Recent Enhancements of the Multi-Sensor Navigation Analysis Tool (MuSNAT)

Markel Arizabaleta, Hepzibah Ernest, Jürgen Dampf, Thomas Kraus, Daniela Sanchez-Morales,
Dominik Dötterböck, Andreas Schütz, Thomas Pany
Universität der Bundeswehr München - Institute of Space Technology and Applications

BIOGRAPHY

Markel Arizabaleta studied his Master's degree in Telecommunication Engineering at Universidad del País Vasco (UPV-EHU) in Bilbao, Spain. He is currently a research associate at the Universität der Bundeswehr München at the Institute of Space Technology and Space Applications (ISTA). His research topics are GNSS receiver signal processing and authentication, and positioning with LTE and 5G signals.

Hepzibah Ernest earned her Masters degree in Communication Engineering from Karunya University in India prior to a Masters in Geodesy and Geoinformation sciences from the Technical University of Berlin. She has since been a research associate at the satellite navigation section of the Institute of Space Technology and Space Applications at the Universität der Bundeswehr München. Her research interest focuses on the signal processing aspects of GNSS receiver primarily dealing with RF interference detection and characterization.

Thomas Kraus is a research associate at the Universität der Bundeswehr München and works for the satellite navigation unit LRT 9.2 of the Institute of Space Technology and Space Applications (ISTA). His research focus is on future receiver design offering a superior detection and mitigation capability of RF interferences. He holds a master's degree in Electrical Engineering from the Technical University of Darmstadt, Germany.

Daniela Sanchez-Morales studied Telematics Engineering at Instituto Tecnológico Autónomo de México (ITAM) in Mexico City. She also holds a Master's degree in satellite applications engineering from the Technical University Munich (TUM). Her main research area is sensor fusion. Her current research focuses on LiDAR, sensor fusion between LiDAR and GNSS/INS, and relative and absolute navigation algorithms particularly for terrestrial applications.

Andreas Schütz is currently a research associate at the Universität der Bundeswehr München at the Institute of Space Technology and Space Applications. He holds a Bachelor's and Master's degree in Geodesy and Geoinformation from the Technical University of Munich (TUM). His research interests span GNSS Precise Point Positioning, Sensor Fusion and Integrity for autonomous applications.

Dominik Dötterböck received his diploma in Electrical Engineering and Information Technology from the Technical University Munich. Since 2007, he is a senior research associate at the Universität der Bundeswehr München at the Institute of Space Technology and Space Applications. His current research interests include signal design, signal-processing algorithms for GNSS receivers and software receivers.

Jürgen Dampf works as a software development engineer at Rohde & Schwarz GmbH & Co. KG in the department for high-end spectrum analysis and as a research associate at the Universität der Bundeswehr München. He received his PhD in Navigation at the Graz University of Technology. Formerly he worked as a GNSS R&D engineer at Trimble Terrasat GmbH, as CTO at IGASPIN GmbH and as GNSS system engineer at IFEN GmbH. His research topics range from Bayesian direct position estimation, GNSS reflectometry, multi-sensor fusion and integrated navigation, synthetic antenna beamforming, efficient GNSS signal processing algorithms for desktop and embedded platforms and jamming/spoofing signal generation and mitigation techniques.

Prof. Thomas Pany is with the Universität der Bundeswehr München at Space Systems Research Center (FZ-Space) where he leads the satellite navigation unit LRT 9.2 of the Institute of Space Technology and Space Applications (ISTA). He teaches navigation focusing on GNSS, sensors fusion and aerospace applications. Within LRT 9.2 a good dozen of full-time researchers investigate GNSS system and signal design, GNSS transceivers and high-integrity multi-sensor navigation (inertial, LiDAR) and is also developing a modular UAV-based GNSS test bed. ISTA also develops the MuSNAT GNSS software receiver and recently focuses on smartphone positioning and GNSS/5G integration. He has a PhD from the Graz University of Technology (sub auspiciis) and worked in the GNSS industry for seven years. He authored around 200 publications including one monography and received five best presentation awards from the US Institute of Navigation. Thomas Pany also organizes the Munich Satellite Navigation Summit.

ABSTRACT

This paper summarizes recent developments carried out to improve the Multi-Sensor Navigation Analysis Tool (MuSNAT) developed at the Universität der Bundeswehr München. A stand-alone Universal Software Radio Peripheral (USRP) from

National Instruments with an embedded PC has been adapted to run MuSNAT in real-time and is characterized with the typical GNSS receiver parameters (front-end, no. of channels, power consumption). The extension to LTE (and 5G) processing as well as developments related to I/Q to intermediate frequency (IF) conversion are illustrated with the example of an multipath estimating discriminator. A C++ to MATLAB interface is introduced to allow rapid prototyping of sophisticated receiver algorithms. The interface is used for PPP processing (Galileo High Accuracy Service), LiDAR odometry, and deep GNSS/INS coupling. The performance of MuSNAT in post-processing on the most powerful desktop CPUs (AMD Threadripper) is characterized.

I. INTRODUCTION

Analysis and development of new navigation signals, services and receiver algorithms requires a continuous improvement of the used tool chains. At our institute, the research focus is currently on multi-antenna systems, LTE/5G signal processing and testing the upcoming Galileo services OS-NMA (Open Service Navigation Message Authentication) and HAS (High Accuracy Service). The research work is partly done using the MuSNAT which resulted in significant feature updates for this tool [1]. Furthermore several projects require to use the software receiver in real-time as stand alone unit to allow for a live demonstration of the conducted developments.

This update shall be covered in this work and includes development of

- hardware (front-end, FPGA and CPU) architecture plus real-time GNSS performance of a stand alone software receiver
- a C++ to MATLAB interface for easy prototyping of PPP, LiDAR and deep coupling algorithms
- extending GNSS reference waveforms (e.g. PRN codes) by a LTE SSS+PSS waveforms (with an extension to 5G synchronization sequences)
- replacing of the fs/4 conversion by a Hilbert transform to convert I/Q input samples to the internally used real-valued sample format,
- tuning of the software to support modern many-core CPUs like AMD Threadripper.

The paper concludes with an outlook of planned updates in 2022.

II. STAND-ALONE-RECEIVER

Up to now the MuSNAT software receiver was running on a dedicated PC with an external USRP as GNSS front-end [1]. However, recent integration efforts from Ettus Research resp. National Instruments combined a powerful generic software radio front-end with an decent x86 PC capable of running the Windows 10 operating system to a product called USRP-2974 and which is shown in Fig. 1. The x86 CPU, an Intel Core i7-6822EQ with a release date Q4/2015, connects to the front-ends ADCs via four lanes of a second generation PCI interface thus allowing to transfer at maximum 2 GByte/s of signal samples. This device allows to run MuSNAT in real-time mode much like a conventional GNSS receiver. The total power consumption has been measured at the 230 V power supply to be at least 43 W in idle mode (only Windows 10 running). With the front-end software running (currently written in LabVIEW) and MuSNAT only receiving samples (but no acquisition and tracking) the power consumption is measured to be at least 77 W and with MuSNAT tracking 18 channels no significant increase in power could be measured (cf. Fig. 2. If in parallel the FFT based acquisition is active (e.g. every 10 s), the power consumption further increases to 95 W. The CPU load was indicated by Windows to be at least 33 % during tracking and nearly 100 % with the acquisition running in parallel. The power measurement showed positive variations up to a few Watt most likely due to other tasks performed by the OS. The MuSNAT receiver can be controlled with a tablet/PC via remote network control or by an external screen and keyboard.

The stand-alone USRP-2974 consists of an onboard processor, a field programmable gate array (FPGA), and the RF front-end. Unfortunately, National Instruments does not support Window 10 and RF daughterboards for the four-channel operation directly. Therefore, we made the following modifications:

- **Operating System:** change the operating system from NI Linux Real-Time to Windows 10
- **RF daughterboards:** replace the two UBX transceiver boards with two TwinRX receiver boards to allow the reception of up to four frequency bands
- **Firmware:** modify the flash memory of the build-in USRP. The modification of the flash memory allows the transformation to a four channel USRP known as USRP-2955

As already stated, the stand-alone USRP-2974 is shipped with only a real-time operating system and a single USRP configuration, which has just two receiving channels. The compact form factor is perfect for a GNSS receiver, which is why we undertook the modification, which partially affects the warranty. The onboard PC with the model “conga-TS170” is from the company

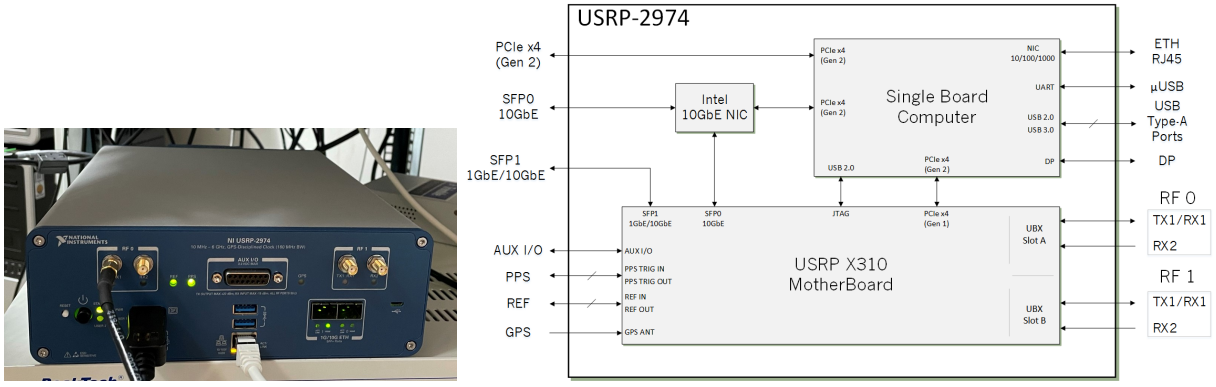

Figure 1: Picture of the USRP-2974 at ISTA and a high level block diagram [2]

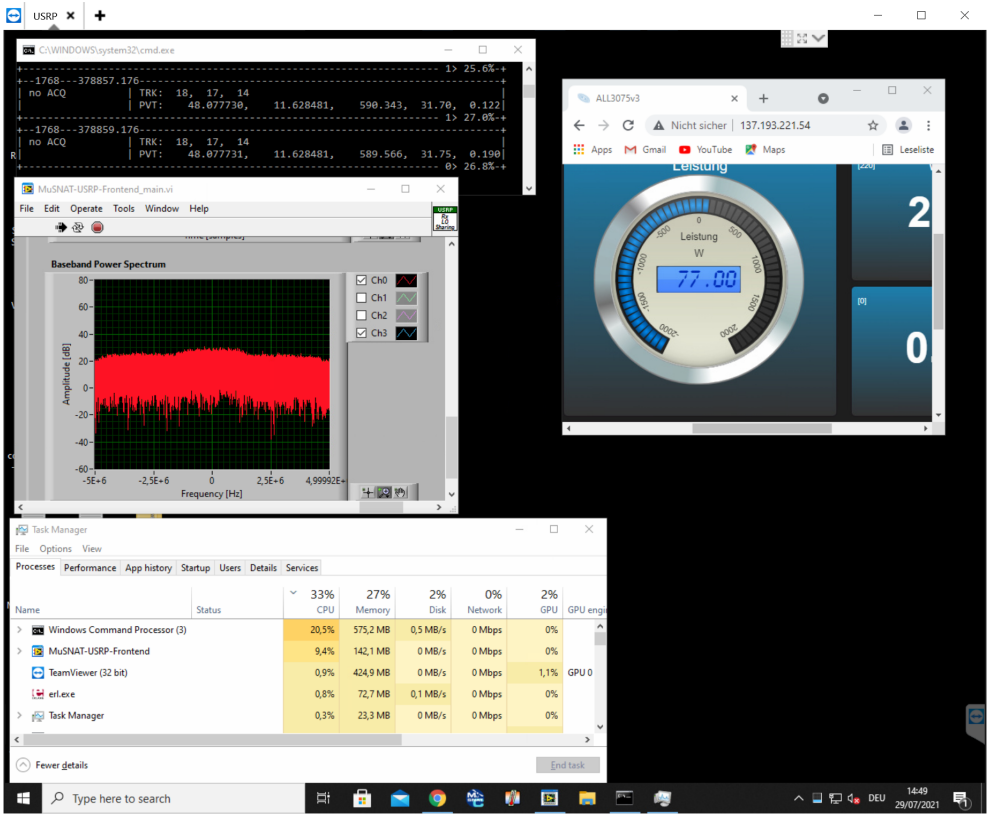

Figure 2: Screenshot of USRP-2974 running MuSNAT ASCII-version, the LabView front-end software, Windows task manager showing CPU-load and external power meter [2]

congatech GmbH and supports officially also Windows 10 as operating system. This modification should not influence the warranty of the USRP-2974, but support from NI can't be expected with this hardware and configuration. However, LabVIEW is native for Windows 10 and therefore nothing is against this modification. The integrated USRP basically matches the model number USRP-2954, which provides two RF receiving channels with a bandwidth of 160 MHz each. In principle, the entire frequency bands of all GNSS systems can be covered. With the ability to also receive LTE/5G signals to support the positioning solution, more than two RF channels are needed. Which is why it is necessary to upgrade to a USRP-2955 by changing the RF daughterboards. With this modification, one unfortunately loses the warranty of the USRP-2974, because a hardware and firmware change is carried out.

MuSNAT shall act as a reference receiver and therefore a good understanding of the RF characteristics is necessary and will be given in the following sub-section. Afterwards the processing capabilities in terms of number of GNSS channels are analyzed.

1. Front-End Modification and Description

The front end of the USRP-2974 provides in his original configuration two receiving RF channels with a maximum bandwidth of 160 MHz. For the capability to capture also LTE and 5G signals, additional input channels with indepentent tuning are required. The modification of the RF daughterboards from UBX to TwinRX leads to four RF channels with a maximum RF bandwidth of up to 80 MHz for each channel. For this hardware modification, a firmware adjustment within the EEPROM of the motherboard X310 (see high level block diagram in Figure 1) is additionally required according to the instructions of [3]. The block diagram

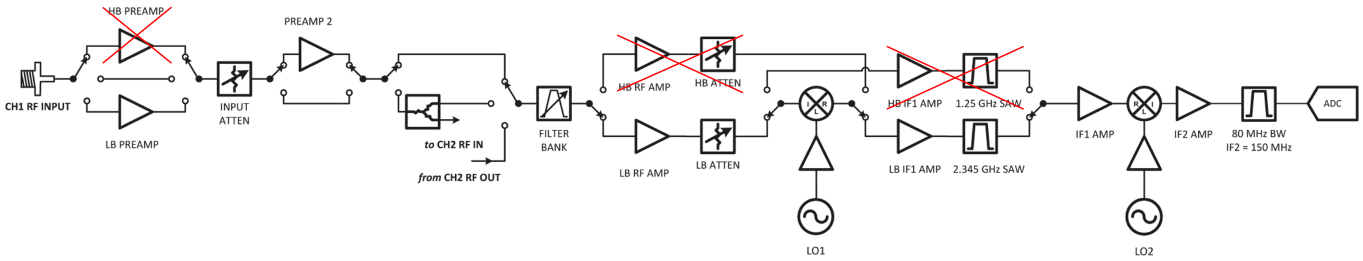

Figure 3: Block diagram of the TwinRX daughterboard used in our modified stand-alone USRP-2974 (this slightly adapted block diagram is based on [2]); GNSS needs only the low-band (LB) RF/IF path

of the TwinRX daughterboard is plotted in Figure 3 taken from [2] and slightly adapted for our purpose. The frequency range for the center frequency is from 10 MHz to 6 GHz, so that LTE and 5G can be captured directly in the majority of cases. The RF chain of the TwinRX is divided into two frequency ranges, which is the low-band (LB) from 10 MHz to 1.8 GHz and the high-band (HB) from 1.8 GHz to 6 GHz. Since GNSS is the core feature of the MuSNAT receiver, only the GNSS specific RF path is discussed here. GNSS requires only the low-band branch, which is why the high-band branches have been crossed out in Figure 3 for a better overview. The TwinRX has a superheterodyne architecture with the feature of four channels (two TwinRX per USRP), which are independent tunable, RF shielded, and capable of local-oscillator (LO) sharing for phased-aligned operations.

Each signal passes through a few RF stages, two intermediate frequency stages with different frequencies and the final digitization by sub-sampling. The individual components in the signal chain are listed sequentially in the following. The bandwidths and frequency ranges for each stage will be highlighted, which is helpful especially for RF interference considerations. The **first stage (RF conditioning)** consists of a bypassable high-pass filter (0.8 GHz HPF), a 14-dB amplifier (PGA-102+ of Mini-Circuits), a programmable attenuator (PE43503 of Peregrine Semiconductor, 31 dB), and another 14-dB amplifier of the same type. The special feature is that all components except of the programmable attenuator can be switched out of the signal chain. This allows the user to decide with his configuration to skip the first stage completely. Because of the bypassable high-pass filter, **the bandwidth of the first stage is 5.2 or 5.99 GHz**. The input frequency is **limited to 6 GHz**. The lower frequency is limited to **10 or 800 MHz**. With frequencies for GNSS, it can be assumed that the high-pass filter is always active, at least in the standard configuration of the USRP. A deactivation could be justified in individual cases by a performance advantage with regard to the noise figure. The noise figure of the amplifier is given as 2.4 dB, which offers an extremely high dynamic range. There are significantly better LNAs in terms of noise figure for GNSS. The TwinRX board allows the use of other and application-specific (external!) LNAs by switching off the entire first stage. The data sheet of the TwinRX board gives a noise figure of less than 5 dB for GNSS applications. Since the noise figure is also dependent on the selected gain, this would have to be measured in the same way as it was done for the USRP-2952 in [4]. The **second stage (RF conditioning)** consists of a filter bank, another programmable attenuator, another 14-dB amplifier, and two ceramic low-pass filters (LFCN-1700+ of Mini-Circuits). The filter bank has eight different filters. Four of them are marked as low-pass (LP) or high-pass (HP). For most GNSS bands, the fourth filter bank with a ceramic high-pass filter (HFCN-1100+ of Mini-Circuits) is applied. **The bandwidth is then 680 MHz with a frequency response of 1.16 to 1.85 GHz.** For central frequencies lower than 1.2 GHz - like Galileo E5 - the third filter bank with a high-pass and low-pass filter (HFCN-740+ and LFCN-1000+ of Mini-Circuits) is active. **The bandwidth is then 480 MHz with a frequency response of 0.76 to 1.24 GHz.** The **third stage (IF conditioning)** starts with a bidirectional mixer consisting of a RF diplexer (2450DP15D of Johanson Technology), a wide-band frequency mixer (SIM-722MH+ of Mini-Circuits) and another diplexer (DPX202700DT of TDK). This mixer stage sets the **first intermediate frequency to 2.345 GHz**. The second part of the third stage significantly restricts the **bandwidth to 90 MHz**. The signal chain is composed of a SAW filter (TA0581A of TAI-SAW Technology) and a 14-dB amplifier in dual alternating design. **Stage 4 (IF conditioning)** uses a mixer (LTC5510 of Linear Technology, today: Analog Devices) to set the signal to a second intermediate frequency, followed by a low-pass filter (LFCN-225+ of Mini-Circuits), a 14-dB amplifier and a custom bandpass filter designed by NI. The bandwidth of this stage is 40 or 80 MHz, which is determined by the frequency of the mixer. **With an intermediate frequency of 125 or 150 MHz, 40 or 80 MHz is obtained, respectively.** Stage 6 (digitization) prepares the signal with a differential amplifier (ADL5565 of Analog Devices) for the ADC of the main board (ADS62P48 of Texas Instruments). The sampling is performed at 200 MHz. In this case, it is a systematic undersampling (or bandpass sampling), since the preceding filtering avoids aliasing effects. Due to the Nyquist theorem, a bandwidth of 100 MHz is available here, since it is a real sampling. The DSP in the FPGA provides the signal to the desired bandwidth of the user in a complex format, i.e. it converts the real-valued IF samples to complex valued IQ samples

(a process which is reverted inside the MuSNAT receiver). The meaningful selectable gain of the USRP-2955 is between 0 and 90 dB, which is achieved by the first two switchable 14-dB amplifiers and the two programmable attenuators with 31 dB each. In front of the filter bank, there is a switchable splitter for sharing to the adjacent channel, which would add another 3 dB of attenuation. Another 6 dB gain could also be induced by the ADC.

A LabVIEW program called "UIM (USRP Interface for MuSNAT)" has been developed to configure the front-end and to collect samples in 16-bit I/Q format for a maximum of 2 RF bands and a sample rate of 20 MHz in real-time. The samples are provided via a TCP/IP based server to MuSNAT. The program is computationally intensive and will be upgraded as discussed in sect. VII. For data collection only, higher sampling rates can be achieved.

2. GNSS Baseband Computational Capability

The onboard processor is an Intel® Core™ i7-6822EQ 2 GHz quad core processor, which is good enough to run the MuSNAT software receiver and simultaneously the USRP front-end server application for the real-time processing.

The real-time behaviour can nicely be measured with the Intel VTune performance profiler [5]. The MuSNAT processing core generates markers each time a integrate-and-dump epoch (while tracking GNSS signals) is reached (i.e. at the end of the primary code). Furthermore the duration of one IF sample packet can be shown within the profiler. The resulting timing diagram can be found in Fig. 4. An input IQ sampling rate of 10 MHz has been selected by the front-end software which is converted inside MuSNAT to 20 MHz (real-valued IF samples). This conversion and the conversion to the 16-bit and 8-bit MuSNAT-internal formats is indicated as 'Samp. Prep.' in Fig. 4. For each GNSS service two OpenMP threads are responsible for tracking. The figure shows clear differences for the different integration times (1 ms for GPS C/A, 4 ms for Galileo E1B/C). The GPS C/A and the E1C signal were processed with 3 correlators (early, prompt, late) and the E1B signal with 5 correlators (very early, early, prompt, late, very late) to ensure correct lock onto the main CBOC-peak.

The brown areas in Fig. 4 indicate the CPU-core/thread to be busy. Red areas indicate synchronization tasks. From the length of the brown areas compared to the packet length, one could roughly estimate that 3 times more channels can be processed still keeping the real-time capability.

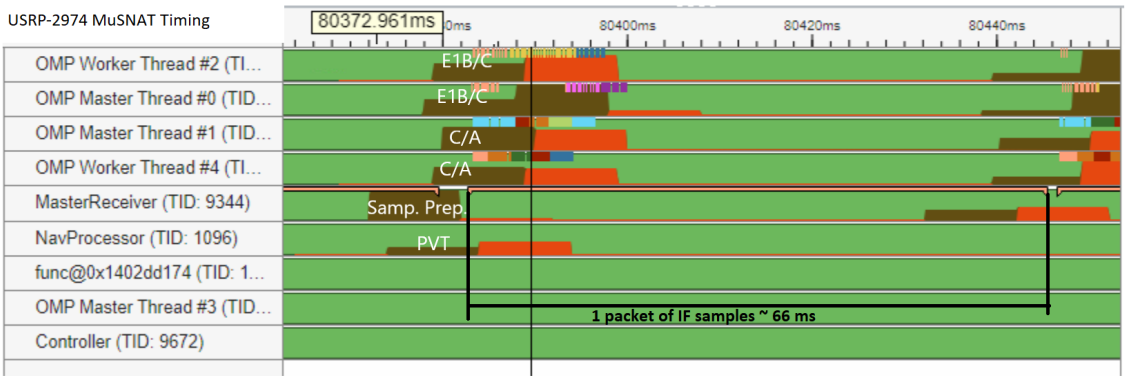

Figure 4: Intel VTune 2021.2.0 timing diagram of MuSNAT on the USRP-2974 tracking 10 GPS C/A and 6 Galileo E1B/C signals in real-time using the fast correlator

To better understand the results of the USRP-2974 complementary tests have been conducted on recent laptop computer equipped with a Intel i7-10875H CPU (release date Q2/20). The GPS C/A and Galileo E1 receiver settings were identical as for the USRP-2974 case, but post-processing was used. Two different correlation algorithms were tested:

- Precise: in the precise mode, the replica generation is done on a continuous basis for each integration interval based in the current NCO values (like in a conventional receiver). All samples are represented by 16-bit integers and the correlation uses the IPP function ippsDotProd (16-bit to 64-bit) (indicated as func@0x1803a77e0 in Fig. 5
- Fast: this mode uses 8-bit unsigned values to represent the the signal samples and assembler code to do the correlation. Replica signals (code and carrier) are read from a look-up table, which is generated if the current NCO values deviate from the values used to generate the previous lookup table more than a certain threshold (e.g. typically a few meters for low-bandwidth signals)

Fig. 5 shows the timing result. The line 'MasterReceiver' indicates the duration of the sample packet, which covers in this case a signal duration of 34 ms. A sample rate of 20 MHz (real-valued IF samples) has been used. The packet was processed (15 signals are tracked) in precise mode in around 7 ms and in the fast mode in around 2.8 ms Four OpenMP-threads were used for tracking

(CPU has 8 cores). The overall processing speed was limited by the Galileo signal processing. In both cases the actual correlation (i.e. the dot-product operation) consumes most of the processing time, however, in the precise mode the replica generation shows up as a significant contribution. The correlation of the fast mode involves a number of conversions to reformulate the dot-product operation with unsigned integers. Nevertheless, the use of 8-bit values reduces the required memory bandwidth (compared to the 16-bit samples) and results finally in roughly two times faster processing.

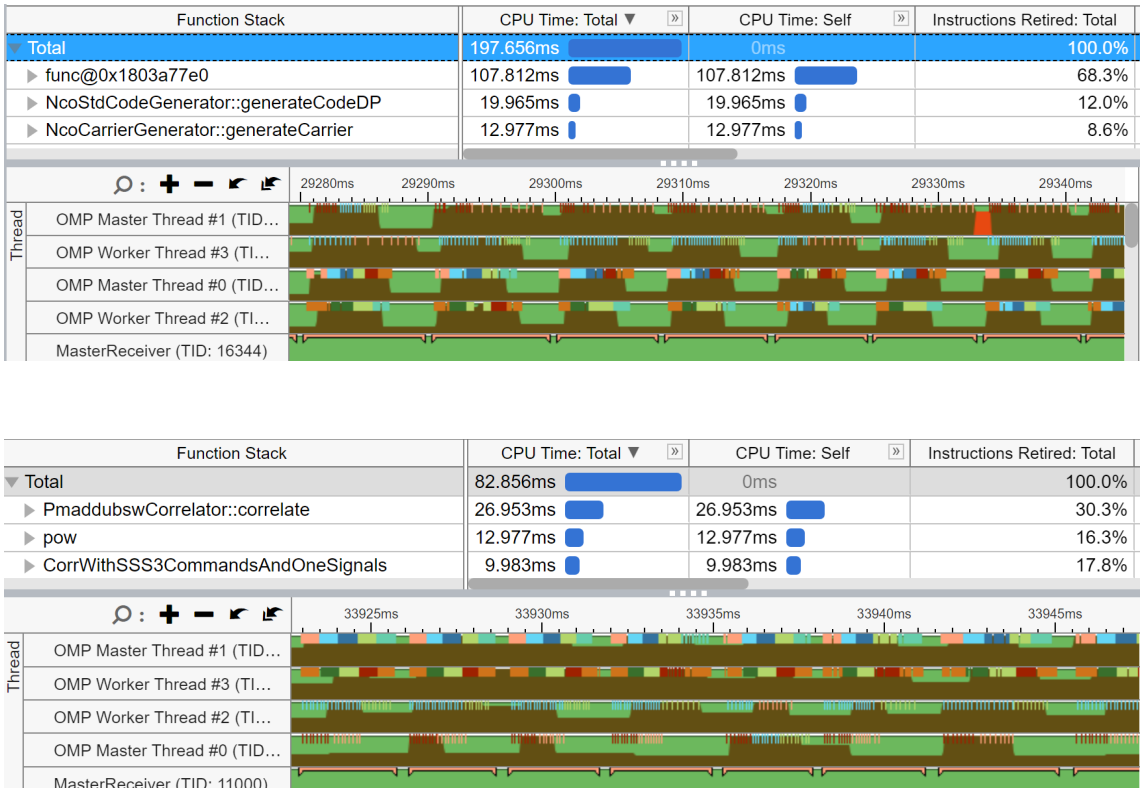

Figure 5: Intel VTune 2021.2.0 timing diagram of MuSNAT on a high-performance laptop tracking 10 GPS C/A and 5 Galileo E1B/C signals in post-processing (upper: 16-bit correlator with precise code/carrier NCO, lower: 8-bit correlator with dynamic look-up-table correlator

III. CONSIDERATION FOR IF AND AND I/Q SAMPLE REPRESENTATION

The MuSNAT works internally with real valued samples to benefit from the higher Nyquist sample rate compared the complex valued Nyquist rate. The higher sampling frequency of the real-valued samples allows for a lower limit of the correlator spacing. More precisely, the software receiver works internally with real-valued sample batches

$$\mathbf{r} = [s_{\mathbf{r},\mathrm{I},i=0}, s_{\mathbf{r},\mathrm{I},i+1}, \ldots, s_{\mathbf{r},\mathrm{I},N_\mathbf{r}-1}] \tag{1}$$

where $\mathbf{r}$ is a vector containing $N_\mathbf{r}$ samples, $s_{\mathbf{r},\mathrm{I},i}$ is a real-valued sample which belongs to the vector $\mathbf{r}$ and I denotes the In-Phase component and i is the sample index. The samples are recorded with the sample rate $f_{s,\mathbf{r}} = 1/T_{s,\mathbf{r}}$ in Hz, where $T_{s,\mathbf{r}}$ is the sample interval in seconds. The real-valued samples are located at an intermediate frequency, most reasonable at the optimum of $f_{\mathrm{IF},\mathbf{r}} = f_{s,\mathbf{r}}/4$ to exploit the full signal bandwidth. But most RF frontends (including the USRP 2974) deliver the samples in the complex In-Phase/Quadrature (I/Q) format such as

$$\mathbf{c} = [s_{\mathbf{c},\mathrm{I},j=0}, s_{\mathbf{c},\mathrm{Q},j=0}, s_{\mathbf{c},\mathrm{I},j+1}, s_{\mathbf{c},\mathrm{Q},j+1}, \ldots, s_{\mathbf{c},\mathrm{I},N_\mathbf{c}-1}, s_{\mathbf{c},\mathrm{Q},N_\mathbf{c}-1}] \tag{2}$$

where the subscript Q denotes the Quadrature component and j the sample index of the complex I/Q pair. For the complex valued samples the baseband signal is typically located at the optimum of $f_{\mathrm{IF},\mathbf{c}} = 0$ Hz, to cover the maximum target signal bandwidth. It should be noted that according to Nyquist the complex valued sample stream requires just half the sample rate of the real valued sample stream $f_{s,\mathbf{c}} = f_{s,\mathbf{r}}/2$ and $N_\mathbf{c} = N_\mathbf{r}/2$ to cover the same signal bandwidth. As the samples are often delivered in I/Q

baseband samples, one of the very first steps in the signal processing chain of the receiver is the conversion to real valued samples. This step is currently done with a fs/4 mixer. This conversion is computationally very efficient and achieves the sampling rate increase before mixing simply by interpreting the Quadrature sample of the I/Q pair as an independently measured and 90 deg phase shifted In-phase sample. The definition of the used fs/4 mixer is given as

$$\mathbf{r} = \mathbf{c}\,\mathbf{m} \tag{3}$$

with

$$\mathbf{m} = \text{sign}(\cos(2\pi f_{\text{IF},\mathbf{r}} t + \pi)) \quad \forall \quad t = [0, 1T_{s,\mathbf{r}}, 2T_{s,\mathbf{r}}, \ldots, (N_{\mathbf{r}} - 1)T_{s,\mathbf{r}}] \tag{4}$$

which results in a mixer support sequence of $\mathbf{m} = [1, -1, -1, 1, \ldots, 1, -1, -1, 1]$. Note that the multiplication in (3) is element-wise. This further results in a trivial realization of the fs/4 mixer by a reassignment of the samples such that

$$\begin{aligned} s_{\mathbf{r},\text{I},i} &= s_{\mathbf{c},\text{I},j} \\ s_{\mathbf{r},\text{I},i+1} &= -s_{\mathbf{c},\text{Q},j} \\ s_{\mathbf{r},\text{I},i+2} &= -s_{\mathbf{c},\text{I},j+1} \\ s_{\mathbf{r},\text{I},i+3} &= s_{\mathbf{c},\text{Q},j+1} \\ &\vdots \end{aligned} \tag{5}$$

The conversion step in (3) causes doubling the sample rate to $f_{s,\mathbf{r}} = 2f_{s,\mathbf{c}}$ and a shift of the intermediate frequency from $f_{\text{IF},\mathbf{c}} = 0$ Hz to $f_{\text{IF},\mathbf{r}} = f_{s,\mathbf{r}}/4$. The algorithm follows a sample-hold scheme for the Quadrature samples to convert them into In-Phase samples. This conversion introduces a second signal which is delayed by 1 sample. This additional delayed signal can be interpreted as a multipath signal which influences the resulting correlation function and derived parameter estimates of the GNSS signals like the C/N0, code phase, carrier phase, etc. It should be noted, that the delay of the multipath component after mixing depends on the sample rate $f_{s,\mathbf{r}}$.

This definition of a fs/4 mixer is computationally very efficient and might be sufficient for all kind of conventional positioning algorithms, but is suboptimal for high-end signal analysis purposes. As an alternative it is proposed to use a sinc-interpolation for the upsampling step realized with Fast Fourier Transform (FFT) methods.

1. FFT-based IQ->IF conversion

The conversion from complex IQ samples to real-valued IF samples can be done in three major steps, an

1. sinc-upsampling step to $2f_{s,\mathbf{c}}$ to obtain $\mathbf{c}_{\text{u}}$
2. mixing the signal from $f_{\text{IF},\mathbf{c}} = 0$ Hz to $f_{\text{IF},\mathbf{c}} = 2f_{s,\mathbf{c}}/4 = f_{s,\mathbf{r}}/4$ to obtain $\mathbf{c}_{\text{u,IF}}$
3. eliminate the imaginary component of $\mathbf{c}_{\text{u,IF}}$ to obtain $\mathbf{r}$

A optimal signal reconstruction for the upsampling step is exploit with a sinc-interpolation. The sinc-interpolation can be realized efficiently for a large number of samples with the use of FFT methods, as zero-padding in the frequency domain equals a sinc-interpolation in the time domain. The upsampled complex samples $\mathbf{c}_{\text{u}}$ can be obtained from

$$\begin{aligned} \mathbf{c}_{\text{FD}} &= \text{FFT}(\mathbf{c}) \\ \mathbf{c}_{\text{FD,zp}} &= [\mathbf{c}_{\text{FD},0}, \ldots, \mathbf{c}_{\text{FD},N_{\mathbf{c}}/2-1}, 0_0, \ldots, 0_{N_{\mathbf{c}}-1}, \mathbf{c}_{\text{FD},N_{\mathbf{c}}/2}, \ldots, \mathbf{c}_{\text{FD},N_{\mathbf{c}}-1}] \\ \mathbf{c}_{\text{u}} &= \text{IFFT}(\mathbf{c}_{\text{FD,zp}}) \end{aligned} \tag{6}$$

where FFT and IFFT denote the discrete complex Fast Fourier Transform and Inverse FFT, $\mathbf{c}_{\text{FD}}$ is the complex valued frequency domain (FD) representation of the samples, $\mathbf{c}_{\text{FD},i}$ denotes a complex sample pair in the FD with it's real and imaginary part at index i and $\mathbf{c}_{\text{FD,zp}}$ denotes the zero-padded (zp) sample vector in the FD. To achieve hightest computational efficiency, e.g. with the Radix-2 algorithm, the input sample length needs to be $N_{\mathbf{c}} = 2^n$, where n is the FFT order. It should be noted, that the IFFT will be of order $n + 1$ due to the zero-padding. The resulting $\mathbf{c}_u$ is of length $N_{\mathbf{c}_u} = 2N_{\mathbf{c}} = N_{\mathbf{r}}$ with a sample rate of $f_{s,\mathbf{c}_u} = 2f_{s,\mathbf{c}} = f_{s,\mathbf{r}}$ and corresponding sampling period of $T_{s,\mathbf{c}_u} = T_{s,\mathbf{r}} = 1/f_{s,\mathbf{c}_u}$. The upsampling step doubles the signal bandwidth.

The mixing process to shift the signal from $f_{\mathrm{IF},\mathbf{c}_u} = 0$ Hz to $f_{\mathrm{IF},\mathbf{c}_u} = f_{s,\mathbf{c}_u}/4$ is defined with

$$\mathbf{c}_{\mathrm{u,IF}} \quad = \quad \mathbf{c}_{\mathrm{u}}\, e^{j2\pi(f_{s,\mathbf{c}_u}/4)t} \quad \forall \quad t = [0, 1T_{s,\mathbf{c}_u}, 2T_{s,\mathbf{c}_u}, \ldots, (N_{\mathbf{c}_u} - 1)T_{s,\mathbf{c}_u}] \tag{7}$$

The elimination of the imaginary part is trivial and is defined with

$$\mathbf{r} \quad = \quad \mathfrak{Re}(\mathbf{c}_{\mathrm{u,IF}}) \tag{8}$$

The elimination step of the imaginary parts half's the signal bandwidth by removing the negative frequency components. This is because real samples correspond to the symmetric spectrum and imaginary samples correspond to the asymmetric spectrum.

It should be noted, that the proposed algorithm does not account for windowing effects and the length of the sample batch equals to the FFT window size. In this case the FFT assumption of periodical signals causes ripples (interpolation errors) in the amplitude and phase at the resulting time domain sample batch edges. This can be improved by using a FFT window size larger than the sample batch, which can be easily implemented by using samples from the previous and next sample batch for the interpolation step.

2. Improved C/N0 and code discriminator noise

It is expected that the suboptimal behaviour of the algorithms become visible in the C/N0 and discriminators. Therefore, the fs/4 and FFT mixer are compared to a baseline scenario, when IF samples are directly processed without any prior conversion. The results are summarized in Tab.1 and shown in Fig. 6. Both reference signals were GPS L1 C/A code signals with a C/N0 of 50.00 dB-Hz, a quantization of 8-bits and filtered with a single-sided 3 dB signal bandwidth of 2.046MHz, the IQ signal has a sample rate of $f_{s,\mathrm{IQ}} = 2$MHz (with IF at 0MHz) and the IF signal has a sample rate of $f_{s,\mathrm{IF}} = 4$MHz (with IF at 1MHz).

	C/N0 [dB-Hz]	$1\sigma_{\mathrm{DLL}}$[m]
IF (baseline)	49.67	8.21
IQ->IF fs/4	49.08	8.84
IQ->IF fft	49.23	8.66

Table 1: This table summarizes the obtained results in C/N0 and 1σ std. dev. of the DLL discriminator.

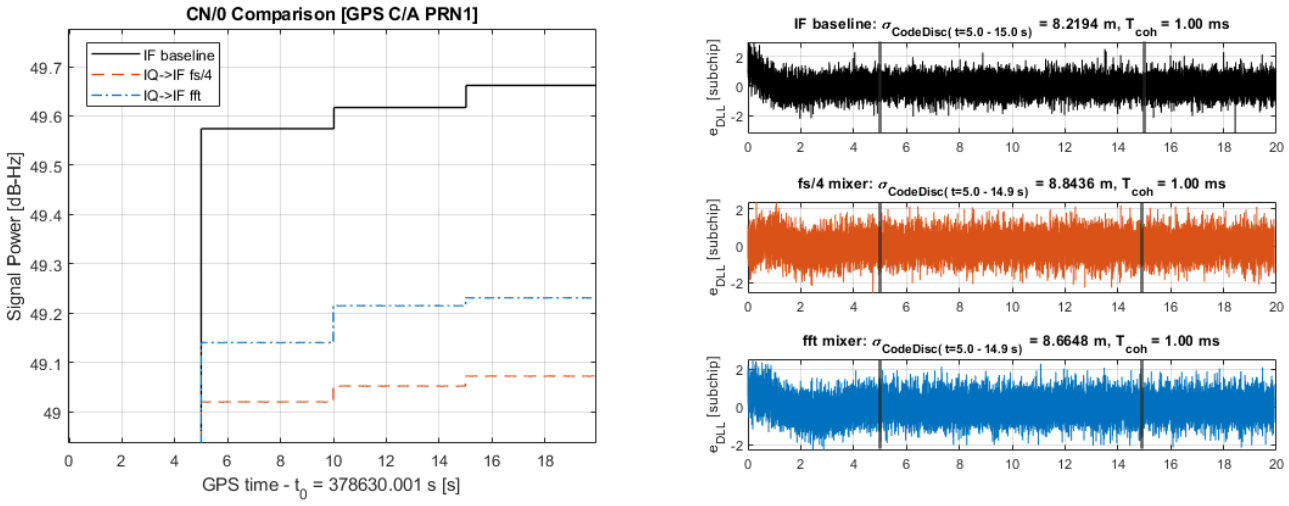

Figure 6: left: C/N0 comparison of using the IF samples directly (baseline) and IQ samples converted with the fs/4 and FFT mixer, right: DLL code discriminator output with 1σ std. deviation. The std. dev. was measured in a stable interval, especially after the pull-in range from 5s to about 15s (between the vertical lines)

The baseline scenario shows a C/N0 of 49.67dB-Hz with a slight internal receiver loss of about 0.33dB (this is explained by the fact, that the N_0 estimation inside the receiver does also capture the signal power C, thus provides a slight overestimation of N_0). The fs/4 mixer induces a further reduction of 0.59dB compared to the baseline scenario, which is quite significant. This corresponds to a increase of the 1σ std. dev. of the DLL discriminator of 0.63m. For IQ-signal analysis purposes this is unsatisfying and needs to be improved. The FFT mixer achieves an improved performance compared to the fs/4 mixer, but still not optimal. The loss in C/N0 to the baseline is 0.44dB, which corresponds to an increase of the 1σ std. dev. of the DLL discriminator of 0.45m. It is assumed, that the small FFT window size and thereof introduced ripples are responsible for the loss, but this needs to be further investigated.

The obtained results can be more precisely seen in Fig. 6. The C/N0 estimation algorithm averages over 5 seconds, which can be seen by the steps in the left C/N0 plot. The right plot shows the DLL discriminator values. For the std. dev. estimate a clean section between the vertical lines is considered. This is important to achieve a meaningful estimate without the pull-in phase.

3. Improved multipath estimation

A multipath (MP) estimating code discriminator (cf. chap. 8 of [6]) was used to investigate the improvement in performance while replacing fs/4 mixer with an FFT mixer for the IQ->IF conversion. The algorithm uses Least-Squares to estimate the signal parameters (code phase, doppler, complex-valued amplitude) for the LOS and multipath components. This is done by fitting these signal parameters of the multi-correlator values of the received signal onto a pre-computed correlation function for each coherent integration interval. A critical factor in the multipath estimation is the pre-computed correlation function. The accuracy of the estimation is improved when matching frontend parameters such as the filter bandwidth is taken into consideration while computing this correlation function. In order to perform the MP-estimation test on the three test cases, namely IF (baseline), IQ->IF conversion using fs/4 mixer and IQ->IF using FFT mixer, IF and IQ signals were first generated following which the correlation functions were computed. The settings under which the correlation function was computed was for a single sided 3dB bandwidth of 2 MHz, sampling rate of 10 MHz, IF of 2.4 MHz. The choice of lower sampling rates was used to identify the differences between the two mixers. Two correlation functions were computed over a code phase span of 4 chip periods. The first correlation function was the correlation between the bandlimited and infinite bandwidth version of the generated signal and the second one between two infinite bandwidth versions of the generated signal. It is important to remove any S-curve bias in the computed correlation functions to guarantee a proper fit. The two correlation functions along with their first and second derivatives after application of the correction bias are seen in Fig. 7.

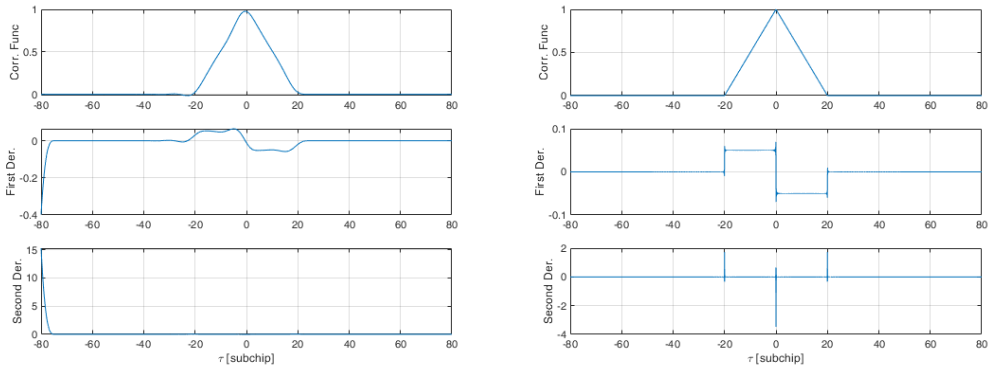

Figure 7: left: Computed correlation function between bandlimited and infinite bandwidth signal and its derivatives, right: Computed correlation function between two infinite bandwidth versions and its derivatives)

The computed correlation functions were installed onto the receiver configuration file to aid in MP-estimation. As a test case a clean GPS C/A signal was generated in IF and IQ format using the MATLAB/CUDA based signal generator YASST and was acquired and tracked by MuSNAT. Each epoch (i.e. every 1 s) the MP estimating discriminator was applied making use of correlation values collected in the previous 1-s interval. The resulting plots for the three cases are seen in the Fig. 8 and Fig. 9. Lower chi-square values indicate good fit of the pre-computed correlation function model and a reliable MP-estimation. The MP-estimation results in Fig. 9 show an absence of multipath components for the IF case and when choosing the FFT mixer. In case of the fs/4 mixer the MP-estimating discriminator (incorrectly) identifies one multipath component which is most likely an expected artefact from the fs/4 mixer, which does a time-shift of 1-sample for all Q-samples.

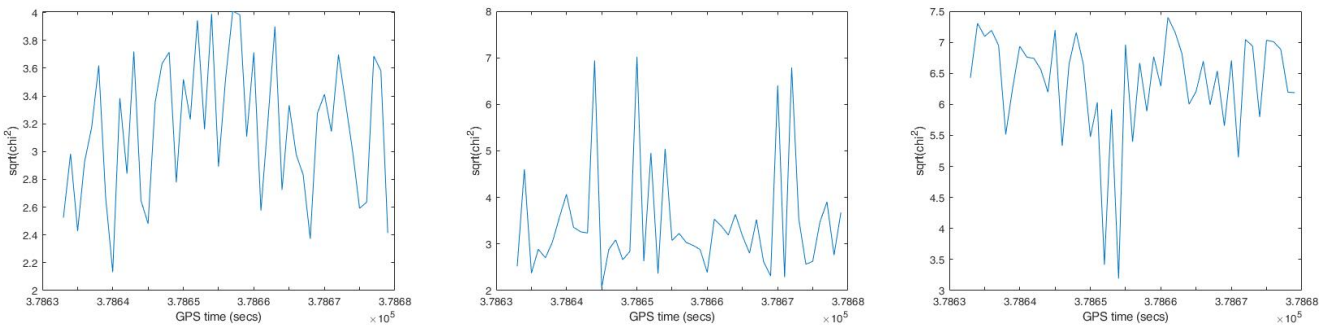

Figure 8: Square root of chi-square values, left: IF, middle: IQ->IF using fs/4 mixer, right: IQ->IF using FFT mixer

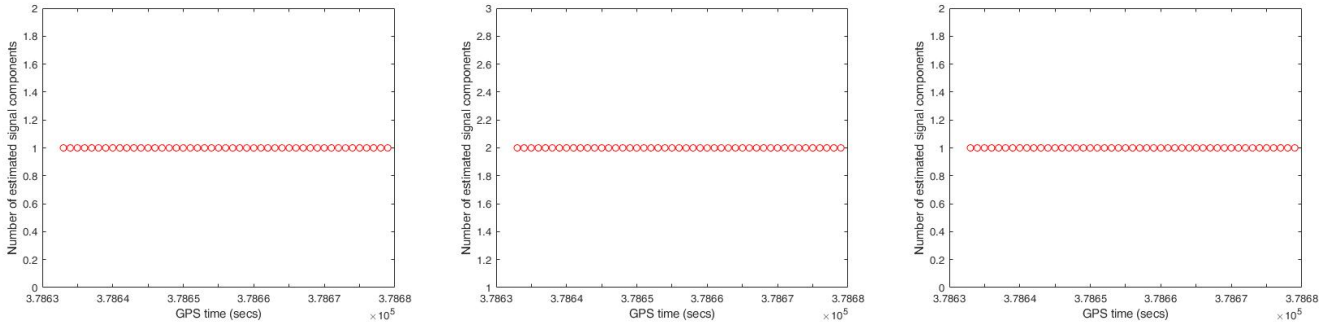

Figure 9: Number of estimated signal components (1= LOS, 2= LOS+1 Multipath), left: IF, middle: IQ->IF using fs/4 mixer, right: IQ->IF using FFT mixer

IV. COMBINED LTE/GNSS SIGNAL PROCESSING CAPABILTIY

MuSNAT provides LTE signal processing capabilities based on the LTE synchronization signals, namely primary synchronization signals (PSS) and secondary synchronization signals (SSS). For the processing in MuSNAT, only downlink signals are considered and the LTE base stations act as pseudolites; each LTE basestation has its unique SSS+PSS signature like the GNSS PRN code. The signature allows to read the LTE transmit time (modulo the frame length) and to measure carrier phase and Doppler.

The LTE synchronization signals are generated based on the 3GPP standard [7]. The LTE processing capability has been included solely for the frequency division duplex (FDD) transmission mode of LTE, which uses different frequency bands for the uplink and downlink. In this transmission mode, the PSS and the SSS are transmitted in every 10 ms-long frame, which contains 10 subframes, each 1 ms long. The PSS and the SSS are transmitted within the subframes 0 and 5. The PSS sequence is a unique sequence that is repeated every 5 ms, while the SSS sequence changes depending on the subframe (or slot) number in which is transmitted, i.e., the SSS in the subframe 0 and 5 are different. As the sequences are repeated every frame (10 ms), both SSS sequences are used for synchronization. The different SSS sequences, allow communication receivers to perform a frame (or time) synchronization.

The acquisition and tracking of the LTE signals can be performed by independently acquiring and tracking the PSS and the SSS, as shown in [8], or by generating a combined 10 ms-long (1 LTE frame long) sequence containing the PSS and SSS sequences. The latter option has been the design choice for the implementation of the LTE signal processing in MuSNAT. This simplifies the PCI detection chain within MuSNAT as both PSS and SSS are detected simultaneously. However, the cross-correlation with other signals must be considered, specially for those synchronization signals sharing the same PSS sequence. The PSS sequence can take a value from 0 to 2, and this means that for different LTE transmitters with physical cell identification (PCI) numbers that are modulo-3 appart, the PSS will repeat (e.g. PCI = 101, PCI = 104, and PCI = 107 will share the same PSS sequence). The cross-correlation of two signals sharing the same PSS sequence will suffer from high cross-correlation peaks, and consequently, the detection threshold needs to be carefully chosen.

Fig. 10 presents the acquisition plot of the LTE signal recorded in Neubiberg in the institute's location at the center frequency of 796 MHz by a static receiver. The figure presents the correlation output of a detected cell ID (CID), i.e. CID 41, and the cross-correlation with nearby CIDs, i.e. CIDs 38, 39, and 40, in the exact timestamp. One point observed is the high cross-correlation between the CID 38 and 41, this is caused due to the common PSS sequence, which appears twice in the 10 ms-long combined sequence. However, it can be observed, that the correlation amplitude is still low in comparison with the present CID.

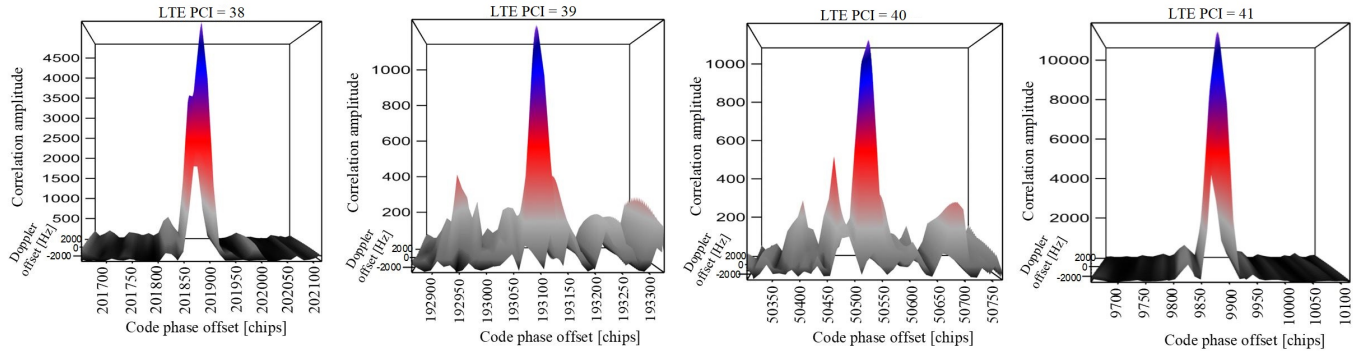

Figure 10: Frame synchronization with combined PSS and SSS sequences at time tag 6.30784 s, for LTE PCIs 38, 39, 40, and 41 with the highest peak for the LTE PCI 41 (the rightmost), corresponding to the present signal. The other PCIs present the cross-correlation with non-present signals, where for PCI 38 (the leftmost) the high cross-correlation due to same PSS as PCI 41 is shown.

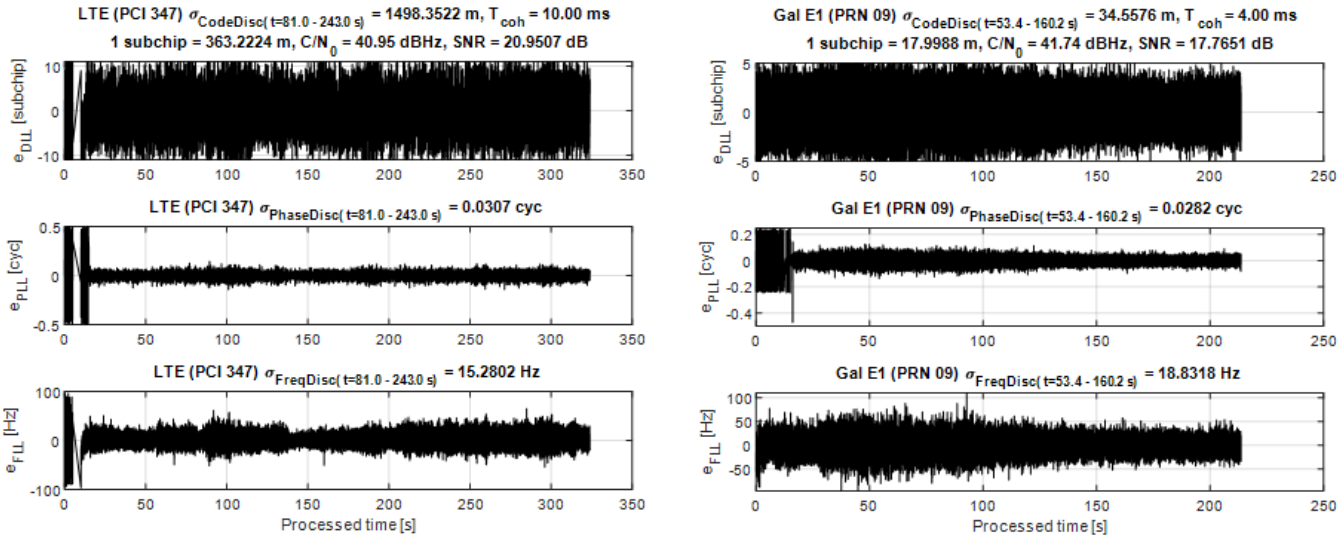

Figure 11: Discriminator values of the joint GNSS+LTE processing over the receiver time provided by MuSNAT. On the left the DLL, FLL and PLL discriminator values for LTE PCI 347 is presented by using the joint PSS and SSS sequences. On the right, the DLL, PLL, and FLL discriminator values are presented for the Galileo PRN 09.

For the same dataset, the tracking functionality is presented in Fig. 11 and Fig. 12 for the LTE signal with PCI 347, and for the Galileo E1 PRN 09 signal which were captured simultaneously with the same USRP and two different RF ports. Fig. 11 presents the DLL, PLL and FLL, discriminator values for each one of the mentioned signals. As it can be observed, a stable tracking can be achieved for both signals. Fig. 12 presents the output of the in-phase (I) and quadrature-phase (Q) correlators for those same signals. In the case of the LTE correlator output, the figure shows in each correlator plot the real part (addressed in red under the name Pilot) and the imaginary part (addressed in blue under the name Data) of the LTE combined PSS and SSS sequence. The LTE signal can thus be seen (from the GNSS perspective) as two pilot signals is phase qudrature).

V. EMBEDDING MATLAB ALGORITHMS

From an overall perspective, considering its numerous capabilities, the MuSNAT is a powerful suite with a vast spectrum of functionalities. This however comes with a downside: Implementing, developing and testing additional features, may become quite cumbersome, due to the use of low-level C++ code. By incorporating MATLAB's 2018 reworked matlab::engine::MATLABEngine into the MuSNAT, we enable quick and easy integration and modification of advanced algorithms (currently only used for PPP-AR, LiDAR odometry and GNSS/INS coupling)). Depending on the requirements, the respective object in the MuSNAT may either create its own MATLABEngine instance or share it with others. Data is then passed to and retrieved from MATLAB in all available proprietary formats by directly declaring and reading variables in the MATLAB workspace, rendering the interface bidirectional. Additionally, visualization and plotting routines may be used for in-run visual debug output. This overall setup now enables two key features: First, algorithms may be developed solely in MATLAB, using

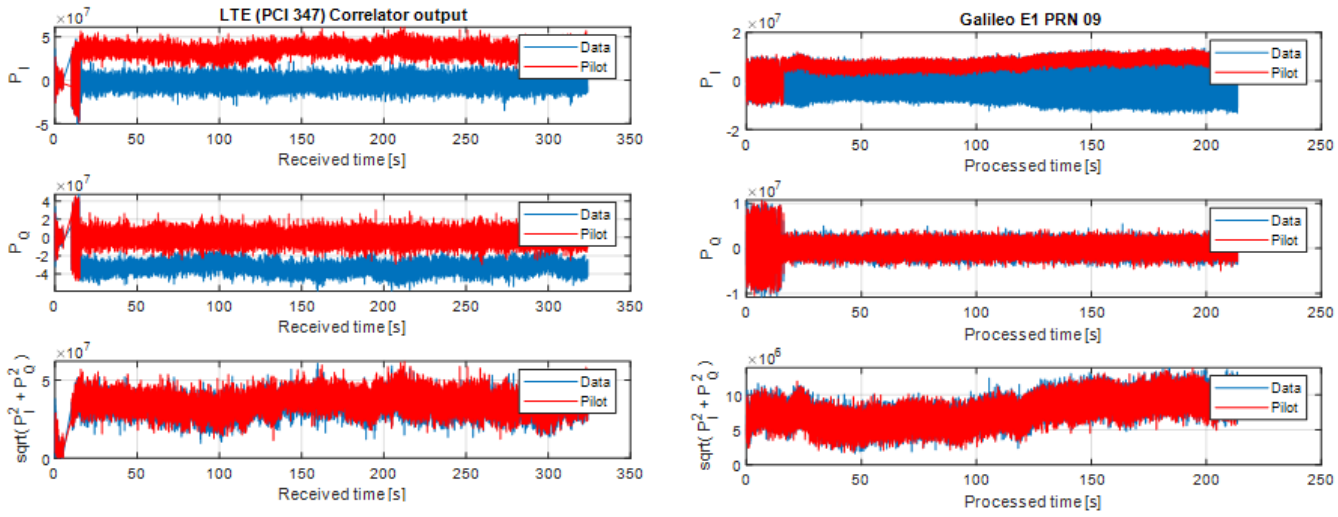

Figure 12: Correlator values for the joint GNSS+LTE signal processing provided by MuSNAT. On the left, the correlator values of the LTE PCI 347 are shown. On the left, the correlator values for the Galileo PRN 09 are shown.

all its accommodations and allowing for a MuSNAT independent development and verification environment. Secondly, easy integration into MuSNAT and modification without compilation becomes possible. However, in order to access this feature, the user has to possess a valid installation and license of MATLAB version 2018a or later (for using the MuSNAT-Matlab interface). If the user opts for full LiDAR and PPP-Module support, Matlab 2021a or later is necessary

1. PPP-Engine

The general data flow and implementation method of the MuSNAT PPP-Engine is depicted in Fig. 13. The following description covers the status obtained in Aug. 2021. Corrections to the raw pseudorange and phase measurements are either modelled or stem from an external source. Modelled corrections include the effect of Earth's rotation, relativistic clock, Shapiro delay, phase windup, dry tropospheric zenith delay and mapping functions. For the latter the user may currently choose from Collins/GPT respective Niell mapping function/GMF. As far as external corrections are concerned, precise orbits, satellite clocks, antenna phase center offsets and variations for both satellites and receiver as well as code and phase biases are obtained. The engine currently supports the observation specific bias products as well as Widelane (WL)/Narrowlane (NL) biases (where NL biases are included in the satellite clock corrections). For more detailed information on the bias products, the reader is referred to [9] and [10].

The biases are only applicable with the corresponding processing method, therefore the MuSNAT PPP-Engine allows the user to choose between WL/NL (realized by forming the Melbourne-Wuebbena and Ionosphere-Free Linear Combination) or uncombined processing with ionospheric delay estimation. Furthermore, the EKF observation model directly supports single-differenced GNSS observations, as user-receiver related hardware delays are unknown in most of the cases (also for our USRP not group/phase delay calibration is planned). If undifferenced processing is selected and ambiguity resolution is enabled, between-satellite-single-differencing at ambiguity level is enforced to be in accordance with the required integer nature of the ambiguities.

Additionally, static and kinematic scenarios are supported by the PPP-Engine, allowing for user velocity estimation in addition to the position. Cycle-slip detection is realized by observing the behaviour of the measured widelane ambiguity over time, and upon exceeding a certain threshold all satellite phase measurements are removed. PPP processing at the moment supports GPS L1/L2 and Galileo E1/E5a.

Fig. 14 shows the difference between the retrieved float solution for a static open sky scenario with 1 second sampling rate and the reference coordinate of the marker. The PPP-filter was configured for kinematic processing, i.e. with large process noise in position/velocity domain. As the engine is still in an experimental stage, ambiguity fixing, fixing verification and cycle slip determination are subject to further investigation. However, the PPP float solution converges to cm-level within reasonable time.

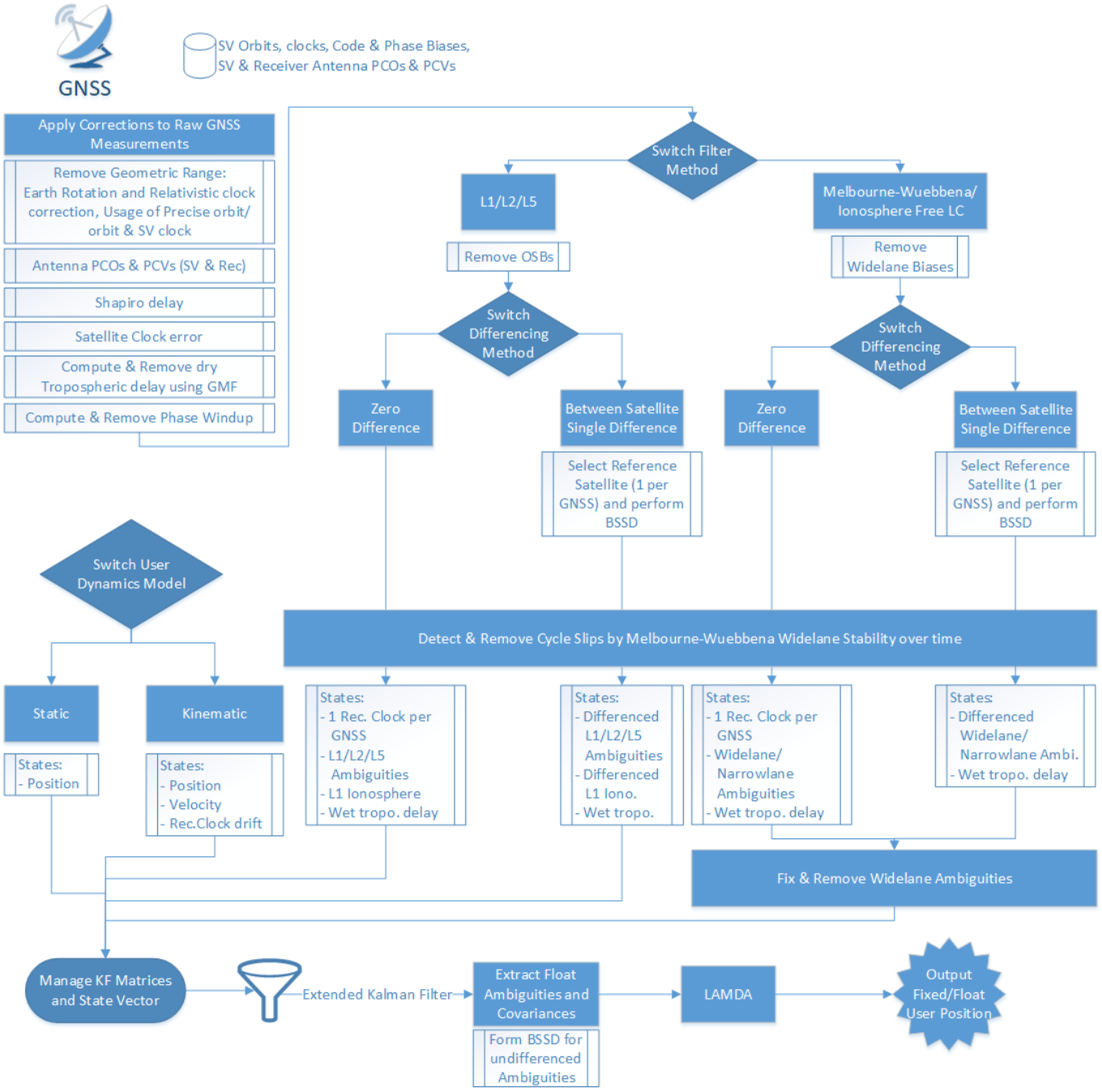

Figure 13: PPP Data/Processing Flow

2. LiDAR-Processing

Fig. 15 summarizes the LiDAR processing chain realized via the MATLAB interface as of Sept. 2021. LiDAR processing is develop within MuSNAT to further stabilize the PPP-based GNSS/INS filter. This section here focuses only on the data flow and detailed results of LiDAR processing will be published later.

The LiDAR processing consist mainly of five modules all written in MATLAB and currently it can receive data coming from a VLP-16 sensor and synthetic data generated in CARLA [11]. The LiDAR point clouds are passed from MuSNAT to the MATLAB modules with time tags that allow synchronization to GNSS related and IMU data.

The first module, ground segmentation, extracts points lying on the ground by applying a 2D line tracking algorithm once the mapping of the 3D point cloud to a 2-dimensional data set has been performed. It is based on [12] as it has been proven that it can operate in real time. Achieving a successful extraction of the ground, helps significantly the following steps of the processing

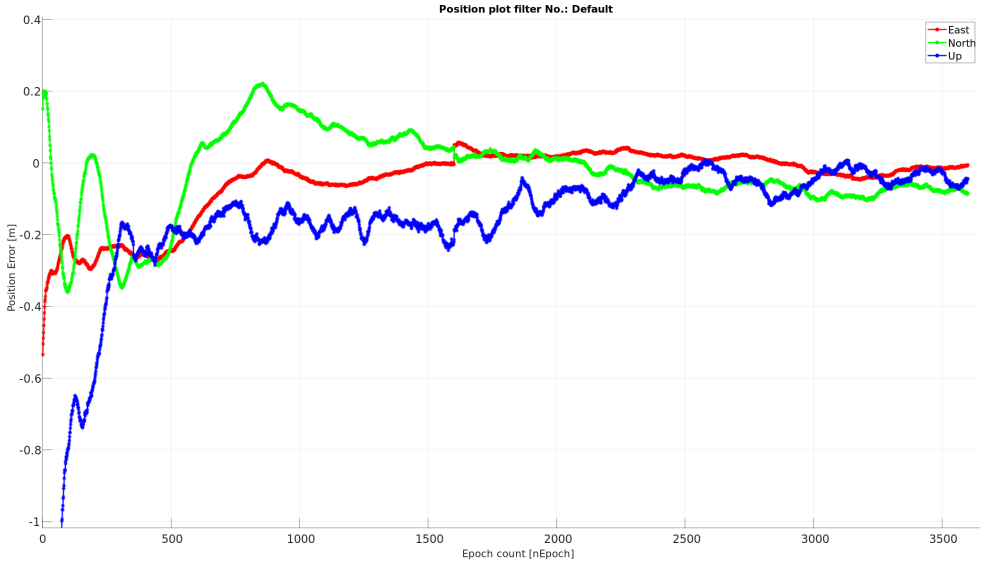

Figure 14: Static PPP convergence, GPS Galileo, Reference - Computed Coordinates, 1 Second Sampling

as it disconnects different objects from the scene, allowing a better recognition of single components.

The following module consists of two steps, generating the range images and segmenting them. Generation of range images from 3D point clouds has been lately used specifically in the field of autonomous driving [13]. We implemented a cylindrical projection, where each of the pixels stores range data. The image size is restricted in height by the number of channels that the LiDAR sensor has and the width has been set to 1200, which is compatible with many image processing algorithms and neural network architectures. For the segmentation of the range image in the third module, we chose a method based on the information that the depth of neighbors in the image can provide [14]. Through a *depth angle* threshold, one can differentiate if the laser beams have been reflected from the same object or not. Again, the method performs in real time as it works in a 2D data set and avoids complicated relationships between points at this stage.

In order to assure better results, we keep objects that can be used for navigation purposes, mainly non-moving man made artifacts like buildings, traffic lights, poles, etc. For that reason the third module is dedicated to identify and keep the features that we support and remove everything else.

Based on this final point cloud, the registration is performed within the fourth module. We choose the iterative-closest-point (ICP) algorithm from `libpointmatcher` library [15] because it offers a re-configurable ICP chain that supports different filters and metrics to stop the iterations depending on the characteristics of the data set or application in which it is being performed. This module produces the rigid transformation from which we derive the displacement and consequently the velocity between a pair of point clouds.

Finally the fifth module provides an estimate of the velocity error. To do so, the module uses the features computed previously as we assume that the quality of the the estimated displacement is directly connected to feature richness of the environment. The less features the environment has, the less accurate the registration will be. Moreover, each of the features is modeled as a multivariate Gaussian distribution whose covariance matrix is characterized based on its orientation relative to the LiDAR. And the final position error covarinace matrix is computed by combining all individual contributions. The processing of the LiDAR shows promising results for simulated data. Fig. 16 shows the reconstructed trajectory for a particular data set and Fig. 17 shows the integrity of the the velocity estimates in a Standford-like integrity plot. It shows nominal operations and that the computed protection level bounds correctly the velocity errors.

3. Deep Coupling GNSS/INS Interface

The integration of GNSS with inertial sensors is despite its long history still topic of interesting research, e.g. to provide a cost efficient and precise PVT solution together with high integrity. Various integration scheme exists, ranging from loose to deep or ultra-tight coupling. The latter ones do promise the highest potential as the data fusion is performed at the lowest level, thus

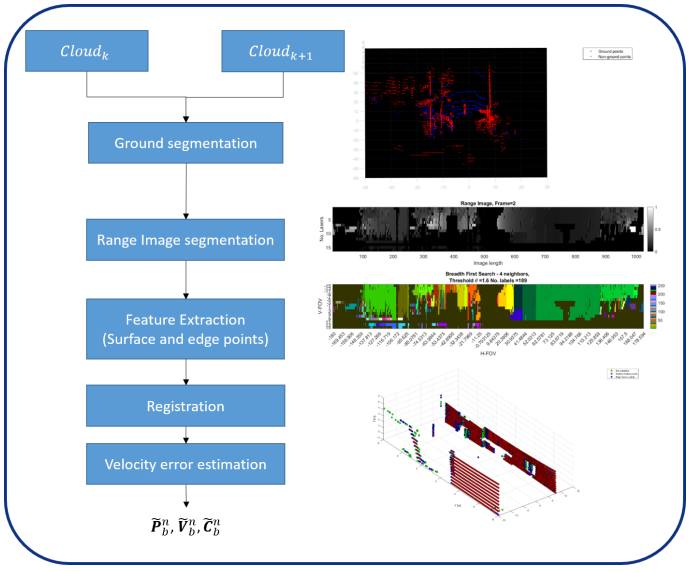

Figure 15: LiDAR Processing flow in MATLAB using the MuSNAT MATLAB/C++ interface

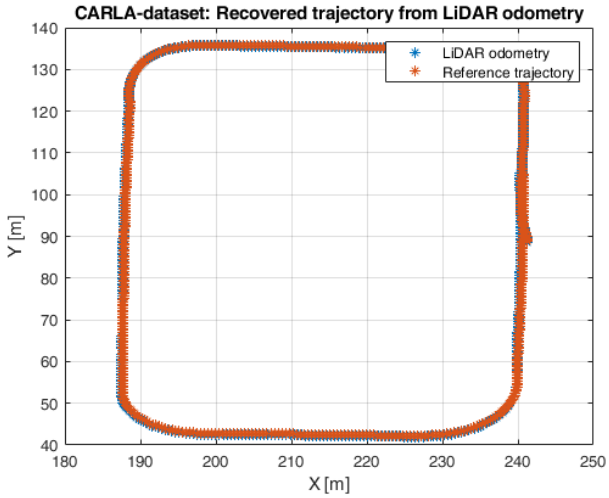

Figure 16: Constructed trajectory based on LiDAR odometry for a simulated data set

ensuring maximal mutual aiding of the GNSS signals and the inertial data [16, 17]. For deep integration, often vector tracking methods are adapted for code and Doppler [18] and even for phase tracking [19].

The algorithms work at a high rate defined by the integrate and dump cycles of the tracking channels, e.g. often 1 ms. Data from the different tracking channels arrives asynchronously at the integration filter together with the inertial measurement unit (IMU) data. The timely synchronization of the data, the coordinated filter update and the low latency requirement to achieve real-time operations renders R&D work in this area highly difficult. To ease this problem to some extend, a C++ to MATLAB interface was created in MuSNAT that uses MuSNAT for signal correlation and data management and passes the data in an optimized format to MATLAB, which is then used to realize a GNSS/INS integration filter of any kind or to prototype vector tracking algorithms.The basic concept of this interface is to transfer data from C++ to MATLAB (and vice versa) only at a low rate (e.g. every 0.5 s) and then use the concept of sufficient statistics (which will be describe below) to interpolate from this data the actual needed early, prompt and late correlation (or any other kind of GNSS data).

A naive implementation of a C++/MATLAB interface for deep GNSS/INS integration that is activated for each integrate-and-dump epoch (e.g. at a rate of 1 kHz for each GPS C/A code signal) would not be suitable in terms of usability as each call to the C++/MATLAB interface consumes a latency time of a few milliseconds.

The mathematical concept exploited to realize the interface is to compress all information contained in the GNSS signal itself, so that all further GNSS-related information can be reconstructed from it. This is known in signal detection and estimation

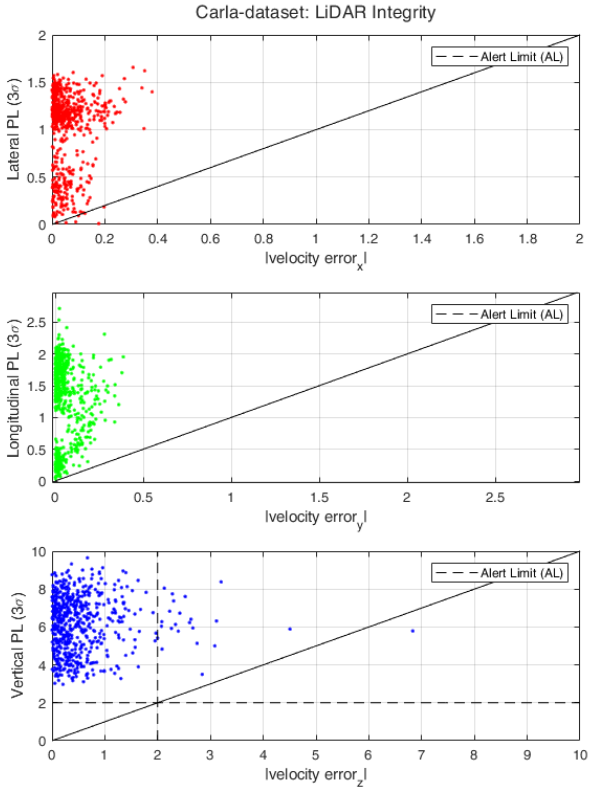

Figure 17: LiDAR Integrity for a simulated data set

theory as the concept of a *sufficient statistic*. A statistic is sufficient with respect to a statistical model and its associated unknown parameter if no other statistic that can be calculated from the same sample provides any additional information as to the value of the parameter [20]. Applying this theorem to code division multiple access (CDMA) signals in additive white Gaussian noise (AWGN), we have the option of producing a finite set of discrete sufficient statistics via correlation, or producing a finite set of approximately sufficient statistics via chip-matched filtering followed by Nyquist sampling [21]. Thus the well-known correlation values (or multi-correlator values) contain the full GNSS information of the raw IF signal, as long as the range of the correlation function covers the region of possible delay (and Doppler) values.

The concept is outlined in Fig. 18. The software receiver is switched to vector tracking and accepts a PVT solution at the measurement rate (e.g. every 0.5 s) as input to control the channel numerical controlled oscillators (NCOs) for the next integration interval [18]. Every tracking channel activates a multi-correlator and the multi-correlator values are collected over the next measurement interval e.g. 0.5 s. All correlator values are time-tagged and the NCO values used for correlation are attached. They are denoted as NCO_{VT} in the following. A typical example of a batch of multi-correlator values is shown in Fig. 20. Those batches are the sufficient statistics. They are produced for all tracked GNSS signals and passed to the MATLAB module together with the IMU data (cf. Fig. 19).

To ensure that the multi-correlator values realize the sufficient statistics, it is required that the provided PVT input to vector tracking and the range of the multi-correlator are compatible in the sense, that the true PVT is within the is range of the multi-correlator (i.e. the difference of true and vector-tracking PVT projected into the line-of-sight (LOS) direction should be smaller than the multi-correlator range in code phase and Doppler direction). It is reasonable to assume that for short integration times, e.g. 1 ms or 4 ms, the Doppler range (basically inversely proportional to the coherent integration time) is much larger than Doppler errors resulting from an error in the velocity part (often much less than a few m/s) of the PVT solution input to the vector tracking. Thus the Doppler dimension is neglected in the following.

The multi-correlator values are obtained with the NCO_{VT} values. On other hand the deep coupling (DC) algorithm maintains its own NCO values (e.g. derived from LOS projection of the DC state vector). They are denoted as NCO_{DC}. Similarly, the

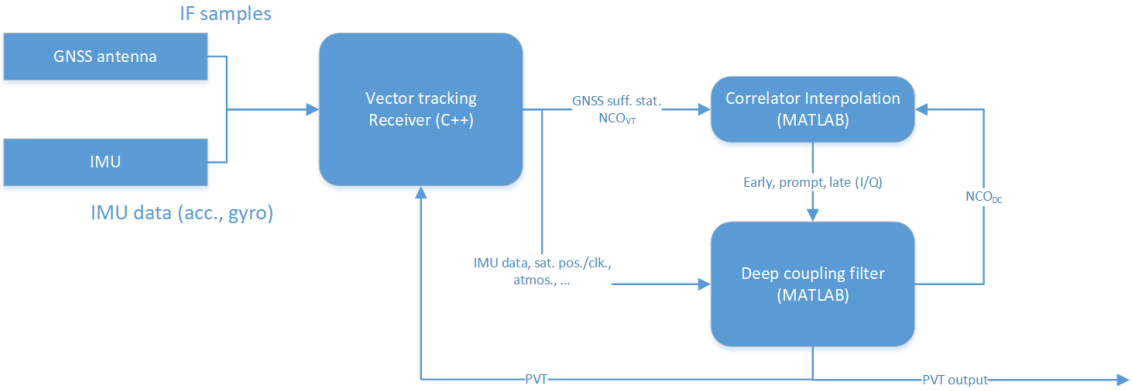

Figure 18: Block diagram and data flow for the C++ / MATLAB interface for deep coupling

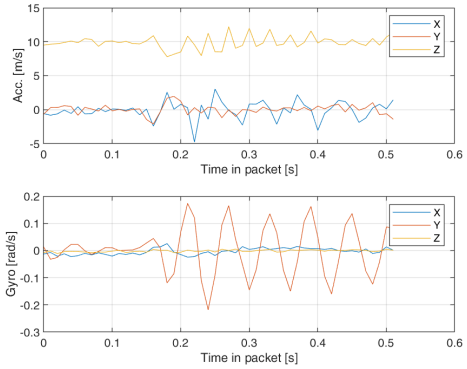

Figure 19: An exemplary batch of IMU data passed through DC interface to MATLAB (upper: accelerometer, lower: gyro)

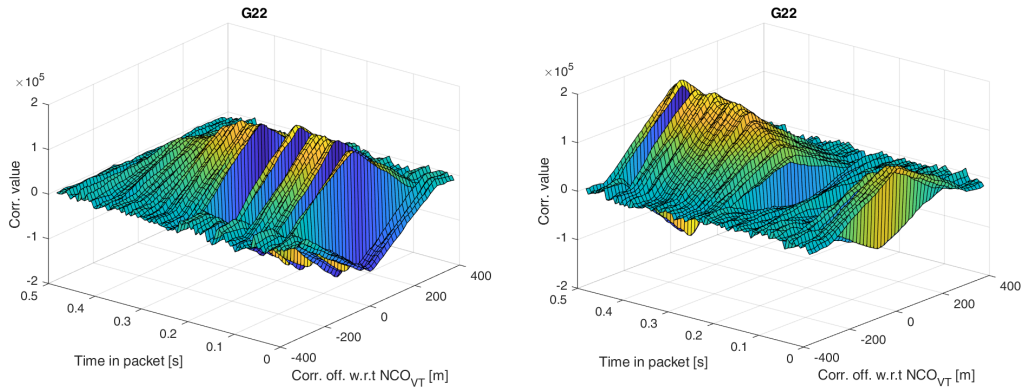

Figure 20: Multi-correlator values used to realize a sufficient statistics and passed through the DC interface to MATLAB for one GPS C/A signal (left: real part, right: imaginary part)

NCO code phase is denoted as $\tau_{VT;DC}$ and the NCO carrier phase as $\phi_{VT;DC}$. The multi-correlator values shall be denoted as $C_{VT,d}$, where d is the multi-correlator code phase offset w.r.t. to NCO_{VT}.

The concept of the sufficient statistics allows to generate correlation values for other NCO values from $C_{VT,d}$ without any information loss. Thus the DC algorithm receives as input exactly the same correlation values as it would receive from a completely synchronized DC implementation. Those values shall be donated as $C_{DC,d}$ and d indicates the offsets for e.g. early $d = -D/2$, prompt $d = 0$ and late $d = +D/2$ correlators. D is the correlator spacing. The mathematical relationship is written as

$$C_{DC,d} = \text{sinc-int}\{C_{VT}; \tau_{DC} - \tau_{VT} + d\} \exp\{-j2\pi(\phi_{VT} - \phi_{DC})\} \tag{9}$$

and 'sinc-int' denotes the sinc-interpolation method, which is well-known from the Nyquist sampling theorem. If the multi-correlator spacing is dense enough i.e. smaller than the inverse of the signal bandwidth, the full correlation function can be retrieved. This is illustrated in Fig. 21. Sinc interpolation is also discussed in Fig. 6.7 of [22].

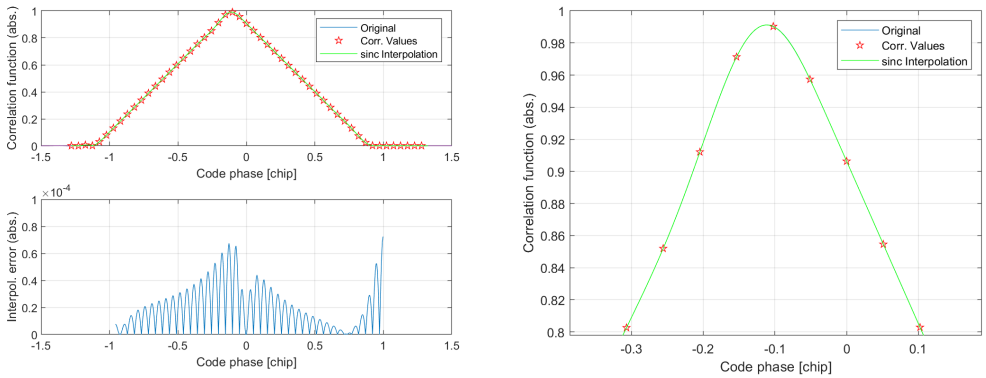

Figure 21: Illustration of sinc-Interpolation for a band-limited BPSK(1) correlation function. right: zoom onto peak from the left upper figure. The interpolation is performed within the 'Corr. Values' obtained from the original correlation function and shown as red stars. The difference between original and interpolated function is barely visible in the plots and shown directly in the lower left part.

To validate the concept, the DC interface was used to realize a standard DLL/PLL tracking scheme in MATLAB. The code and phase pseudorange can then be compared to ones which are produced by MuSNAT in standard tracking mode having the same settings (early-late tracking of a GPS C/A code signal, D=200 ns, 2nd order DLL and PLL, T_{coh}=1 ms, B_{DLL}=1 Hz, B_{PLL}=9 Hz). The results (code minus carrier (CMC) was chosen for visualization) shown in Fig. 22 demonstrate that the standard tracking implementation in MuSNAT and the MATLAB implementation via the DC interface show basically the same results. The CMC plot derived from the vector tracking mode alone is shown in Fig. 23 and exhibits for the same period as for Fig. 22 the expected discontinuous behaviour resulting from the vector tracking PVT input every 0.5 s. The interpolation scheme (9) removes the discontinuities and the resulting code and phase pseudoranges are again smooth.

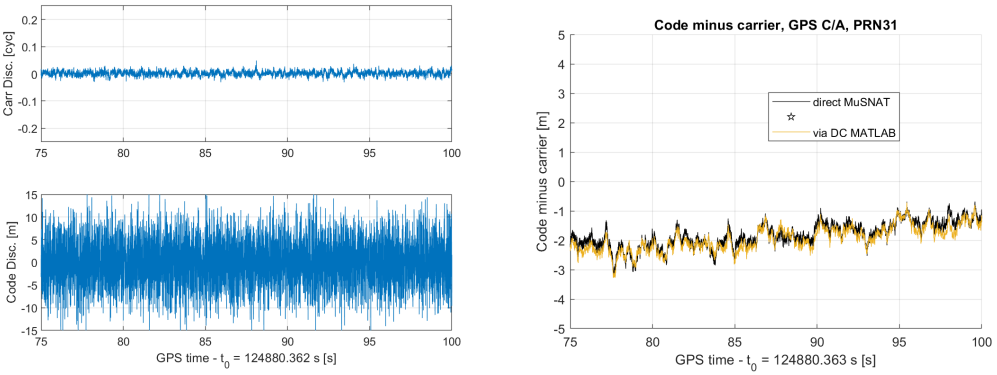

Figure 22: left: code and phase discriminator while DLL/PLL tracking GPS C/A - PRN31 via the DC-MATLAB interface, right: CMC plots for native MuSNAT-tracking and tracking via DC-MATLAB interface

Currently, we do not yet have a full DC implementation exploiting this interface and this is a topic for future work. Finally, it should be noted that the shear processing power of conventional PCs does allow to compute the sufficient statistics fast enough even for a real-time operation. The transfer of the data to MATLAB (e.g. every 0.5 s) does consume only a short time (appox. 10-15 ms). It is very likely that the DC implementation can also be done efficiently so that at least near real-time processing is possible.

VI. PROCESSING PERFORMANCE ON DESKTOP PCS

The development of the Core of MuSNAT goes back to times of single core CPUs without real parallelism for processing different signals of different sources and frequencies and their respective components. Nowadays multi-core CPUs are fundamental

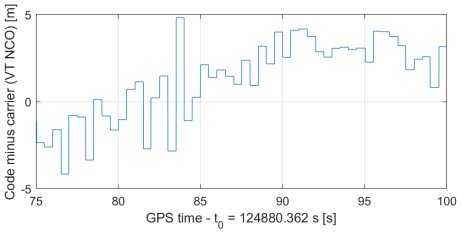

Figure 23: CMC plots corresponding to Fig. 22 and based on NCO_{VT}, i.e. $\tau_{VT} - \lambda\phi_{VT}$)

for energy efficiency and rising processing demands of applications. One prominent example for modern many-core CPUs is the AMD Threadripper; it features up to 64 cores [23]. In order to make applications like real-time multi-antenna correlator processing possible on CPUs, MuSNAT was profiled and optimized for the Threadripper architecture. Fig. 24 shows the timing diagram of MuSNAT running on an AMD Ryzen Threadripper 3970X with 32 cores, tracking 40 C/A code signals with 80 MHz sampling rate from 40 different signal streams. This special configuration is coming from the requirements of an encrypted chip retrieval application [24]. Ideally the PRNs from several in view satellites can be retrieved in parallel from a large antenna array together with MuSNAT. Contrary to the behaviour with only a few streams, additionally to the main work of replica generation and correlations the stream reading and processing gets more dominant and had to get parallelized. Fig. 25 compares the effective CPU utilisation with two different stream packet sizes. The lower histogram shows that with longer packets the CPU gets into a state when all 40 channels are working in parallel and sometimes for reading and conversion with additional threads more than 60 logical cores are used (on average above 38). In the smaller packet case (upper plot) only 26 cores are used on average. Tracking several satellites per antenna stream or frequency band will decrease the thread waiting times because of more work to be performed per packet and allow efficient use of the whole CPU capabilities.

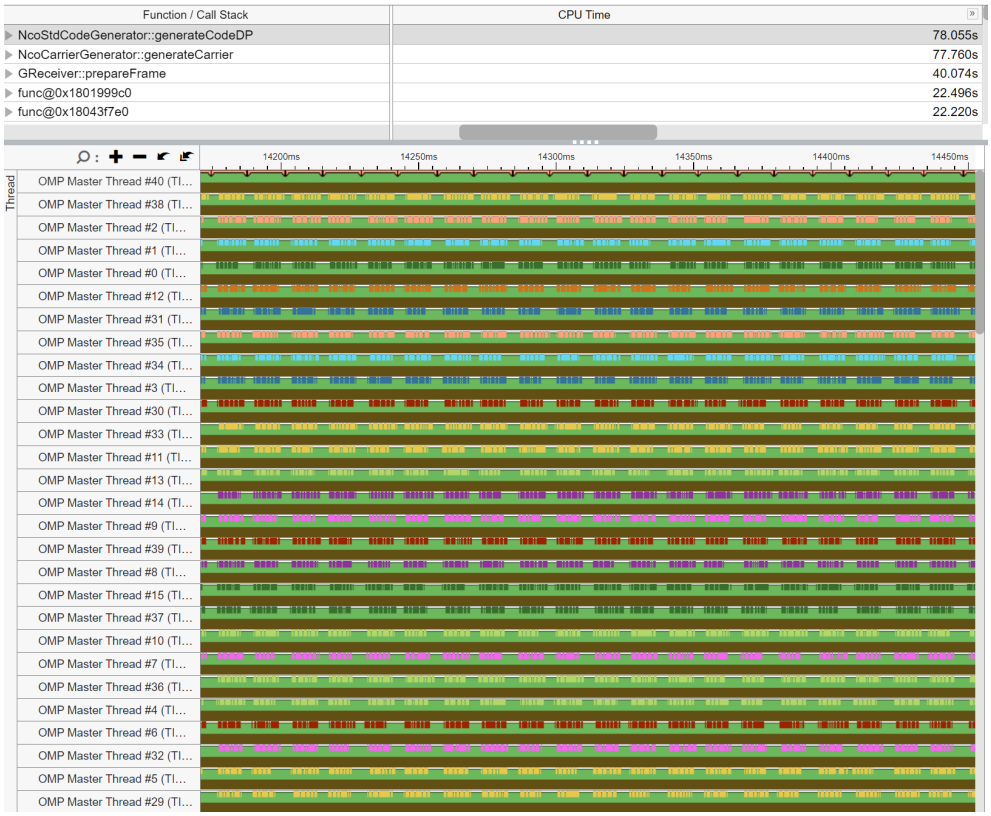

Figure 24: Intel VTune 2021.2.0 timing diagram of MuSNAT ASCII-version on a Ryzen 3970X tracking 40 GPS C/A signals in post-processing

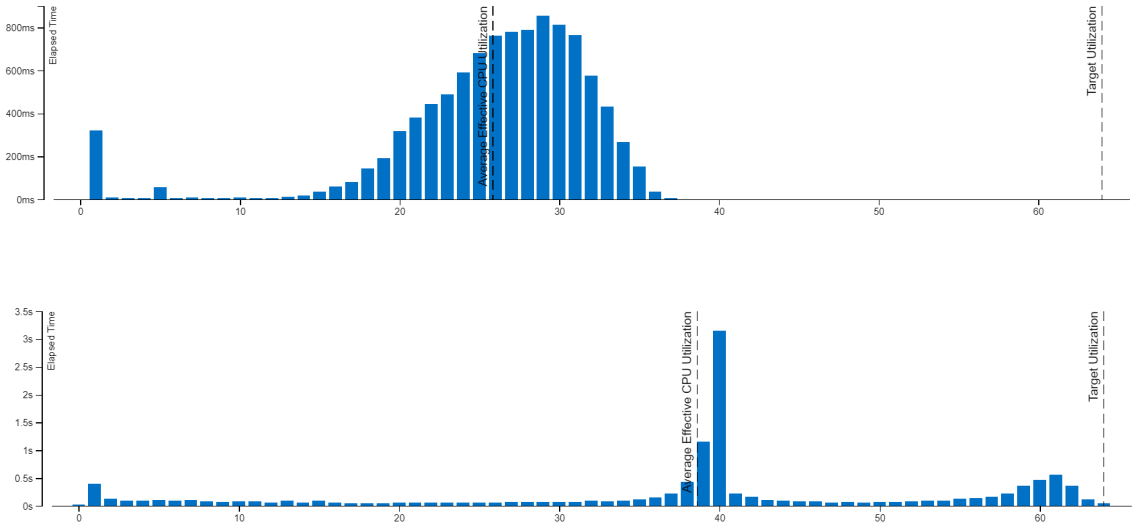

Figure 25: Intel VTune 2021.2.0 logical core usage histograms of MuSNAT ASCII-version on a Ryzen 3970X tracking 40 GPS C/A signals in post-processing (upper: short sample packages, lower: 10 times longer packages

VII. FUTURE WORK

In terms of low level receiver software extension, two points shall be mentioned here.

For the USRP 2974, the extension to more frequency bands and lower (!) native bit width requires a FPGA firmware modification of the USRP 2974 and is currently ongoing. With it 2, 4 and 8-bit will be supported. Regarding the transfer from USRP to the PC, the compressed data format of [25] might be considered. Furthermore, a firmware extension the support of the connection of up to two inertial measurement units (IMUs) over the general-purpose input/output (GPIO) interface of the USRP, which guaranties a synchronization with the on-board clock of the USRP is planned. Our setup will support two types of IMUs: a) XSens MTi-G-710 and b) iFOG-IMU-1-A.

Porting the signal correlation for tracking from the CPU to the GPU is still ongoing research. In [1] the latency when calling GPU functions was identified as the major bottleneck. Using a similar methodology like for the deep coupling interface, this latency problem could be solved.

Needless to say, that the use of the software receiver in upcoming R&D projects is also part of the future work.

ACKNOWLEDGEMENTS

The results presented in this work were developed within the projects *"Galileo FUSION and OPA3L"* funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) and administered by the Project Management Agency for Aeronautics Research of the German Space Agency (DLR) in Bonn, Germany (grant no. 50NA2001 and 50NA1910).

REFERENCES

[1] T. Pany, D. Dötterböck, H. Gomez-Martinez, M. S. Hammed, F. Hörkner, T. Kraus, D. Maier, D. Sanchez-Morales, A. Schütz, P. Klima, and D. Ebert, "The multi-sensor navigation analysis tool (MuSNAT) – architecture, LiDAR, GPU/CPU GNSS signal processing," in *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*. Institute of Navigation, Oct. 2019. [Online]. Available: https://doi.org/10.33012/2019.17128

[2] "USRP-2974 webpage from Ettus Research," https://kb.ettus.com/USRP-2974, 2021-07-16.

[3] Ettus Research, "Converting an X310 into an NI-USRP Rio," 22.1.2021. [Online]. Available: https://kb.ettus.com/Converting_an_X310_into_an_NI-USRP_Rio

[4] T. Kraus, T. Pany, and B. Eissfeller, "Maximum Theoretical Interference Mitigation Capability of a GNSS Receiver as Limited by the GNSS Frontend," in *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*, ser. ION GNSS+, The International Technical Meeting of the Satellite Division of The Institute of Navigation. Institute of Navigation, 2017, pp. 3471–3480.

[5] Intel Corp., "Intel VTune Profiler," 2021. [Online]. Available: https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/vtune-profiler.html#gs.bxxpg8

[6] T. Pany, *Navigation Signal Processing for GNSS Software Receivers*. Artech House, 2010.

[7] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.211, 04 2017, version 14.2.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2425

[8] C. Yang, T. Pany, and P. Weitkemper, "Effect of Antenna Ports on TOA Estimation with 4G LTE Signals in Urban Mobile Environments," in *Proceedings of the 33rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2020)*, ser. ION GNSS+, The International Technical Meeting of the Satellite Division of The Institute of Navigation. Institute of Navigation, 2020, pp. 2166–2181.

[9] G. Katsigianni, S. Loyer, and F. Perosanz, "Ppp and ppp-ar kinematic post-processed performance of gps-only, galileo-only and multi-gnss," *Remote Sensing*, vol. 11, no. 21, 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/21/2477

[10] S. Schaer, A. Villiger, D. Arnold, R. Dach, L. Prange, and A. Jäggi, "The CODE ambiguity-fixed clock and phase bias analysis products: generation, properties, and performance," *Journal of Geodesy*, vol. 95, no. 7, p. 81, 2021.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[12] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 2010, pp. 560–565.

[13] T. Wu, H. Fu, B. Liu, H. Xue, R. Ren, and Z. Tu, "Detailed analysis on generating the range image for lidar point cloud processing," *Electronics*, vol. 10, no. 11, p. 1224, 2021.

[14] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3d laser scans," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 41–52, 2017.

[15] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[16] M. Lashley and D. M. Bevly, "Performance comparison of deep integration and tight coupling," *NAVIGATION, Journal of the Institute of Navigation*, vol. 60, no. 3, pp. 159–178, 2013.

[17] T. Pany, R. Kaniuth, and B. Eissfeller, "Deep integration of navigation solution and signal processing," in *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2005)*, 2005, pp. 1095–1102.

[18] T. Pany and B. Eissfeller, "Use of a vector delay lock loop receiver for gnss signal power analysis in bad signal conditions," in *Proceedings of IEEE/ION PLANS 2006*, 2006, pp. 893–903.

[19] S. Chen and Y. Gao, "Improvement of carrier phase tracking in high dynamics conditions using an adaptive joint vector tracking architecture," *GPS Solutions*, vol. 23, no. 1, pp. 1–10, 2019.

[20] R. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 222, no. 594-604, pp. 309–368, Jan. 1922.

[21] P. Fearnhead, "Markov chain monte carlo, sufficient statistics, and particle filters," *Journal of Computational and Graphical Statistics*, vol. 11, no. 4, pp. 848–862, 2002.

[22] J. Dampf, "Probability analysis for bayesian directposition estimation," dissertation, Graz University of Technology, 2021.

[23] "AMD Ryzen Threadripper Processors," https://www.amd.com/de/products/ryzen-threadripper.html, 2021-07-16.

[24] D. Dötterböck, M. S. Hammed, T. Pany, R. Lesjak, and T. Prechtl, "Retrieval of encrypted prn sequences via a self-calibrating 40-element low-cost antenna array: Demonstration of proof-of-concept)," in *Proceedings of the 33nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*. Institute of Navigation, 2020. [Online]. Available: https://doi.org/10.33012/2020.17679

[25] Z. Clements, P. Iannucci, T. Humphreys, and T. Pany, "Optimized Bitpacking for Software-Defined GNSS," in *Proceedings of the 34rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2021)*, ser. ION GNSS+, The International Technical Meeting of the Satellite Division of The Institute of Navigation. Institute of Navigation, 2021.