task to realize on smartphones (see ETTLINGER (2017)), as it has to be set manually by using special apps (e.g. ClockSync[1]). A major advantage of ROS

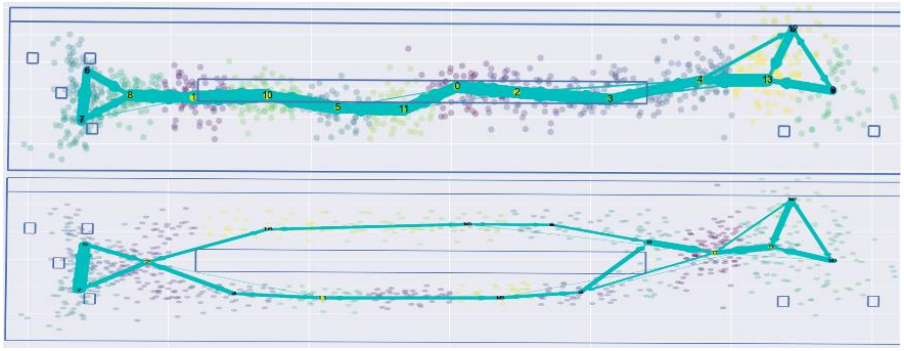

Fig. 9: (top) Raw smartphone positions; (bottom) refined positions with sensor infrastructure

[1] https://play.google.com/store/apps/details?id=ru.org.amip.ClockSync&hl=de (22.11.2018)

is the possibility to run the nodes on Android. The sensor data is sent directly to the ROS Master via Wi-Fi. There it receives a synchronized timestamp, leading to a more convenient realization of timing reference and data storage. Figure 9 shows the result of an experiment in the measurement lab, where the raw smartphone positions are refined by using the sensor events of the distributed motion detectors (BURGESS (2018)).

6 Kinematic Systems using RTS

Whereas the preceding two case studies refer to the first category of stationary SNs, we have also used our system in several kinematic mobile mapping projects. We have used robotic total stations (RTS) to track railway locomotives as a reference to evaluate an onboard GNSS-based positioning system as well as to directly georeference UAV (Unmanned Aerial Vehicle)-carried laser scanning observations. Our research group also conducts research in the field of an autonomous ground-based MMS (UGV, Unmanned Ground Vehicle). Compare Figure 10.

All these applications have been implemented using Raspberry Pi's model 3 for data acquisition. As in the case study for indoor positioning time referencing and synchronization between spatially distributed sensors is a challenging task. For RTS we have developed a synchronization routine using GNSS and PPS (pulse per second) as well as the Network Time Protocol (NTP) (THALMANN & NEUNER 2018). These routines can be used by simply adding the corresponding nodes.

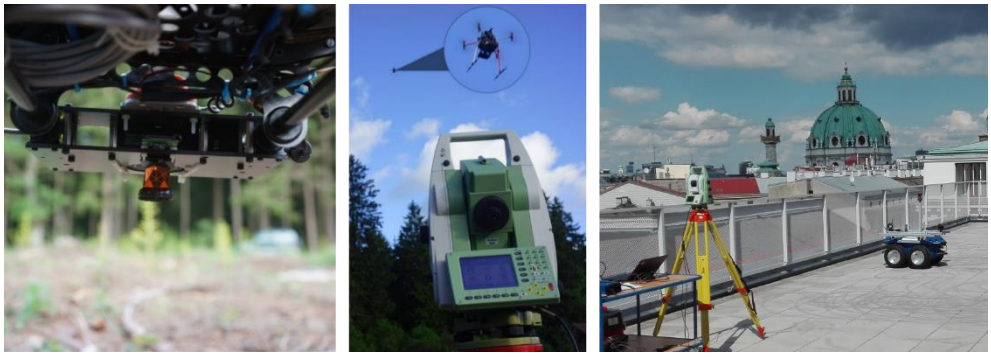

Fig. 10: (left) Prism mounted on UAV; (center) RTS tracking UAV; (right) RTS tracking UGV

Using a synchronized RTS in addition to an IMU and a single frequency GNSS receiver on the UAV enabled us to compute several solutions for the trajectory and further referencing laser scans. A GNSS-IMU Filter is compared to an RTS-IMU Filter and to poses from photogrammetry using ground control points. The results can be seen in Fig. 11.

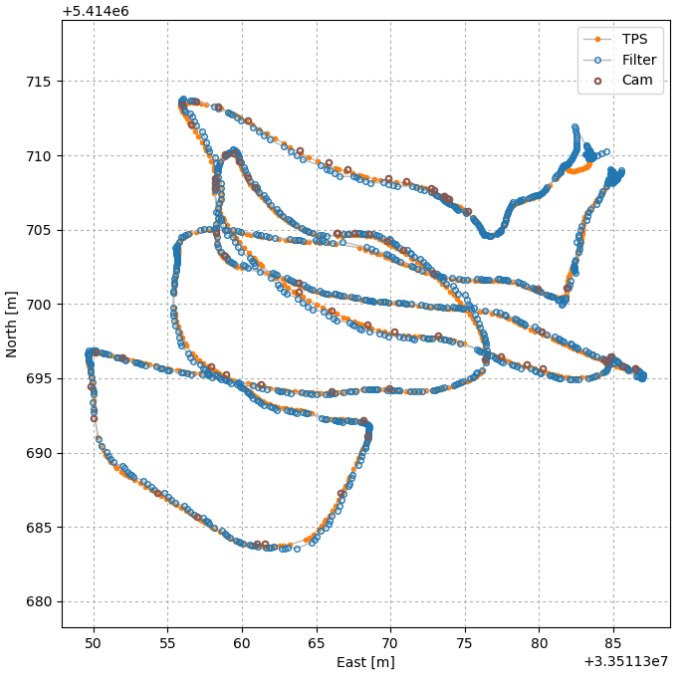

Fig. 11: GNSS-IMU Filter results in blue, RTS-IMU (TPS) solution in orange and poses from bundle-block adjustment in brown

7 Summary

Data acquisition tasks in engineering geodesy can be summarized by two schemas/concepts: kinematic MSSs for mobile mapping and stationary SNs for static as well as kinematic applications. This paper gives an overview about a modular and flexible software framework called ROS. We have highlighted the key concepts of ROS and showed how it can be used for both categories of data acquisition tasks. Furthermore, we have shown the ongoing developments at the research group engineering geodesy at TU Wien concerning SNs. This included a responsive graphical web interface as well as a reusable database schema for observation data.

Three case studies were presented and serve as a proof of concept of our framework.

References

BURGESS, T., METZLER, B., ETTLINGER, A., NEUNER, H. (2018). Geometric Constraint Model and Mobility Graphs for Building Utilization Intelligence. In 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN). IEEE

ETTLINGER, A., NEUNER, H., BURGESS, T. (2017). Orientierungsberechnung mit Smartphone-Sensoren. In AVN - Allgemeine Vermessungsnachrichten, 2018(4), pp. 91-104

ENGEL, P., & SCHWEIMLER, B. (2015). Development of an Open-Source Automatic Deformation Monitoring System for Geodetical and Geotechnical Measurements. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 40.

FOPPE, K., SCHWIEGER, V., & STAIGER, R. (2004). Grundlagen kinematischer Mess- und Auswertetechniken. Schriftenreihe des DVW, Band 45/2004: Kinematische Messmethoden – Vermessung in Bewegung.

KELLER, F., WILLEMSEN, T., STERNBERG, H. (2012). Calibration of smartphones for the use in indoor navigation. In 2012 International Conference on In-door Positioning and Indoor Navigation (IPIN). IEEE

LINZER, F., BARNEFSKE, E., & STERNBERG, H. (2018). Konzeption eines modularen MMS innerhalb der Robot Operating System (ROS) - Umgebung im geodätischen Zusammenhang. In Photogrammetrie, Laserscanning & Optische 3D-Messtechnik - Beiträge der Oldenburger 3D-Tage 2018. Wichmann - VDE Verlag.

ROS METRICS (2018) Community Metrics Report July 2018. http://download.ros.org/downloads/metrics/metrics-report-2018-07.pdf (22.11.2018)

ROS WIKI (2018). Robot operating system. Open Source Robotics Foundation. http://wiki.ros.org/Services (21.11.2018)

SCHWIEGER, V., & STERNBERG, H. (2014). Multi-Sensor-Systeme in der Ingenieurgeodäsie – Grundlagen und Überblick. Schriftenreihe des DVW, Band 75/2014: Multi-Sensor-System – Bewegte Zukunftsfelder.

STEMPFHUBER, W. (2012). Leistungsfähigkeit von geodätischen Monitoringsystemen. Bautechnik, 89(11), 794–800.

THALMANN, T., & NEUNER, H. (2018). Untersuchung des Network Time Protocols für die Synchronisation von Multi-Sensor-Systemen. AVN - Allgemeine Vermessungsnachrichten, 125(6), 163–174.

Appendix

```
1  # file: sphere2cartNode.py
2  #!/usr/bin/env python
3  import rospy
4  import math
5
6  from igros_tps.msg import tpsObsMsg, tpsPointMsg
7
8
9  def sphere2cart(msg):
10     # convert measured spherical coordinates to cartesian coordinates
11     x = msg.r * math.cos(msg.hz) * math.sin(msg.zd)  # [m]
12     y = msg.r * math.sin(msg.hz) * math.sin(msg.zd)  # [m]
13     z = msg.r * math.cos(msg.zd)  # [m]
14
15     # publish a pointMsg (x, y, z)
16     pt_msg = pointMsg(x=x, y=y, z=z)
17     pub.publish(pt_msg)
18
19
20 def listener():
21     # name of the node
22     rospy.init_node('sphere2cart', anonymous=True)
23
24     # topic to get the data from
25     rospy.Subscriber('/tps1/tpsObs', tpsObsMsg, sphere2cart)
26
27     # keep working until process is terminated
28     rospy.spin()
29
30
31 if __name__ == '__main__':
32     # topic to publish the converted data to
33     pub = rospy.Publisher('/tps1/cartCoords', tpsPointMsg, queue_size=10)
34     listener()
35
36
37 # file: tpsObsMsg.msg
38 float64 hz
39 float64 zd
40 float64 r
41
42
43 # file: tpsPointMsg.msg
44 float64 x
45 float64 y
46 float64 z
47
```

Line 25 defines the incoming topic, e.g. the data stream from the total station driver node.

Line 33 defines the outgoing topic.

Lines 37 – 47 are located in separate files and define the two message types describing the data types of the two topics.